

ISO/IEC JTC 1/SC 27**Information security, cybersecurity and privacy protection****Secretariat: DIN (Germany)**

Document type:	Text for DIS ballot
Title:	Text_f_DIS_ISO-IEC_15408-3 - Information security, cybersecurity and privacy protection — Evaluation criteria for IT security — Part 3: Security assurance components
Status:	In accordance with Recommendation 16 (contained in SC 27 N20053) of the 59 th SC 27/WG 3 meeting held in Paris, France, 14 th – 18 th October 2019, this document was sent to the ISO Central Secretariat (ISO/CS) for a 12 week 1 st DIS ballot. It is circulated within SC 27 for information.
Date of document:	2020-03-27
Source:	Project editor/Co-editor (Christian Noetzel, Soohyeun Lee)
Expected action:	INFO
No. of pages:	1 + 232
Email of secretary:	krystyna.passia@din.de
Committee URL:	https://isotc.iso.org/livelink/livelink/open/jtc1sc27

ISO/IEC 15408-3:XXXX(E)

ISO/IEC JTC 1/SC 27/WG 3 N1654

Secretariat: DIN

**Information security, cybersecurity and privacy protection —
Evaluation criteria for IT security — Part 3: Security assurance
components**

CD stage

Warning for WDs and CDs

This document is not an ISO International Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an International Standard.

Recipients of this draft are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

© ISO 2017, Published in Switzerland

All rights reserved. It is permitted to download this electronic file, to make a copy and to print out the content for the purpose of preparing ISO and IEC documents only. You may not copy or “mirror” the file, or any part of it, for any other purpose without permission in writing from the publishers.

International Organization for Standardization International Electrotechnical Commission

Chemin de Blandonnet 8

Case postale 401

CH-1214 Vernier, Geneva

Tel. + 41 22 749 01 11

Fax + 41 22 733 34 30

3, rue de Varembé

Case postale 131

CH-1211 Geneva 20

Tel. + 41 22 919 02 11

Fax + 41 22 919 03 00

E-mail central@iso.org E-mail inmail@iec.org

Web www.iso.org Web www.iec.org

1	Contents	Page
2	1 Scope	1
3	2 Normative references	1
4	3 Terms and definitions, symbols and abbreviated terms.....	1
5	4 Overview	1
6	5 Assurance paradigm	2
7	5.1 Introduction	2
8	5.2 ISO/IEC 15408 philosophy	2
9	5.3 Assurance approach	2
10	5.3.1 Introduction	2
11	5.3.2 Significance of vulnerabilities.....	2
12	5.3.3 Cause of vulnerabilities	3
13	5.3.4 ISO/IEC 15408 series assurance	3
14	5.3.5 Assurance through evaluation.....	3
15	5.4 ISO/IEC 15408 series evaluation assurance scale	4
16	6 Security assurance components.....	4
17	6.1 Introduction	4
18	6.2 Assurance class structure.....	4
19	6.2.1 Class name.....	4
20	6.2.2 Class introduction	5
21	6.2.3 Assurance families	5
22	6.3 Assurance family structure.....	5
23	6.3.1 Family name	5
24	6.3.2 Objectives	6
25	6.3.3 Component levelling.....	6
26	6.3.4 Application notes.....	6
27	6.3.5 Assurance components.....	6
28	6.4 Assurance component structure	6
29	6.4.1 Introduction	6
30	6.4.2 Component identification.....	7
31	6.4.3 Objectives	7
32	6.4.4 Application notes.....	7
33	6.4.5 Dependencies.....	7
34	6.4.6 Assurance elements	8
35	6.5 Assurance elements	9
36	6.6 Component taxonomy	9
37	7 Class APE: Protection Profile evaluation.....	9
38	7.1 Introduction	9
39	7.2 PP introduction (APE_INT).....	10
40	7.2.1 Objectives	10
41	7.2.2 APE_INT.1 PP introduction.....	10
42	7.3 Conformance claims (APE_CCL).....	10
43	7.3.1 Objectives	10
44	7.3.2 APE_CCL.1 Conformance claims.....	11
45	7.4 Security problem definition (APE_SPD)	13
46	7.4.1 Objectives	13

47	7.4.2	APE_SPD.1 Security problem definition	13
48	7.5	Security objectives (APE_OBJ)	14
49	7.5.1	Objectives	14
50	7.5.2	Component levelling.....	14
51	7.5.3	APE_OBJ.1 Security objectives for the operational environment.....	14
52	7.5.4	APE_OBJ.2 Security objectives.....	15
53	7.6	Extended components definition (APE_ECD)	16
54	7.6.1	Objectives	16
55	7.6.2	APE_ECD.1 Extended components definition	16
56	7.7	Security requirements (APE_REQ)	17
57	7.7.1	Objectives	17
58	7.7.2	Component levelling.....	17
59	7.7.3	APE_REQ.1 Direct rationale PP-Module security requirements.....	17
60	7.7.4	APE_REQ.2 Derived security requirements.....	18
61	8	Class ACE: Protection Profile Configuration evaluation	20
62	8.1	Introduction	20
63	8.2	PP-Module introduction (ACE_INT).....	21
64	8.2.1	Objectives	21
65	8.2.2	ACE_INT.1 PP-Module introduction	21
66	8.3	PP-Module conformance claims (ACE_CCL).....	22
67	8.3.1	Objectives	22
68	8.3.2	ACE_CCL.1 PP-Module conformance claims.....	22
69	8.4	PP-Module security problem definition (ACE_SPD)	24
70	8.4.1	Objectives	24
71	8.4.2	ACE_SPD.1 PP-Module Security problem definition.....	24
72	8.5	PP-Module security objectives (ACE_OBJ)	25
73	8.5.1	Objectives	25
74	8.5.2	Component levelling.....	25
75	8.5.3	ACE_OBJ.1 Direct Rationale PP-Module security objectives.....	25
76	8.5.4	ACE_OBJ.2 PP-Module Security objectives	26
77	8.6	PP-Module extended components definition (ACE_ECD)	27
78	8.6.1	Objectives	27
79	8.6.2	ACE_ECD.1 PP-Module extended components definition.....	27
80	8.7	PP-Module security requirements (ACE_REQ)	28
81	8.7.1	Objectives	28
82	8.7.2	Component levelling.....	28
83	8.7.3	ACE_REQ.1 PP-Module stated security requirements	28
84	8.7.4	ACE_REQ.2 PP-Module derived security requirements	30
85	8.8	PP-Module consistency (ACE_MCO).....	31
86	8.8.1	Objectives	31
87	8.8.2	ACE_MCO.1 PP-Module consistency.....	31
88	8.9	PP-Configuration consistency (ACE_CCO)	33
89	8.9.1	Objectives	33
90	8.9.2	ACE_CCO.1 PP-Configuration consistency	33
91	9	Class ASE: Security Target evaluation	36
92	9.1	Introduction	36
93	9.2	ST introduction (ASE_INT).....	37
94	9.2.1	Objectives	37
95	9.2.2	ASE_INT.1 ST introduction.....	37
96	9.3	Conformance claims (ASE_CCL)	38
97	9.3.1	Objectives	38
98	9.3.2	ASE_CCL.1 Conformance claims	38

99	9.4	Security problem definition (ASE_SPD).....	41	
100	9.4.1	Objectives	41	
101	9.4.2	ASE_SPD.1 Security problem definition.....	41	
102	9.5	Security objectives (ASE_OBJ)	41	
103	9.5.1	Objectives	41	
104	9.5.2	Component levelling.....	42	
105	9.5.3	ASE_OBJ.1 Direct rationale Security objectives for the operational environment	42	
106	9.5.4	ASE_OBJ.2 Security objectives	42	
107	9.6	Extended components definition (ASE_ECD)	43	
108	9.6.1	Objectives	43	
109	9.6.2	ASE_ECD.1 Extended components definition	44	
110	9.7	Security requirements (ASE_REQ).....	45	
111	9.7.1	Objectives	45	
112	9.7.2	Component levelling.....	45	
113	9.7.3	ASE_REQ.1 Direct Rationale Stated security requirements.....	45	
114	9.7.4	ASE_REQ.2 Derived security requirements.....	47	
115	9.8	TOE summary specification (ASE_TSS)	48	
116	9.8.1	Objectives	48	
117	9.8.2	Component levelling.....	48	
118	9.8.3	ASE_TSS.1 TOE summary specification	48	
119	9.8.4	ASE_TSS.2 TOE summary specification with architectural design summary	49	
120	9.9	Consistency of composite product Security Target (ASE_COMP)	50	
121	9.9.1	Objectives	50	
122	9.9.2	ASE_COMP.1 Consistency of Security Target	50	50
123	10	Class ADV: Development	53	
124	10.1	Introduction	53	
125	10.2	Security Architecture (ADV_ARC)	58	
126	10.2.1	Objectives	58	
127	10.2.2	Component levelling.....	58	
128	10.2.3	Application notes	58	
129	10.2.4	ADV_ARC.1 Security architecture description.....	59	
130	10.3	Functional specification (ADV_FSP)	60	
131	10.3.1	Objectives	60	
132	10.3.2	Component levelling.....	60	
133	10.3.3	Application notes	60	
134	10.3.4	ADV_FSP.1 Basic functional specification	63	
135	10.3.5	ADV_FSP.2 Security-enforcing functional specification.....	64	
136	10.3.6	ADV_FSP.3 Functional specification with complete summary	65	
137	10.3.7	ADV_FSP.4 Complete functional specification.....	66	
138	10.3.8	ADV_FSP.5 Complete semi-formal functional specification with additional error		
139		information.....	67	
140	10.3.9	ADV_FSP.6 Complete semi-formal functional specification with additional formal		
141		specification	68	
142	10.4	Implementation representation (ADV_IMP).....	69	
143	10.4.1	Objectives	69	
144	10.4.2	Component levelling.....	70	
145	10.4.3	Application notes	70	
146	10.4.4	ADV_IMP.1 Implementation representation of the TSF	71	
147	10.4.5	ADV_IMP.2 Complete mapping of the implementation representation of the TSF	72	
148	10.5	TSF internals (ADV_INT)	72	
149	10.5.1	Objectives	72	
150	10.5.2	Component levelling.....	73	

151	10.5.3 Application notes	73
152	10.5.4 ADV_INT.1 Well-structured subset of TSF internals	73
153	10.5.5 ADV_INT.2 Well-structured internals	74
154	10.5.6 ADV_INT.3 Minimally complex internals	75
155	10.6 Security policy modelling (ADV_SPM)	76
156	10.6.1 Objectives	76
157	10.6.2 Component levelling	76
158	10.6.3 Application notes	77
159	10.6.4 ADV_SPM.1 Formal TOE security policy model	78
160	10.7 TOE design (ADV_TDS)	79
161	10.7.1 Objectives	79
162	10.7.2 Component levelling	80
163	10.7.3 Application notes	80
164	10.7.4 ADV_TDS.1 Basic design	81
165	10.7.5 ADV_TDS.2 Architectural design	82
166	10.7.6 ADV_TDS.3 Basic modular design	83
167	10.7.7 ADV_TDS.4 Semiformal modular design	85
168	10.7.8 ADV_TDS.5 Complete semiformal modular design	86
169	10.7.9 ADV_TDS.6 Complete semiformal modular design with formal high-level design presentation	87
170		
171	10.8 Composite design compliance (ADV_COMP)	89
172	10.8.1 Objectives	89
173	10.8.2 Component levelling	89
174	10.8.3 Application notes	89
175	10.8.4 ADV_COMP.1 Design compliance with the platform certification report, guidance and ETR_COMP	90
176	11 Class AGD: Guidance documents	91
177	11.1 Introduction	91
178	11.2 Operational user guidance (AGD_OPE)	91
179	11.2.1 Objectives	91
180	11.2.2 Component levelling	92
181	11.2.3 Application notes	92
182	11.2.4 AGD_OPE.1 Operational user guidance	92
183	11.3 Preparative procedures (AGD_PRE)	93
184	11.3.1 Objectives	93
185	11.3.2 Component levelling	94
186	11.3.3 Application notes	94
187	11.3.4 AGD_PRE.1 Preparative procedures	94
188	12 Class ALC: Life-cycle support	95
189	12.1 Introduction	95
190	12.2 CM capabilities (ALC_CMC)	96
191	12.2.1 Objectives	96
192	12.2.2 Component levelling	96
193	12.2.3 Application notes	96
194	12.2.4 ALC_CMC.1 Labelling of the TOE	97
195	12.2.5 ALC_CMC.2 Use of the CM system	98
196	12.2.6 ALC_CMC.3 Authorisation controls	99
197	12.2.7 ALC_CMC.4 Production support, acceptance procedures and automation	101
198	12.2.8 ALC_CMC.5 Advanced support	104
199	12.3 CM scope (ALC_CMS)	108
200	12.3.1 Objectives	108
201	12.3.2 Component levelling	108
202	12.3.3 Application notes	108

203	12.3.4 ALC_CMS.1 TOE CM coverage	108	
204	12.3.5 ALC_CMS.2 Parts of the TOE CM coverage	109	
205	12.3.6 ALC_CMS.3 Implementation representation CM coverage.....	110	
206	12.3.7 ALC_CMS.4 Problem tracking CM coverage	110	
207	12.3.8 ALC_CMS.5 Development tools CM coverage.....	111	
208	12.4 Delivery (ALC_DEL)	112	
209	12.4.1 Objectives	112	
210	12.4.2 Component levelling.....	113	
211	12.4.3 Application notes	113	
212	12.4.4 ALC_DEL.1 Delivery procedures	113	
213	12.5 Developer environment security (ALC_DVS)	114	
214	12.5.1 Objectives	114	
215	12.5.2 Component levelling.....	114	
216	12.5.3 Application notes	114	
217	12.5.4 ALC_DVS.1 Identification of security controls	114	
218	12.5.5 ALC_DVS.2 Sufficiency of security controls.....	115	
219	12.6 Flaw remediation (ALC_FLR)	116	
220	12.6.1 Objectives	116	
221	12.6.2 Component levelling.....	116	
222	12.6.3 Application notes	116	
223	12.6.4 ALC_FLR.1 Basic flaw remediation	116	
224	12.6.5 ALC_FLR.2 Flaw reporting procedures.....	117	
225	12.6.6 ALC_FLR.3 Systematic flaw remediation	118	
226	12.7 Development Life-cycle definition (ALC_LCD).....	120	
227	12.7.1 Objectives	120	
228	12.7.2 Component levelling.....	120	
229	12.7.3 Application notes	121	
230	12.7.4 ALC_LCD.1 Developer defined life-cycle processes	121	
231	12.7.5 ALC_LCD.2 Measurable life-cycle model	122	
232	12.8 TOE Development Artifacts (ALC_TDA)	123	
233	12.8.1 Objectives	123	
234	12.8.2 Component levelling.....	124	
235	12.8.3 Application notes	124	
236	12.8.4 ALC_TDA.1 Uniquely identifying implementation representation.....	124	
237	12.8.5 ALC_TDA.2 Matching CMS scope of implementation representation.....	126	
238	12.8.6 ALC_TDA.3 Regenerate TOE with well-defined development tools.....	129	
239	12.9 Tools and techniques (ALC_TAT)	132	
240	12.9.1 Objectives	132	
241	12.9.2 Component levelling.....	132	
242	12.9.3 Application notes	132	
243	12.9.4 ALC_TAT.1 Well-defined development tools.....	132	
244	12.9.5 ALC_TAT.2 Compliance with implementation standards.....	133	
245	12.9.6 ALC_TAT.3 Compliance with implementation standards - all parts.....	134	
246	12.10 Integration of composition parts and consistency check of delivery procedures		
247	(ALC_COMP)	135	
248	12.10.1	Objectives	135
249	12.10.2	Component levelling	135
250	12.10.3	ALC_COMP.1	Integ
251	13 Class ATE: Tests.....	136	
252	13.1 Introduction	136	
253	13.2 Coverage (ATE_COV)	137	
254	13.2.1 Objectives	137	

255	13.2.2 Component levelling.....	137
256	13.2.3 Application notes.....	137
257	13.2.4 ATE_COV.1 Evidence of coverage	137
258	13.2.5 ATE_COV.2 Analysis of coverage.....	138
259	13.2.6 ATE_COV.3 Rigorous analysis of coverage	139
260	13.3 Depth (ATE_DPT)	140
261	13.3.1 Objectives	140
262	13.3.2 Component levelling.....	140
263	13.3.3 Application notes.....	140
264	13.3.4 ATE_DPT.1 Testing: basic design.....	140
265	13.3.5 ATE_DPT.2 Testing: security enforcing modules	141
266	13.3.6 ATE_DPT.3 Testing: modular design.....	142
267	13.3.7 ATE_DPT.4 Testing: implementation representation	143
268	13.4 Functional tests (ATE_FUN).....	144
269	13.4.1 Objectives	144
270	13.4.2 Component levelling.....	144
271	13.4.3 Application notes.....	144
272	13.4.4 ATE_FUN.1 Functional testing	144
273	13.4.5 ATE_FUN.2 Ordered functional testing	145
274	13.5 Independent testing (ATE_IND).....	146
275	13.5.1 Objectives	146
276	13.5.2 Component levelling.....	146
277	13.5.3 Application notes.....	146
278	13.5.4 ATE_IND.1 Independent testing - conformance	147
279	13.5.5 ATE_IND.2 Independent testing - sample.....	148
280	13.5.6 ATE_IND.3 Independent testing - complete.....	149
281	13.6 Composite functional testing (ATE_COMP)	150
282	13.6.1 Objectives	150
283	13.6.2 Application notes.....	151
284	13.6.3 ATE_COMP.1 Composite product functional testing	151
285	14 Class AVA: Vulnerability assessment.....	152
286	14.1 Introduction	152
287	14.2 Application notes.....	152
288	14.3 Vulnerability analysis (AVA_VAN)	153
289	14.3.1 Objectives	153
290	14.3.2 Component levelling.....	153
291	14.3.3 AVA_VAN.1 Vulnerability survey.....	153
292	14.3.4 AVA_VAN.2 Vulnerability analysis.....	154
293	14.3.5 AVA_VAN.3 Focused vulnerability analysis.....	155
294	14.3.6 AVA_VAN.4 Methodical vulnerability analysis	157
295	14.3.7 AVA_VAN.5 Advanced methodical vulnerability analysis	158
296	14.4 Composite vulnerability assessment (AVA_COMP)	159
297	14.4.1 Objectives	159
298	14.4.2 AVA_COMP.1 Composite product vulnerability assessment	160
299	15 Class ACO: Composition.....	161
300	15.1 Introduction	161
301	15.2 Composition rationale (ACO_COR)	164
302	15.2.1 Objectives	164
303	15.2.2 Component levelling.....	164
304	15.2.3 ACO_COR.1 Composition rationale	164
305	15.3 Development evidence (ACO_DEV)	164
306	15.3.1 Objectives	164

307	15.3.2 Component levelling.....	164
308	15.3.3 Application notes.....	165
309	15.3.4 ACO_DEV.1 Functional Description	165
310	15.3.5 ACO_DEV.2 Basic evidence of design.....	166
311	15.3.6 ACO_DEV.3 Detailed evidence of design	167
312	15.4 Reliance of dependent component (ACO_REL)	168
313	15.4.1 Objectives	168
314	15.4.2 Component levelling.....	168
315	15.4.3 Application notes.....	168
316	15.4.4 ACO_REL.1 Basic reliance information.....	169
317	15.4.5 ACO_REL.2 Reliance information	169
318	15.5 Composed TOE testing (ACO_CTT).....	170
319	15.5.1 Objectives	170
320	15.5.2 Component levelling.....	170
321	15.5.3 Application notes.....	170
322	15.5.4 ACO_CTT.1 Interface testing	171
323	15.5.5 ACO_CTT.2 Rigorous interface testing.....	172
324	15.6 Composition vulnerability analysis (ACO_VUL).....	173
325	15.6.1 Objectives	173
326	15.6.2 Component levelling.....	173
327	15.6.3 Application notes.....	173
328	15.6.4 ACO_VUL.1 Composition vulnerability review	174
329	15.6.5 ACO_VUL.2 Composition vulnerability analysis.....	175
330	15.6.6 ACO_VUL.3 Enhanced-Basic Composition vulnerability analysis.....	175
331	Annex A (informative) Development (ADV)	177
332	A.1 ADV_ARC: Supplementary material on security architectures	177
333	A.1.1 Security architecture properties	177
334	A.1.2 Security architecture descriptions.....	178
335	A.2 ADV_FSP: Supplementary material on functional specification	180
336	A.2.1 Non-TSF part of the TOE.....	181
337	A.2.2 Determining the TSFI	182
338	A.2.3 Example: A complex DBMS.....	185
339	A.2.4 Example Functional Specification.....	186
340	A.3 ADV_INT: Supplementary material on TSF internals.....	188
341	A.3.1 Structure of procedural software	188
342	A.3.2 Complexity of procedural software.....	190
343	A.4 ADV_TDS: Subsystems and Modules	191
344	A.4.1 Subsystems.....	191
345	A.4.2 Modules	192
346	A.4.3 Levelling Approach	194
347	A.4.4 Security relevance	196
348	A.5 Supplementary material on formal methods	197
349	Annex B (informative) Composition (ACO)	199
350	B.1 Necessity for composed TOE evaluations	199
351	B.2 Performing Security Target evaluation for a composed TOE.....	200
352	B.3 Interactions between composed IT entities	201
353	Annex C (informative) Cross reference of assurance component dependencies.....	207
354		

Foreword

To be review by ISO-Editor

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.

International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.

The main task of the joint technical committee is to prepare International Standards. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75 % of the national bodies casting a vote.

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

ISO/IEC 15408-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 27, *Information security, cybersecurity and privacy protection*.

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

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

Editors note: The following para will be updated when the new content of this part is stable.

This corrected version of ISO/IEC 15408-3:XXXX incorporates miscellaneous editorial corrections mainly related to EAL4 and EAL6 assurance components, ADV_FSP, ADV_TDS, ATE_DPT.2, ATE_IND, and ALC.

383 Legal Notice

384 The governmental organizations listed below contributed to the development of this version of the
 385 Common Criteria for Information Technology Security Evaluations. As the joint holders of the
 386 copyright in the Common Criteria for Information Technology Security Evaluations, version 3.1 Parts 1
 387 through 3 (called CC 3.1), they hereby grant non-exclusive license to ISO/IEC to use CC 3.1 in the
 388 continued development/maintenance of the ISO/IEC 15408 series international standard. However,
 389 these governmental organizations retain the right to use, copy, distribute, translate or modify CC 3.1 as
 390 they see fit.

391	Australia/New Zealand:	The Defence Signals Directorate and the Government Communications
392		Security Bureau respectively;
393	Canada:	Communications Security Establishment;
394	France:	Agence nationale de la sécurité des systèmes d'information (ANSSI);
395	Germany:	Bundesamt für Sicherheit in der Informationstechnik;
396	Japan:	Information Technology Promotion Agency;
397	Netherlands:	Netherlands National Communications Security Agency;
398	Spain:	Ministerio de Administraciones Públicas and Centro Criptológico
399	Nacional;	
400	United Kingdom:	Communications-Electronic Security Group;
401	United States:	The National Security Agency and the National Institute of Standards
402	and	Technology.

403 Introduction

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

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

410 The audience for this document includes consumers, developers, technical working groups, evaluators
 411 of secure IT products and others. ISO/IEC 15408-1:XXXX, Clause 5 provides additional information on
 412 the target audience of the ISO/IEC 15408 series, and on the use of the ISO/IEC 15408 series by the
 413 groups that comprise the target audience. These groups may use this document as follows:

- 414 a) Consumers, who use this document when selecting components to express assurance
 415 requirements to satisfy the security objectives expressed in a PP or ST, determining required
 416 levels of security assurance of the TOE.
- 417 b) Developers, who respond to actual or perceived consumer security requirements in constructing a
 418 TOE, reference this document when interpreting statements of assurance requirements and
 419 determining assurance approaches of TOEs.
- 420 c) Evaluators, who use the assurance requirements defined in this document as a mandatory
 421 statement of evaluation criteria when determining the assurance of TOEs and when evaluating
 422 PPs and STs.

Information security, cybersecurity and privacy protection — Evaluation criteria for IT security — Part 3: Security assurance components

1 Scope

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

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies..

ISO/IEC 15408-1, Information security -- Evaluation criteria for IT security — *Part 1: Introduction and general model*

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

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

3 Terms and definitions, symbols and abbreviated terms

For the purposes of this document, the terms, definitions, symbols and abbreviated terms given in ISO/IEC 15408-1 apply.

4 Overview

Clause 5 describes the paradigm used in the security assurance requirements of this document.

Clause 6 describes the presentation structure of the assurance classes, families, components, evaluation assurance levels along with their relationships, and the structure of the composed assurance packages. It also characterises the assurance classes and families found in Clauses 7 through 15.

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

Annex A provides further explanations and examples of the concepts behind the Development class.

Annex B provides an explanation of the concepts behind composed TOE evaluations and the Composition class.

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

5 Assurance paradigm

5.1 Introduction

The purpose of clause 5 is to document the philosophy that underpins the ISO/IEC 15408 series approach to assurance. An understanding of clause 5 will permit the reader to understand the rationale behind this document assurance requirements.

5.2 ISO/IEC 15408 philosophy

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

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

5.3 Assurance approach

5.3.1 Introduction

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

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

5.3.2 Significance of vulnerabilities

It is assumed that there are threat agents that will actively seek to exploit opportunities to violate security policies both for illicit gains and for well-intentioned, but nonetheless insecure actions. Threat agents may also accidentally trigger security vulnerabilities, causing harm to the organisation. Due to the need to process sensitive information and the lack of availability of sufficiently trusted products, there is significant risk due to failures of IT. It is, therefore, likely that IT security breaches could lead to significant loss.

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

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

- a) eliminated - that is, active steps should be taken to expose, and remove or neutralise, all exercisable vulnerabilities;

- b) minimised - that is, active steps should be taken to reduce, to an acceptable residual level, the potential impact of any exercise of a vulnerability;
- c) monitored - that is, active steps should be taken to ensure that any attempt to exercise a residual vulnerability will be detected so that steps can be taken to limit the damage.

5.3.3 Cause of vulnerabilities

Vulnerabilities can arise through failures in:

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

5.3.4 ISO/IEC 15408 series assurance

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

5.3.5 Assurance through evaluation

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

- a) analysis and checking of process(es) and procedure(s);
- b) checking that process(es) and procedure(s) are being applied;
- c) analysis of the correspondence between TOE design representations;
- d) analysis of the TOE design representation against the requirements;
- e) verification of proofs;

- f) analysis of guidance documents;
- g) analysis of functional tests developed and the results provided;
- h) independent functional testing;
- i) analysis for vulnerabilities (including flaw hypotheses);
- j) penetration testing;
- k) analysis of the delivery process;
- l) analysis of the maintenance process.

5.4 ISO/IEC 15408 series evaluation assurance scale

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

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

6 Security assurance components

6.1 Introduction

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

Figure 1 illustrates the security assurance requirements (SARs) defined in this document. Note that the most abstract collection of SARs is referred to as a class. Each class contains assurance families, which then contain assurance components, which in turn contain assurance elements. Classes and families are used to provide a taxonomy for classifying SARs, while components are used to specify SARs in a PP/ST.

6.2 Assurance class structure

Figure 1 illustrates the assurance class structure.

6.2.1 Class name

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

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

6.2.2 Class introduction

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

6.2.3 Assurance families

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

Figure 1 illustrates the assurance family structure.

Common criteria assurance requirements

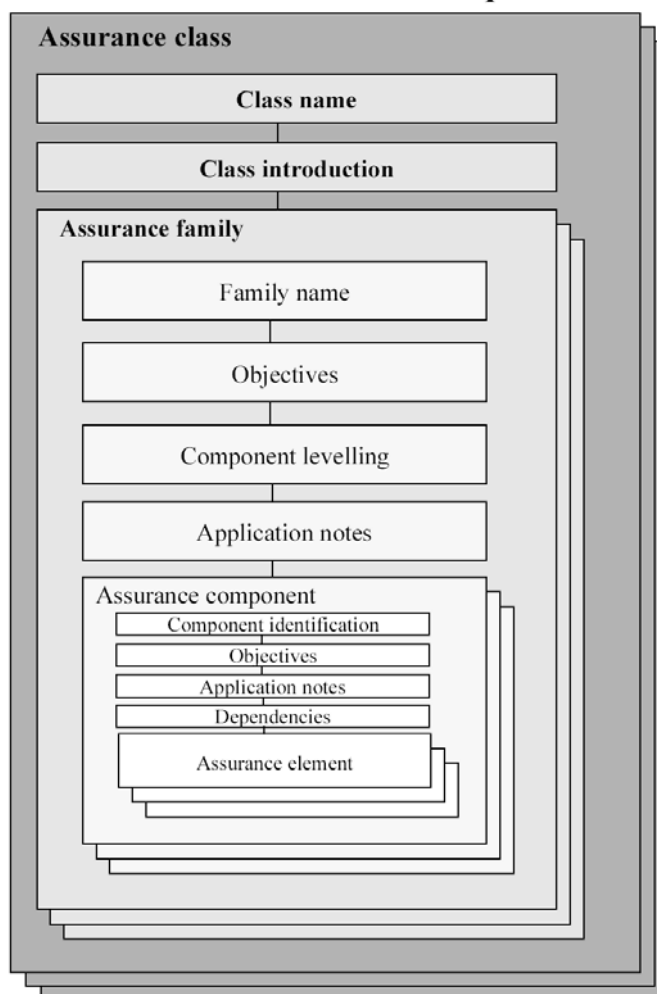


Figure 1 — Assurance class/family/component/element hierarchy

6.3 Assurance family structure

6.3.1 Family name

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

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

6.3.2 Objectives

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

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

6.3.3 Component levelling

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

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

6.3.4 Application notes

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

6.3.5 Assurance components

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

6.4 Assurance component structure

6.4.1 Introduction

Figure 2 illustrates the assurance component structure.

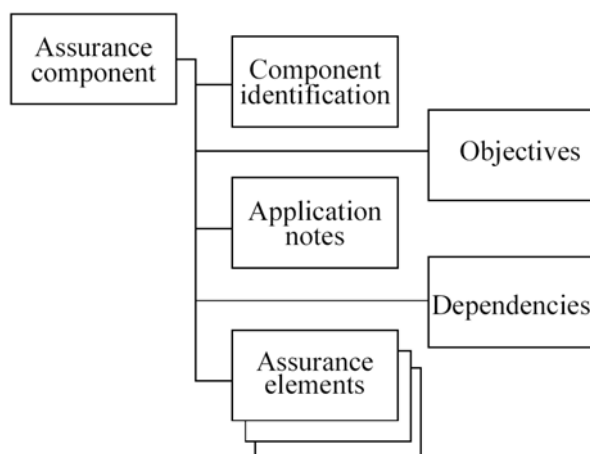


Figure 2 — Assurance component structure

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

6.4.2 Component identification

The component identification subclause provides descriptive information necessary to identify, categorise, register, and reference a component.

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

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

6.4.3 Objectives

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

6.4.4 Application notes

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

6.4.5 Dependencies

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

Each assurance component provides a complete list of dependencies to other assurance components. Some components may list “No dependencies”, to indicate that no dependencies have been identified. The components depended upon may have dependencies on other components.

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

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

6.4.6 Assurance elements

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

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

- a) Developer action elements: the activities that shall be performed by the developer. This set of actions is further qualified by evidential material referenced in the following set of elements. Requirements for developer actions are identified by appending the letter "D" to the element number.
- b) Content and presentation of evidence elements: the evidence required, what the evidence shall demonstrate, and what information the evidence shall convey. Requirements for content and presentation of evidence are identified by appending the letter "C" to the element number.
- c) Evaluator action elements: the activities that shall be performed by the evaluator. This set of actions explicitly includes confirmation that the requirements prescribed in the content and presentation of evidence elements have been met. It also includes explicit actions and analysis that shall be performed in addition to that already performed by the developer. Implicit evaluator actions are also to be performed as a result of developer action elements which are not covered by content and presentation of evidence requirements. Requirements for evaluator actions are identified by appending the letter "E" to the element number.

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

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

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

6.5 Assurance elements

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

6.6 Component taxonomy

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

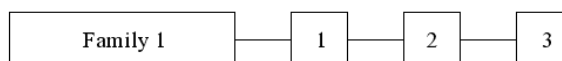


Figure 3 — Sample class decomposition diagram

In Figure 3, above, the class as shown contains a single family. The family contains three components that are linearly hierarchical (i.e. component 2 requires more than component 1, in terms of specific actions, specific evidence, or rigour of the actions or evidence). The assurance families in this document are all linearly hierarchical, although linearity is not a mandatory criterion for assurance families that may be added in the future.

7 Class APE: Protection Profile evaluation

7.1 Introduction

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

Clause 7 should be used in conjunction with Annexes A, B and C in ISO/IEC 15408-1:2009, as these annexes clarify the concepts here and provide many examples.

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

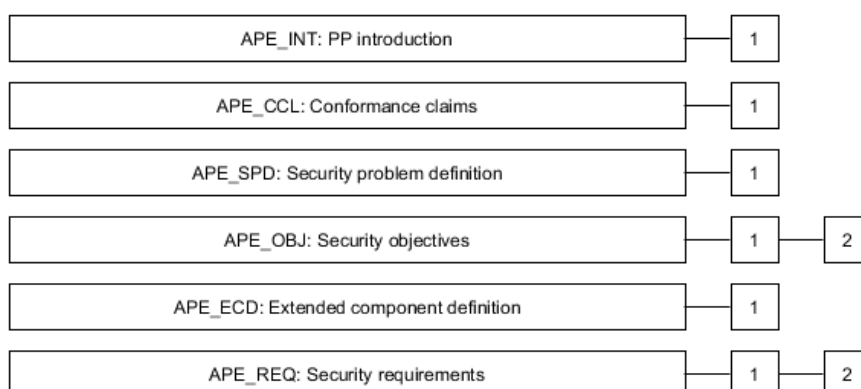


Figure 4 — APE: Protection Profile evaluation class decomposition

697 **7.2 PP introduction (APE_INT)**

698 **7.2.1 Objectives**

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

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

702 **7.2.2 APE_INT.1 PP introduction**

703 Dependencies: No dependencies.

704 **Developer action elements**

705 **APE_INT.1.1D**

706 **The developer shall provide a PP introduction.**

707 **Content and presentation elements**

708 **APE_INT.1.1C**

709 **The PP introduction shall contain a PP reference and a TOE overview.**

710 **APE_INT.1.2C**

711 **The PP reference shall uniquely identify the PP.**

712 **APE_INT.1.3C**

713 **The TOE overview shall summarise the usage and major security features of the TOE.**

714 **APE_INT.1.4C**

715 **The TOE overview shall identify the TOE type.**

716 **APE_INT.1.5C**

717 **The TOE overview shall identify any non-TOE hardware/software/firmware available to the**
718 **TOE.**

719 **Evaluator action elements**

720 **APE_INT.1.1E**

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

723 **7.3 Conformance claims (APE_CCL)**

724 **7.3.1 Objectives**

725 The objective of this family is to determine the validity of the conformance claim. In addition, this
726 family specifies how STs and other PPs are to claim conformance with the PP.

727 **7.3.2 APE_CCL.1 Conformance claims**

728 Dependencies: APE_INT.1 PP introduction

729 APE_ECD.1 Extended components definition

730 APE_REQ.1 Direct rationale PP-Module security requirements

731 **Developer action elements**

732 **APE_CCL.1.1D**

733 **The developer shall provide a conformance claim.**

734 **APE_CCL.1.2D**

735 **The developer shall provide a conformance claim rationale.**

736 **APE_CCL.1.3D**

737 **The developer shall provide a conformance statement.**

738 **Content and presentation elements**

739 **APE_CCL.1.1C**

740 **The conformance claim shall identify the ISO/IEC 15408 edition to which the PP claims**
741 **conformance.**

742 **APE_CCL.1.2C**

743 **The conformance claim shall describe the conformance of the PP to ISO/IEC 15408-2 as either**
744 **ISO/IEC 15408-2 conformant or ISO/IEC 15408-2 extended.**

745 **APE_CCL.1.3C**

746 **The conformance claim shall describe the conformance of the PP as either “ISO/IEC 15408-3**
747 **conformant” or ISO/IEC 15408-3 extended.”**

748 **APE_CCL.1.4C**

749 **The conformance claim shall be consistent with the extended components definition.**

750 **APE_CCL.1.5C**

751 **The conformance claim shall identify all PPs, PP-Configurations and functional packages to**
752 **which the PP claims conformance.**

753 **APE_CCL.1.6C**

754 **The conformance claim shall describe any conformance of the PP to a functional package as one**
755 **of package-conformant, package-augmented, or package-tailored.**

756 APE_CCL.1.7C

757 The conformance claim shall describe any conformance of the PP to an assurance package as
758 either package-conformant or package-augmented.

759 APE_CCL.1.8C

760 The conformance claim rationale shall demonstrate that the TOE type is consistent with the
761 TOE type in the PPs or PP-Configurations for which conformance is being claimed.

762 APE_CCL.1.9C

763 The conformance claim rationale shall demonstrate that the statement of the security problem
764 definition is consistent with the statement of the security problem definition in the PP-
765 Configuration, PPs and any functional packages for which conformance is being claimed..

766 APE_CCL.1.10C

767 The conformance claim rationale shall demonstrate that the statement of security objectives is
768 consistent with the statement of security objectives in the PP-Configuration, PPs and any
769 functional packages for which conformance is being claimed.

770 APE_CCL.1.11C

771 The conformance claim rationale shall demonstrate that the statement of security
772 requirements is consistent with the statement of security requirements in the PP-
773 Configuration, PPs and any functional packages for which conformance is being claimed.

774 APE_CCL.1.12C

775 The conformance statement shall describe the conformance required of any PPs/STs to the PP
776 as exact-PP, strict-PP, or demonstrable-PP conformance.

777 APE_CCL.1.13C

778 For an exact conformance PP, the conformance statement shall contain an allowed-with
779 statement that identifies the set of PPs (if any) to which, in combination with the PP under
780 evaluation, exact conformance is allowed to be claimed.

781 APE_CCL.1.14C

782 For an exact conformance PP, the conformance statement shall contain an allowed-with
783 statement that identifies the set of PP-Modules (if any) that are allowed to be used with the PP
784 under evaluation in a PP-Configuration.

785 APE_CCL.1.15C

786 The conformance statement shall identify the set of derived Evaluation Methods and Evaluation
787 Activities (if any) that shall be used with the PP under evaluation. This list shall contain:

- 788 • any Evaluation Methods and Evaluation Activities that are specified for the PP under
789 evaluation
- 790 • any Evaluation Methods and Evaluation Activities specified in conformance statements
791 of PPs to which conformance is being claimed by the PP under evaluation

- 792 • any Evaluation Methods and Evaluation Activities specified in the Security
793 Requirements sections of packages to which conformance is being claimed by the PP
794 under evaluation.

795 Evaluator action elements

796 APE_CCL.1.1E

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

799 7.4 Security problem definition (APE_SPD)

800 7.4.1 Objectives

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

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

805 7.4.2 APE_SPD.1 Security problem definition

806 Dependencies: No dependencies.

807 Developer action elements

808 APE_SPD.1.1D

809 The developer shall provide a security problem definition.

810 Content and presentation elements

811 APE_SPD.1.1C

812 The security problem definition shall describe the threats.

813 APE_SPD.1.2C

814 All threats shall be described in terms of a threat agent, an asset, and an adverse action.

815 APE_SPD.1.3C

816 The security problem definition shall describe the OSPs.

817 APE_SPD.1.4C

818 The security problem definition shall describe the assumptions about the operational
819 environment of the TOE.

820 Evaluator action elements

821 APE_SPD.1.1E

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

824 **7.5 Security objectives (APE_OBJ)**

825 **7.5.1 Objectives**

826 The security objectives are a concise statement of the intended response to the security problem
827 defined through the Security problem definition (APE_SPD) family.

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

831 **7.5.2 Component levelling**

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

834 **7.5.3 APE_OBJ.1 Security objectives for the operational environment**

835 Dependencies: No dependencies.

836 **Developer action elements**

837 **APE_OBJ.1.1D**

838 **The developer shall provide a statement of security objectives.**

839 **APE_OBJ.1.2D**

840 **The developer shall provide a security objectives rationale.**

841 **Content and presentation elements**

842 **APE_OBJ.1.1C**

843 **The statement of security objectives shall describe the security objectives for the operational**
844 **environment.**

845 **APE_OBJ.1.2C**

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

849 **APE_OBJ.1.3C**

850 **The security objectives rationale shall demonstrate that the security objectives for the**
851 **operational environment uphold all assumptions.**

852 **Evaluator action elements**

853 **APE_OBJ.1.1E**

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

856 **7.5.4 APE_OBJ.2 Security objectives**

857 Dependencies: APE_SPD.1 Security problem definition

858 **Developer action elements**

859 **APE_OBJ.2.1D**

860 The developer shall provide a statement of security objectives.

861 **APE_OBJ.2.2D**

862 The developer shall provide a security objectives rationale.

863 **Content and presentation elements**

864 **APE_OBJ.2.1C**

865 The statement of security objectives shall describe the security objectives for the **TOE and the**
866 **security objectives for the** operational environment.

867 **APE_OBJ.2.2C**

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

870 **APE_OBJ.2.3C**

871 The security objectives rationale shall trace each security objective for the operational environment
872 back to threats countered by that security objective, OSPs enforced by that security objective, and
873 assumptions upheld by that security objective.

874 **APE_OBJ.2.4C**

875 **The security objectives rationale shall demonstrate that the security objectives counter all**
876 **threats.**

877 **APE_OBJ.2.5C**

878 **The security objectives rationale shall demonstrate that the security objectives enforce all**
879 **OSPs.**

880 **APE_OBJ.2.6C**

881 The security objectives rationale shall demonstrate that the security objectives for the operational
882 environment uphold all assumptions.

883 **Evaluator action elements**

884 **APE_OBJ.2.1E**

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

887 **7.6 Extended components definition (APE_ECD)**

888 **7.6.1 Objectives**

889 Extended security requirements are requirements that are not based on components from ISO/IEC
890 15408-2 or this document, but are based on extended components: components defined by the PP
891 author.

892 Evaluation of the definition of extended components is necessary to determine that they are clear and
893 unambiguous, and that they are necessary, i.e. they may not be clearly expressed using existing
894 ISO/IEC 15408-2 or this document components.

895 **7.6.2 APE_ECD.1 Extended components definition**

896 Dependencies: No dependencies.

897 **Developer action elements**

898 **APE_ECD.1.1D**

899 **The developer shall provide a statement of security requirements.**

900 **APE_ECD.1.2D**

901 **The developer shall provide an extended components definition.**

902 **Content and presentation elements**

903 **APE_ECD.1.1C**

904 **The statement of security requirements shall identify all extended security requirements.**

905 **APE_ECD.1.2C**

906 **The extended components definition shall define an extended component for each extended**
907 **security requirement.**

908 **APE_ECD.1.3C**

909 **The extended components definition shall describe how each extended component is related to**
910 **the existing ISO/IEC 15408 series components, families, and classes.**

911 **APE_ECD.1.4C**

912 **The extended components definition shall use the existing ISO/IEC 15408 series components,**
913 **families, classes, and methodology as a model for presentation.**

914 **APE_ECD.1.5C**

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

917 **Evaluator action elements**

918 **APE_ECD.1.1E**

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

921 **APE_ECD.1.2E**

922 **The evaluator shall confirm that no extended component may be clearly expressed using**
 923 **existing components.**

924 **7.7 Security requirements (APE_REQ)**

925 **7.7.1 Objectives**

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

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

931 **7.7.2 Component levelling**

932 The components in this family are levelled on whether they are stated as is, or whether the SFRs are
 933 derived from security objectives for the TOE.

934 **7.7.3 APE_REQ.1 Direct rationale PP-Module security requirements**

935 Dependencies: APE_ECD.1 Extended components definition

936 **Developer action elements**

937 **APE_REQ.1.1D**

938 **The developer shall provide a statement of security requirements.**

939 **APE_REQ.1.2D**

940 **The developer shall provide a security requirements rationale.**

941 **Content and presentation elements**

942 **APE_REQ.1.1C**

943 **The statement of security requirements shall describe the SFRs and the SARs.**

944 **APE_REQ.1.2C**

945 **All subjects, objects, operations, security attributes, external entities and other terms that are**
 946 **used in the SFRs and the SARs shall be defined.**

947 **APE_REQ.1.3C**

948 **The statement of security requirements shall include a natural language description, part of**
 949 **which describes how the SFRs combine together to provide security functionality in terms of**
 950 **the architecture that is observable to Administrators and other users, or in terms of internal**
 951 **features or properties.**

952 **APE_REQ.1.4C**

953 **The statement of security requirements shall identify all operations on the security**
 954 **requirements.**

955 **APE_REQ.1.5C**

956 **All operations shall be performed correctly.**

957 **APE_REQ.1.6C**

958 **Each dependency of the security requirements shall either be satisfied, or the security**
 959 **requirements rationale shall justify the dependency not being satisfied.**

960 **APE_REQ.1.7C**

961 **The security requirements rationale shall trace each SFR back to the threats countered by that**
 962 **SFR and the OSPs enforced by that SFR.**

963 **APE_REQ.1.8C**

964 **The security requirements rationale shall demonstrate that the SFRs (in conjunction with the**
 965 **security objectives for the environment) counter all threats for the TOE.**

966 **APE_REQ.1.9C**

967 **The security requirements rationale shall demonstrate that the SFRs (in conjunction with the**
 968 **security objectives for the environment) enforce all OSPs for the TOE.**

969 **APE_REQ.1.10C**

970 **The statement of security requirements shall be internally consistent.**

971 **Evaluator action elements**972 **APE_REQ.1.1E**

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

975 **7.7.4 APE_REQ.2 Derived security requirements**

976 Dependencies: APE_OBJ.2 Security objectives

977 APE_ECD.1 Extended components definition

978 **Developer action elements**

979 **APE_REQ.2.1D**

980 The developer shall provide a statement of security requirements.

981 **APE_REQ.2.2D**

982 The developer shall provide a security requirements rationale.

983 **Content and presentation elements**

984 **APE_REQ.2.1C**

985 The statement of security requirements shall describe the SFRs and the SARs.

986 **APE_REQ.2.2C**

987 All subjects, objects, operations, security attributes, external entities and other terms that are used in
988 the SFRs and the SARs shall be defined.

989 **APE_REQ.2.3C**

990 The statement of security requirements shall identify all operations on the security requirements.

991 **APE_REQ.2.4C**

992 All operations shall be performed correctly.

993 **APE_REQ.2.5C**

994 Each dependency of the security requirements shall either be satisfied, or the security requirements
995 rationale shall justify the dependency not being satisfied.

996 **APE_REQ.2.6C**

997 **The security requirements rationale shall trace each SFR back to the security objectives for the**
998 **TOE enforced by that SFR.**

999 **APE_REQ.2.7C**

1000 **The security requirements rationale shall demonstrate that the SFRs meet all security**
1001 **objectives for the TOE.**

1002 **APE_REQ.2.8C**

1003 The security requirements rationale shall explain why the SARs were chosen.

1004 **APE_REQ.2.9C**

1005 The statement of security requirements shall be internally consistent.

1006 **Evaluator action elements**1007 **APE_REQ.2.1E**

1008 The evaluator shall confirm that the information provided meets all requirements for content and
 1009 presentation of evidence.

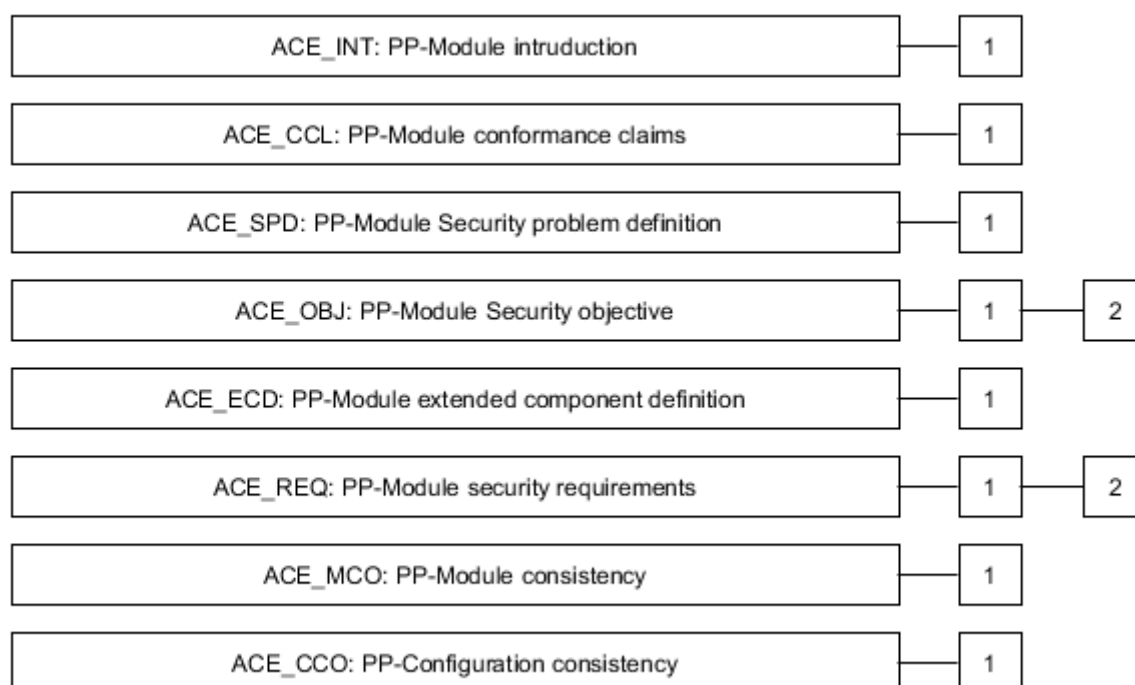
1010 **8 Class ACE: Protection Profile Configuration evaluation**1011 **8.1 Introduction**

1012 Evaluating a PP-Configuration is required to demonstrate that the PP-Configuration is sound and
 1013 consistent. These properties are necessary for the PP-Configuration to be suitable for use as the basis
 1014 for writing an ST.

1015 The class ACE is defined for the evaluation of a PP-Configuration composed of at least one PP and one
 1016 other component (PPs and/or PP-Modules). The evaluation of PPs is addressed in Class APE. The class
 1017 ACE defines the requirements for:

- 1018 • Evaluating the PP-Modules in the framework of their PP-Modules Base(s) (components
 1019 ACE_INT.1, ACE_CCL.1, ACE_SPD.1, ACE_OBJ.1 or ACE_OBJ.2, ACE_REQ.1 or ACE_REQ.2, and
 1020 ACE_MCO.1.
- 1021 • Evaluating the consistency of the combination of all the PPs and PP-Modules that belong to the
 1022 PP-Configuration (see ACE_CCO.1).

1023 Clause 8 should be used in conjunction with Annex C of ISO/IEC 15408-1.



1024

1025 **Figure 5: ACE: Protection Profile Configuration evaluation class decomposition**

1026 **8.2 PP-Module introduction (ACE_INT)**

1027 **8.2.1 Objectives**

1028 The objective of this family is to describe the TOE in a narrative way.

1029 The evaluation of the PP-Module introduction is required to demonstrate that the PP-Module is
1030 correctly identified, and that the PP-Module reference and TOE overview are consistent with each
1031 other.

1032 **8.2.2 ACE_INT.1 PP-Module introduction**

1033 Dependencies: No dependencies.

1034 **Developer action elements**

1035 **ACE_INT.1.1D**

1036 **The developer shall provide a PP-Module introduction.**

1037 **Content and presentation elements**

1038 **ACE_INT.1.1C**

1039 **The PP-Module introduction shall contain a PP-Module reference, the identification of the PP-**
1040 **Module Base(s) and a TOE overview.**

1041 **ACE_INT.1.2C**

1042 **The PP-Module reference shall uniquely identify the PP-Module.**

1043 **ACE_INT.1.3C**

1044 **The identification of the PP-Module Base shall consist of a list of at least one PP and possibly**
1045 **other PPs and PP-Modules on which the PP-Module depends.**

1046 **ACE_INT.1.4C**

1047 **The identification of the PP-Module Base(s) shall describe the dependency structure of the PP-**
1048 **Module Base(s).**

1049 **ACE_INT.1.5C**

1050 **The PP-Module introduction shall contain as many TOE overviews as alternative PP-Module**
1051 **Bases.**

1052 **ACE_INT.1.6C**

1053 **The TOE overview shall summarise the usage and major security features of the TOE.**

1054 **ACE_INT.1.7C**

1055 **The TOE overview shall identify the TOE type.**

- 1056 **ACE_INT.1.8C**
- 1057 **The TOE overview shall identify any non-TOE hardware/software/firmware available to the**
1058 **TOE.**
- 1059 **ACE_INT.1.9C**
- 1060 **The TOE overview shall describe the differences of the TOE with regard to the TOEs defined in**
1061 **the PP-Module Base(s).**
- 1062 **Evaluator action elements**
- 1063 **ACE_INT.1.1E**
- 1064 **The evaluator shall confirm that the information provided meets all requirements for content**
1065 **and presentation of evidence.**
- 1066 **8.3 PP-Module conformance claims (ACE_CCL)**
- 1067 **8.3.1 Objectives**
- 1068 The objective of this family is to determine the validity of the conformance claim and conformance
1069 statement. A PP-Module cannot claim conformance to any PP, PP-Configuration, or another PP-Module.
- 1070 **8.3.2 ACE_CCL.1 PP-Module conformance claims**
- 1071 Dependencies: ACE_INT.1 PP-Module introduction
- 1072 ACE_ECD.1 PP-Module extended components definition
- 1073 ACE_REQ.1 PP-Module stated security requirements or ACE_REQ.2 PP-Module
1074 derived security requirements
- 1075 **Developer action elements**
- 1076 **ACE_CCL.1.1D**
- 1077 **The developer shall provide a conformance claim.**
- 1078 **ACE_CCL.1.2D**
- 1079 **The developer shall provide a conformance statement.**
- 1080 **Content and presentation elements**
- 1081 **ACE_CCL.1.1C**
- 1082 **The conformance claim shall identify the ISO/IEC 15408 edition to which the PP-Module claims**
1083 **conformance.**
- 1084 **ACE_CCL.1.2C**
- 1085 **The conformance claim shall describe the conformance of the PP-Module to ISO/IEC 15408-2 as**
1086 **either ISO/IEC 15408-2 conformant or ISO/IEC 15408-2 extended.**

- 1087 **ACE_CCL.1.3C**
- 1088 **The conformance statement shall describe the conformance type required of any ST to the PP-**
 1089 **Module (as part of a PP-Configuration) as one of exact, strict, or demonstrable.**
- 1090 **ACE_CCL.1.4C**
- 1091 **The conformance claim shall describe the conformance of the PP-Module to this document as**
 1092 **either “ISO/IEC 15408-3 conformant” or ISO/IEC 15408-3 extended.”**
- 1093 **ACE_CCL.1.5C**
- 1094 **The conformance claim shall be consistent with the extended components definition.**
- 1095 **ACE_CCL.1.6C**
- 1096 **The conformance claim shall identify all functional packages to which the PP-Module claims**
 1097 **conformance.**
- 1098 **ACE_CCL.1.7C**
- 1099 **The conformance claim shall describe any conformance of the PP-Module to a functional**
 1100 **package as either package-conformant, package-augmented or package-tailored.**
- 1101 **ACE_CCL.1.8C**
- 1102 **The conformance claim shall identify all assurance packages to which the PP-Module claims**
 1103 **conformance.**
- 1104 **ACE_CCL.1.9C**
- 1105 **The conformance claim shall describe any conformance of the PP-Module to an assurance**
 1106 **package as either package-conformant or package-augmented.**
- 1107 **ACE_CCL.1.10C**
- 1108 **For exact conformance, the PP-Module’s conformance statement shall contain an allowed-with**
 1109 **statement that identifies the set of PPs and PP-Modules (exclusive of those PPs and PP-Modules**
 1110 **that are included in the PP-Module Base) to which, in combination with the PP-Module under**
 1111 **evaluation, exact conformance is allowed to be claimed.**
- 1112 **ACE_CCL.1.11C**
- 1113 **The conformance statement may identify the set of ISO/IEC 18045-derived Evaluation Methods**
 1114 **and Evaluation Activities that shall be used with the PP-Module under evaluation. This list shall**
 1115 **contain any Evaluation Methods and Evaluation Activities that are specified in the PP-Module**
 1116 **but also any Evaluation Methods and Evaluation Activities specified in the PP-Module Base(s)**
 1117 **and/or in the packages (if any) for which conformance is being claimed by the PP-Module**
 1118 **under evaluation.**

- 1119 **Evaluator action elements**
- 1120 **ACE_CCL.1.1E**
- 1121 **The evaluator shall confirm that the information provided meets all requirements for content**
 1122 **and presentation of evidence.**
- 1123 **8.4 PP-Module security problem definition (ACE_SPD)**
- 1124 **8.4.1 Objectives**
- 1125 This part of the PP-Module defines the security problem to be addressed by the TOE and the
 1126 operational environment of the TOE.
- 1127 Evaluation of the security problem definition is required to demonstrate that the security problem
 1128 intended to be addressed by the TOE and its operational environment, is clearly defined.
- 1129 **8.4.2 ACE_SPD.1 PP-Module Security problem definition**
- 1130 Dependencies: No dependencies.
- 1131 **Developer action elements**
- 1132 **ACE_SPD.1.1D**
- 1133 **The developer shall provide a security problem definition.**
- 1134 **Content and presentation elements**
- 1135 **ACE_SPD.1.1C**
- 1136 **The security problem definition shall describe the threats.**
- 1137 **ACE_SPD.1.2C**
- 1138 **All threats shall be described in terms of a threat agent, an asset, and an adverse action.**
- 1139 **ACE_SPD.1.3C**
- 1140 **The security problem definition shall describe the OSPs.**
- 1141 **ACE_SPD.1.4C**
- 1142 **The security problem definition shall describe the assumptions about the operational**
 1143 **environment of the TOE.**
- 1144 **Evaluator action elements**
- 1145 **ACE_SPD.1.1E**
- 1146 **The evaluator shall confirm that the information provided meets all requirements for content**
 1147 **and presentation of evidence.**

1148 **8.5 PP-Module security objectives (ACE_OBJ)**

1149 **8.5.1 Objectives**

1150 The security objectives are a concise statement of the intended response to the security problem
1151 defined through the Security problem definition (APE_SPD) family.

1152 Evaluation of the security objectives is required to demonstrate that the security objectives adequately
1153 and completely address the security problem definition and that the division of this problem between
1154 the TOE and its operational environment is clearly defined.

1155 **8.5.2 Component levelling**

1156 The components in this family are levelled on whether they prescribe only security objectives for the
1157 operational environment (see ACE_OBJ.1), or also security objectives for the TOE (see ACE_OBJ.2).

1158 **8.5.3 ACE_OBJ.1 Direct Rationale PP-Module security objectives**

1159 Dependencies: No dependencies.

1160 **Developer action elements**

1161 **ACE_OBJ.1.1D**

1162 **The developer shall provide a statement of security objectives for the PP-Module.**

1163 **ACE_OBJ.1.2D**

1164 **The developer shall provide a security objectives rationale for the PP-Module.**

1165 **Content and presentation elements**

1166 **ACE_OBJ.1.1C**

1167 **The statement of security objectives shall describe the security objectives for the operational**
1168 **environment.**

1169 **ACE_OBJ.1.2C**

1170 **The security objectives rationale shall trace each security objective for the operational**
1171 **environment back to threats countered by that security objective, OSPs enforced by that**
1172 **security objective, and assumptions upheld by that security objective.**

1173 **ACE_OBJ.1.3.C**

1174 **The security objectives rationale shall demonstrate that the security objectives for the**
1175 **operational environment uphold all assumptions.**

1176 **Evaluator action elements**

1177 **ACE_OBJ.1.1E**

1178 **The evaluator shall confirm that the information provided meets all requirements for content**
1179 **and presentation of evidence.**

1180 **8.5.4 ACE_OBJ.2 PP-Module security objectives**

1181 Dependencies: ACE_SPD.1 PP-Module security problem definition.

1182 **Developer action elements**

1183 **ACE_OBJ.2.1D**

1184 **The developer shall provide a statement of security objectives for the PP-Module.**

1185 **ACE_OBJ.2.2D**

1186 **The developer shall provide a security objectives rationale for the PP-Module.**

1187 **Content and presentation elements**

1188 **ACE_OBJ.2.1C**

1189 **The statement of security objectives shall describe the security objectives for the TOE and the**
 1190 **security objectives for the operational environment.**

1191 **ACE_OBJ.2.2C**

1192 **The security objectives rationale shall trace each security objective for the TOE back to threats**
 1193 **countered by that security objective and OSPs enforced by that security objective.**

1194 **ACE_OBJ.2.3C**

1195 **The security objectives rationale shall trace each security objective for the operational**
 1196 **environment back to threats countered by that security objective, OSPs enforced by that**
 1197 **security objective, and assumptions upheld by that security objective.**

1198 **ACE_OBJ.2.4C**

1199 **The security objectives rationale shall demonstrate that the security objectives counter all**
 1200 **threats.**

1201 **ACE_OBJ.2.5C**

1202 **The security objectives rationale shall demonstrate that the security objectives enforce all**
 1203 **OSPs.**

1204 **ACE_OBJ.2.6C**

1205 **The security objectives rationale shall demonstrate that the security objectives for the**
 1206 **operational environment uphold all assumptions.**

1207 **Evaluator action elements**

1208 **ACE_OBJ.2.1E**

1209 **The evaluator shall confirm that the information provided meets all requirements for content**
 1210 **and presentation of evidence.**

1211 **8.6 PP-Module extended components definition (ACE_ECD)**

1212 **8.6.1 Objectives**

1213 Extended security functional requirements are requirements that are not based on components from
1214 ISO/IEC 15408-2 or this document, but are based on extended components: components defined by
1215 the PP-Module author.

1216 Evaluation of the definition of extended functional components is necessary to determine that they are
1217 clear and unambiguous, and that they are necessary, i.e. they may not be clearly expressed using
1218 existing ISO/IEC 15408-2 or this document components.

1219 **8.6.2 ACE_ECD.1 PP-Module extended components definition**

1220 Dependencies: No dependencies.

1221 **Developer action elements**

1222 **ACE_ECD.1.1D**

1223 **The developer shall provide a statement of security requirements for the PP-Module.**

1224 **ACE_ECD.1.2D**

1225 **The developer shall provide an extended components definition for the PP-Module.**

1226 **Content and presentation elements**

1227 **ACE_ECD.1.1C**

1228 **The statement of security requirements shall identify all the extended security requirements.**

1229 **ACE_ECD.1.2C**

1230 **The extended components definition shall define an extended component for each extended**
1231 **security requirement.**

1232 **ACE_ECD.1.3C**

1233 **The extended components definition shall describe how each extended component is related to**
1234 **the existing ISO/IEC 15408 series components, families, and classes.**

1235 **ACE_ECD.1.4C**

1236 **The extended components definition shall use the existing ISO/IEC 15408 series components,**
1237 **families, classes, and methodology as a model for presentation.**

- 1238 **ACE_ECD.1.5C**
- 1239 **The extended components shall consist of measurable and objective elements such that**
 1240 **conformance or nonconformance to these elements may be demonstrated.**
- 1241 **Evaluator action elements**
- 1242 **ACE_ECD.1.1E**
- 1243 **The evaluator shall confirm that the information provided meets all requirements for content**
 1244 **and presentation of evidence.**
- 1245 **ACE_ECD.1.2E**
- 1246 **The evaluator shall confirm that no extended component may be clearly expressed using**
 1247 **existing components.**
- 1248 **8.7 PP-Module security requirements (ACE_REQ)**
- 1249 **8.7.1 Objectives**
- 1250 The SFRs form a clear, unambiguous and well-defined description of the expected security behaviour
 1251 of the TOE. The SARs form a clear, unambiguous and well-defined description of the expected activities
 1252 that will be undertaken to gain assurance in the TOE.
- 1253 Evaluation of the security requirements is required to ensure that they are clear, unambiguous and
 1254 well-defined.
- 1255 **8.7.2 Component levelling**
- 1256 The components in this family are levelled on whether they are stated as is (see ACE_REQ.1), or
 1257 whether the SFRs are derived from the security objectives for the TOE (see ACE_REQ.2.).
- 1258 **8.7.3 ACE_REQ.1 PP-Module stated security requirements**
- 1259 Dependencies: APE_ECD.1 Extended components definition
- 1260 ACE_SPD.1 PP-Module security problem definition
- 1261 **Developer action elements**
- 1262 **ACE_REQ.1.1D**
- 1263 **The developer shall provide a statement of security requirements for the PP-Module.**
- 1264 **ACE_REQ.1.2D**
- 1265 **The developer shall provide a security requirements rationale for the PP-Module.**
- 1266 **Content and presentation elements**
- 1267 **ACE_REQ.1.1C**
- 1268 **The statement of security requirements shall describe the SFRs and SARs (the SARs that apply**
 1269 **to the PP-Module may be explicitly stated, or inherited from the PP-Module Base(s)).**

1270 **ACE_REQ.1.2C**

1271 **All subjects, objects, operations, security attributes, external entities and other terms that are**
 1272 **used in the SFRs and the SARs shall be defined.**

1273 **ACE_REQ.1.3C**

1274 **The statement of security requirements shall include a natural language description, part of**
 1275 **which describes how the SFRs combine together to provide security functionality in terms of**
 1276 **the architecture that is observable to Administrators and other users, or in terms of internal**
 1277 **features or properties.**

1278 **ACE_REQ.1.4C**

1279 **The statement of security requirements shall identify all operations on the security**
 1280 **requirements.**

1281 **ACE_REQ.1.5C**

1282 **All operations shall be performed correctly.**

1283 **ACE_REQ.1.6C**

1284 **Each dependency of the security requirements shall either be satisfied, or the security**
 1285 **requirements rationale shall justify the dependency not being satisfied.**

1286 **ACE_REQ.1.7C**

1287 **The security requirements rationale shall trace each SFR back to the threats countered by that**
 1288 **SFR and the OSPs enforced by that SFR.**

1289 **ACE_REQ.1.8C**

1290 **The security requirements rationale shall demonstrate that the SFRs (in conjunction with the**
 1291 **security objectives for the environment) counter all the threats for the TOE.**

1292 **ACE_REQ.1.9C**

1293 **The security requirements rationale shall demonstrate that the SFRs (in conjunction with the**
 1294 **security objectives for the environment) enforce all the OSPs for the TOE.**

1295 **ACE_REQ.1.10C**

1296 **The security requirements rationale shall explain why the SARs were chosen.**

1297 **ACE_REQ.1.11C**

1298 **The statement of security requirements shall be internally consistent.**

1299 **Evaluator action elements**1300 **ACE_REQ.1.1E**

1301 **The evaluator shall confirm that the information provided meets all requirements for content**
 1302 **and presentation of evidence.**

1303 **8.7.4 ACE_REQ.2 PP-Module derived security requirements**

1304 Dependencies: ACE_ECD.1 PP-Module extended components definition

1305 ACE_OBJ.2 PP-Module security objectives

1306 **Developer action elements**

1307 **ACE_REQ.2.1D**

1308 The developer shall provide a statement of security requirements for the PP-Module.

1309 **ACE_REQ.2.2D**

1310 The developer shall provide a security requirement rationale for the PP-Module.

1311 **Content and presentation elements**

1312 **ACE_REQ.2.1C**

1313 **The statement of security requirements shall describe the SFRs and SARs (the SARs that apply**
 1314 **to the PP-Module may be explicitly stated, or inherited from the PP-Module Base(s)).**

1315 **ACE_REQ.2.2C**

1316 **All subjects, objects, operations, security attributes, external entities and other terms that are**
 1317 **used in the SFRs and the SARs shall be defined.**

1318 **ACE_REQ.2.3C**

1319 **The statement of security requirements shall identify all operations on the security**
 1320 **requirements.**

1321 **ACE_REQ.2.4C**

1322 **All operations shall be performed correctly.**

1323 **ACE_REQ.2.5C**

1324 **Each dependency of the security requirements shall either be satisfied, or the security**
 1325 **requirements rationale shall justify the dependency not being satisfied.**

1326 **ACE_REQ.2.6C**

1327 **The security requirements rationale shall trace each SFR back to the security objectives for the**
 1328 **TOE enforced by that SFR.**

1329 **ACE_REQ.2.7C**

1330 **The security requirements rationale shall demonstrate that the SFRs meet all security**
 1331 **objectives for the TOE.**

1332 **ACE_REQ.2.8C**

1333 **The security requirements rationale shall explain why the SARs were chosen.**

- 1334 **ACE_REQ.2.9C**
- 1335 **The statement of security requirements shall be internally consistent.**
- 1336 **Evaluator action elements**
- 1337 **ACE_REQ.2.1E**
- 1338 **The evaluator shall confirm that the information provided meets all requirements for content**
 1339 **and presentation of evidence.**
- 1340 **8.8 PP-Module consistency (ACE_MCO)**
- 1341 **8.8.1 Objectives**
- 1342 The objective of this family is to determine the consistency of the PP-Module and to state the
 1343 correspondence between the PP-Module and its PP-Module Base(s).
- 1344 **8.8.2 ACE_MCO.1 PP-Module consistency**
- 1345 Dependencies: ACE_INT.1 PP-Module introduction
- 1346 ACE_SPD.1 PP-Module Security problem definition
- 1347 ACE_OBJ.1 Direct Rationale PP-Module Security objectives for the environment or
 1348 ACE_OBJ.2 PP-Module Security objectives
- 1349 ACE_REQ.1 Direct Rationale PP-Module security requirements or ACE_REQ.2 PP-
 1350 Module derived security requirements
- 1351 **Developer action elements**
- 1352 **ACE_MCO.1.1D**
- 1353 **The developer shall provide a consistency rationale of the PP-Module for each of the alternative**
 1354 **PP-Module Bases identified in the PP-Module introduction.**
- 1355 **ACE_MCO.1.2D**
- 1356 **The developer shall provide an assurance rationale of the PP-Module for each of the alternative**
 1357 **PP-Module Bases identified in the PP-Module introduction.**
- 1358 **Content and presentation elements**
- 1359 **ACE_MCO.1.1C**
- 1360 **The consistency rationale shall demonstrate that the TOE type of the PP-Module and the TOE**
 1361 **types of its PP-Module Base(s) are consistent.**
- 1362 **ACE_MCO.1.2C**
- 1363 **The consistency rationale shall identify the assets of the PP-Module's SPD that also belong to**
 1364 **some of its PP-Module Bases and amongst them those for which the PP-Module and the PP-**
 1365 **Module Base define different security problems.**

1366 **ACE_MCO.1.3C**1367 **The consistency rationale shall demonstrate that:**

- 1368 • **the statement of the security problem definition is consistent with the statement of the**
1369 **security problem definition of its PP-Module Base(s);**
- 1370 • **the statement of the security problem definition is consistent with the statement of the**
1371 **security problem definition of any functional package for which conformance is being**
1372 **claimed.**

1373 **ACE_MCO.1.4C**1374 **The consistency rationale shall demonstrate that:**

- 1375 • **the security objectives definition is consistent with the security objectives of its PP-**
1376 **Module Base(s);**
- 1377 • **the security objectives definition is consistent with the security objectives of any**
1378 **functional package for which conformance is being claimed.**

1379 **ACE_MCO.1.5C**1380 **The consistency rationale shall demonstrate that:**

- 1381 • **the security functional requirements definition is consistent with the security**
1382 **functional requirements of its PP-Modules Base(s);**
- 1383 • **the security functional requirements definition is consistent with the security**
1384 **functional requirements of any functional package for which conformance is being**
1385 **claimed.**

1386 **ACE_MCO.1.6C**

1387 **The assurance rationale shall demonstrate the internal consistency of the set of security**
1388 **assurance requirements of the PP-Module with regard to its security problem definition.**

1389 **ACE_MCO.1.7C**

1390 **The assurance rationale shall demonstrate the consistency of the set of security assurance**
1391 **requirements of the PP-Module with regard to the security assurance requirements of the PP-**
1392 **Module Base(s).**

1393 **Evaluator action elements**1394 **ACE_MCO.1.1E**

1395 **The evaluator shall confirm that the information provided meets all requirements for content**
1396 **and presentation of evidence. If the PP-Module specifies alternative PP-Module Bases, the**
1397 **evaluator shall perform this action for each consistency rationale.**

- 1398 **8.9 PP-Configuration consistency (ACE_CCO)**
- 1399 **8.9.1 Objectives**
- 1400 The objective of this family is to determine the well-formedness and the consistency of the PP-
1401 Configuration.
- 1402 **8.9.2 ACE_CCO.1 PP-Configuration consistency**
- 1403 Dependencies: ACE_INT.1 PP-Module introduction
- 1404 ACE_CCL.1 PP-Module conformance claims
- 1405 ACE_SPD.1 PP-Module security problem definition
- 1406 ACE_OBJ.1 Direct Rationale PP-Module security objectives for the environment or
1407 ACE_OBJ.2 PP-Module security objectives
- 1408 ACE_ECD.1 PP-Module extended component definition
- 1409 ACE_REQ.1 Direct Rational PP-Module security requirements or ACE_REQ.2 PP-
1410 Module derived security requirements
- 1411 ACE_MCO.1 PP-Module consistency
- 1412 APE_* (all APE components)
- 1413 **Developer action elements**
- 1414 **ACE_CCO.1.1D**
- 1415 **The developer shall provide the reference of the PP-Configuration.**
- 1416 **ACE_CCO.1.2D**
- 1417 **The developer shall provide a components statement.**
- 1418 **ACE_CCO.1.3D**
- 1419 **The developer shall provide a TOE overview.**
- 1420 **ACE_CCO.1.4D**
- 1421 **The developer shall provide a conformance claim.**
- 1422 **ACE_CCO.1.5D**
- 1423 **The developer shall provide a conformance statement within the conformance claim.**
- 1424 **ACE_CCO.1.6D**
- 1425 **The developer shall provide a consistency rationale.**

- 1426 **ACE_CCO.1.7D**
- 1427 **The developer shall provide a SAR statement.**
- 1428 **ACE_CCO.1.8D**
- 1429 **The developer shall provide the set of Evaluation Methods and/or Activities that are applicable**
 1430 **to the PP-Configuration.**
- 1431 **Content and presentation elements**
- 1432 **ACE_CCO.1.1C**
- 1433 **The PP-Configuration reference shall uniquely identify the PP-Configuration.**
- 1434 **ACE_CCO.1.2C**
- 1435 **The PP-Configuration components statement shall uniquely identify the PPs and PP-Modules**
 1436 **that compose the PP-Configuration.**
- 1437 **ACE_CCO.1.3C**
- 1438 **For each PP-Module identified in the PP-Configuration components statement, the components**
 1439 **statement shall include the PP-Module Base required by the identified PP-Module. If the PP-**
 1440 **Module specifies alternative PP-Module Bases, only one of these PP-Module Bases shall be**
 1441 **referred to in the PP-Configuration.**
- 1442 **ACE_CCO.1.4C**
- 1443 **For a multi-assurance PP-Configuration, the components statement shall describe the**
 1444 **organisation of the TSF in terms of the sub-TSFs defined in the PPs and PP-Modules defined in**
 1445 **the PP-Configuration.**
- 1446 **ACE_CCO.1.5C**
- 1447 **The TOE overview shall identify the TOE type.**
- 1448 **ACE_CCO.1.6C**
- 1449 **The TOE overview shall describe the usage and major security features of the TOE.**
- 1450 **ACE_CCO.1.7C**
- 1451 **The TOE overview shall identify any non-TOE hardware/software/firmware available to the**
 1452 **TOE.**
- 1453 **ACE_CCO.1.8C**
- 1454 **The conformance claim shall identify the ISO/IEC 15408 edition(s) to which the PP-**
 1455 **Configuration components claim conformance.**
- 1456 **ACE_CCO.1.9C**
- 1457 **The conformance claim shall describe the conformance of the PP-Configuration to ISO/IEC**
 1458 **15408-2 as either ISO/IEC 15408-2 conformant or ISO/IEC 15408-2 extended.**

1459 **ACE_CCO.1.10C**

1460 **The conformance claim shall describe the conformance of the PP-Configuration to this**
 1461 **document as either “ISO/IEC 15408-3 conformant” or ISO/IEC 15408-3 extended.”**

1462 **ACE_CCO.1.11C**

1463 **The conformance claim shall be consistent with the conformance claims of the PP-**
 1464 **Configuration components.**

1465 **ACE_CCO.1.12C**

1466 **The conformance claim of a PP-Configuration shall include an assurance package conformance**
 1467 **claim consisting of statements describing any conformance of the PP-Configuration to an**
 1468 **assurance package as either package-conformant or package-augmented.**

1469 **ACE_CCO.1.13C**

1470 **The conformance statement shall specify the required conformance to the PP-Configuration as**
 1471 **one of exact, strict, demonstrable, or it shall provide the list of conformance types that are**
 1472 **required by each of the PP-Configuration components.**

1473 **ACE_CCO.1.14C**

1474 **For the exact conformance case, the allowed-with statement of the conformance statement of**
 1475 **each PP included in the components statement of the PP-Configuration shall identify all the PP-**
 1476 **Configuration components as being allowed to be used in combination with the PP in a PP-**
 1477 **Configuration.**

1478 **ACE_CCO.1.15C**

1479 **For the exact conformance case, the allowed-with statement of the conformance statement of**
 1480 **each PP-Module included in the components statement of the PP-Configuration shall identify all**
 1481 **the PP-Configuration components that are not in the PP-Module Base(s) for that particular PP-**
 1482 **Module as being allowed to be used in combination with the PP-Module in a PP-Configuration.**

1483 **ACE_CCO.1.16C**

1484 **For PP-Configurations that are not of exact conformance type (i.e. for PP-Configurations of**
 1485 **strict or demonstrable conformance type), the conformance statement of a PP-Configuration**
 1486 **may include an Evaluation Methods and Evaluation Activities reference statement that**
 1487 **identifies the set of ISO/IEC 18045-derived Evaluation Methods and Evaluation Activities that**
 1488 **are applicable to the PP-Configuration under evaluation.**

1489 **ACE_CCO.1.17C**

1490 **The consistency rationale shall demonstrate that the TOE type defined in the PP-Configuration**
 1491 **is consistent with the TOE types defined in the PPs and PP-Modules that belong to the PP-**
 1492 **Configuration components statement.**

1493 **ACE_CCO.1.18C**

1494 **The consistency rationale shall demonstrate that the union of all the SPDs, security objectives**
 1495 **and security functional requirements defined in the PP-Configuration components is**
 1496 **consistent.**

1497 **ACE_CCO.1.19C**

1498 **For a single-assurance PP-Configuration, the SAR statement shall define a single set of SARs**
 1499 **that applies to the entire TOE. For strict and demonstrable conformance, the set of SARs shall**
 1500 **include the SARs identified in each of the PP-Configuration components. For exact**
 1501 **conformance, the set of SARs shall be identical to the set of SARs identified in each of the PP-**
 1502 **Configuration components.**

1503 **ACE_CCO.1.20C**

1504 **For a multi-assurance PP-Configuration, the SAR statement shall define the global set of SARs**
 1505 **that applies to the entire TOE and the SARs that apply to each sub-TSF. For strict and**
 1506 **demonstrable conformance, the global assurance set of SARs shall include the set of common**
 1507 **SARs among the PP-Configuration components, and each set of SARs that apply to a sub-TSF**
 1508 **shall include those identified for the PP-Configuration components associated with that sub-**
 1509 **TSF. For exact conformance, the global assurance set of SARs shall be the set of common SARs**
 1510 **among the PP-Configuration components, and each set of SARs that apply to a sub-TSF shall be**
 1511 **identical to those identified for the PP-Configuration components associated with that sub-TSF.**

1512 **ACE_CCO.1.21C**

1513 **The SAR statement of a PP-Configuration shall include an assurance rationale that**
 1514 **demonstrates the consistency of the applicable set of SARs with those defined in the**
 1515 **components of the PP-Configuration under evaluation and their associated Evaluation Methods**
 1516 **and Evaluation Activities. For a multi-assurance PP-Configuration, the assurance rationale shall**
 1517 **demonstrate:**

- 1518 • **that the global set of SARs is consistent with the threats as defined in the SPDs of the PP-**
 1519 **Configuration components, and**
- 1520 • **that the global set of SARs and the sets of SARs for each sub-TSF are consistent with each**
 1521 **other.**

1522 **Evaluator action elements**1523 **ACE_CCO.1.1E**

1524 **The evaluator shall confirm that the information provided meets all requirements for content**
 1525 **and presentation of evidence.**

1526 **ACE_CCO.1.2E**

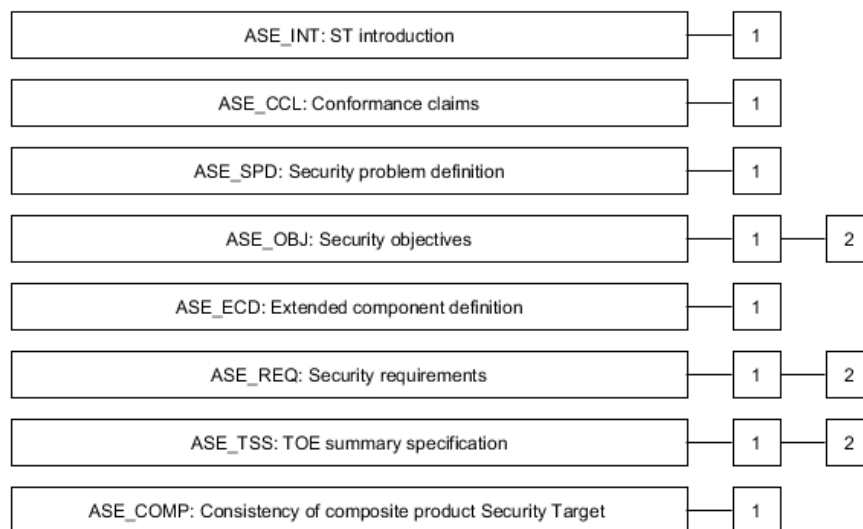
1527 **The evaluator shall check that the PP-Configuration consisting of all the PPs and PP-Modules**
 1528 **identified in the component statement is consistent.**

1529 **9 Class ASE: Security Target evaluation**1530 **9.1 Introduction**

1531 **Evaluating an ST is required to demonstrate that the ST is sound and internally consistent, and, if the**
 1532 **ST is based on a PP-Configuration, or one or more PPs or packages, that the ST is a correct**
 1533 **instantiation of the PP-Configuration, PPs, and packages. These properties are necessary for the ST to**
 1534 **be suitable for use as the basis for a TOE evaluation.**

1535 Clause 9 should be used in conjunction with Annexes B, C and D in ISO/IEC 15408-1:XXXX, as these
 1536 annexes clarify the concepts here and provide many examples.

1537 Figure 6 shows the families within this class, and the hierarchy of components within the families.



1538

1539 **Figure 6 — ASE: Security Target evaluation class decomposition**

1540 **9.2 ST introduction (ASE_INT)**

1541 **9.2.1 Objectives**

1542 The objective of this family is to describe the TOE in a narrative way on three levels of abstraction:
 1543 TOE reference, TOE overview and TOE description.

1544 Evaluation of the ST introduction is required to demonstrate that the ST and the TOE are correctly
 1545 identified, that the TOE is correctly described at three levels of abstraction and that these three
 1546 descriptions are consistent with each other.

1547 **9.2.2 ASE_INT.1 ST introduction**

1548 Dependencies: No dependencies.

1549 **Developer action elements**

1550 **ASE_INT.1.1D**

1551 **The developer shall provide an ST introduction.**

1552 **Content and presentation elements**

1553 **ASE_INT.1.1C**

1554 **The ST introduction shall contain an ST reference, a TOE reference, a TOE overview and a TOE**
 1555 **description.**

- 1556 **ASE_INT.1.2C**
- 1557 **The ST reference shall uniquely identify the ST.**
- 1558 **ASE_INT.1.3C**
- 1559 **The TOE reference shall uniquely identify the TOE.**
- 1560 **ASE_INT.1.4C**
- 1561 **The TOE overview shall summarise the usage and major security features of the TOE.**
- 1562 **ASE_INT.1.5C**
- 1563 **The TOE overview shall identify the TOE type.**
- 1564 **ASE_INT.1.6C**
- 1565 **The TOE overview shall identify any non-TOE hardware/software/firmware required by the**
 1566 **TOE.**
- 1567 **ASE_INT.1.7C**
- 1568 **For a multi-assurance ST, the TOE overview shall describe the TSF organization in terms of the**
 1569 **sub-TSFs defined in the PP-Configuration the ST claims conformance to.**
- 1570 **ASE_INT.1.8C**
- 1571 **The TOE description shall describe the physical scope of the TOE.**
- 1572 **ASE_INT.1.9C**
- 1573 **The TOE description shall describe the logical scope of the TOE.**
- 1574 **Evaluator action elements**
- 1575 **ASE_INT.1.1E**
- 1576 **The evaluator shall confirm that the information provided meets all requirements for content**
 1577 **and presentation of evidence.**
- 1578 **ASE_INT.1.2E**
- 1579 **The evaluator shall confirm that the TOE reference, the TOE overview, and the TOE description**
 1580 **are consistent with each other.**
- 1581 **9.3 Conformance claims (ASE_CCL)**
- 1582 **9.3.1 Objectives**
- 1583 The objective of this family is to determine the validity of the conformance claim. In addition, this
 1584 family specifies how STs are to claim conformance with the PP or PP-Configuration.
- 1585 **9.3.2 ASE_CCL.1 Conformance claims**
- 1586 Dependencies: ASE_INT.1 ST introduction

- 1587 ASE_ECD.1 Extended components definition
- 1588 ASE_REQ.1 Direct Rationale security requirements
- 1589 **Developer action elements**
- 1590 **ASE_CCL.1.1D**
- 1591 **The developer shall provide a conformance claim.**
- 1592 **ASE_CCL.1.2D**
- 1593 **The developer shall provide a conformance claim rationale.**
- 1594 **Content and presentation elements**
- 1595 **ASE_CCL.1.1C**
- 1596 **The conformance claim shall identify the edition of ISO/IEC 15408 to which the ST and the TOE**
 1597 **claim conformance.**
- 1598 **ASE_CCL.1.2C**
- 1599 **The conformance claim shall describe the conformance of the ST to ISO/IEC 15408-2 as either**
 1600 **ISO/IEC 15408-2 conformant or ISO/IEC 15408-2 extended.**
- 1601 **ASE_CCL.1.3C**
- 1602 **The conformance claim shall describe the conformance of the ST as either “ISO/IEC 15408-3**
 1603 **conformant” or ISO/IEC 15408-3 extended.”**
- 1604 **ASE_CCL.1.4C**
- 1605 **The conformance claim shall be consistent with the extended components definition.**
- 1606 **ASE_CCL.1.5C**
- 1607 **The conformance claim shall identify a PP-Configuration, or all PPs and security requirement**
 1608 **packages to which the ST claims conformance.**
- 1609 **ASE_CCL.1.6C**
- 1610 **The conformance claim shall describe any conformance of the ST to a package as either**
 1611 **package-conformant or package-augmented.**
- 1612 **ASE_CCL.1.7C**
- 1613 **The conformance claim shall describe any conformance of the ST to a PP as PP-Conformant.**
- 1614 **ASE_CCL.1.8C**
- 1615 **If the ST claims conformance to a PP-Configuration, the conformance shall be included in the**
 1616 **conformance claim. A ST shall claim conformance to exactly one PP-Configuration and no**
 1617 **additional PP or functional package.**

1618 ASE_CCL.1.9C

1619 The conformance claim rationale shall demonstrate that the TOE type is consistent with the
1620 TOE type in the PP-Configuration or PPs for which conformance is being claimed.

1621 ASE_CCL.1.10C

1622 The conformance claim rationale shall demonstrate that the statement of the security problem
1623 definition is consistent with the statement of the security problem definition in the PP-
1624 Configuration¹, PPs and any functional packages for which conformance is being claimed.

1625 ASE_CCL.1.11C

1626 The conformance claim rationale shall demonstrate that the statement of security objectives is
1627 consistent with the statement of security objectives in the PP-Configuration², PPs, and any
1628 functional package for which conformance is being claimed.

1629 ASE_CCL.1.12C

1630 The conformance claim rationale shall demonstrate that the statement of security
1631 requirements is consistent with the statement of security requirements in the PP-
1632 Configuration³, PPs, and any functional packages for which conformance is being claimed.

1633 ASE_CCL.1.13C

1634 The conformance claim for PP(s) or a PP-Configuration shall be exact, strict, or demonstrable
1635 or a list of conformance types.

1636 ASE_CCL.1.14C

1637 If the conformance claim identifies a set of Evaluation Methods and Evaluation Activities
1638 derived from ISO/IEC 18045 work units that shall be used to evaluate the TOE then this set
1639 shall include all those that are included in any package, PP, or PP-Module in a PP-Configuration
1640 to which the ST claims conformance, and no others.

1641 Evaluator action elements

1642 ASE_CCL.1.1E

1643 The evaluator shall confirm that the information provided meets all requirements for content
1644 and presentation of evidence.

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

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

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

1645 **9.4 Security problem definition (ASE_SPD)**

1646 **9.4.1 Objectives**

1647 This part of the ST defines the security problem to be addressed by the TOE and the operational
1648 environment of the TOE.

1649 Evaluation of the security problem definition is required to demonstrate that the security problem
1650 intended to be addressed by the TOE and its operational environment, is clearly defined.

1651 **9.4.2 ASE_SPD.1 Security problem definition**

1652 Dependencies: No dependencies.

1653 **Developer action elements**

1654 **ASE_SPD.1.1D**

1655 **The developer shall provide a security problem definition.**

1656 **Content and presentation elements**

1657 **ASE_SPD.1.1C**

1658 **The security problem definition shall describe the threats.**

1659 **ASE_SPD.1.2C**

1660 **All threats shall be described in terms of a threat agent, an asset, and an adverse action.**

1661 **ASE_SPD.1.3C**

1662 **The security problem definition shall describe the OSPs.**

1663 **ASE_SPD.1.4C**

1664 **The security problem definition shall describe the assumptions about the operational**
1665 **environment of the TOE.**

1666 **Evaluator action elements**

1667 **ASE_SPD.1.1E**

1668 **The evaluator shall confirm that the information provided meets all requirements for content**
1669 **and presentation of evidence.**

1670 **9.5 Security objectives (ASE_OBJ)**

1671 **9.5.1 Objectives**

1672 The security objectives are a concise statement of the intended response to the security problem
1673 defined through the Security problem definition (ASE_SPD) family.

1674 Evaluation of the security objectives is required to demonstrate that the security objectives adequately
1675 and completely address the security problem definition, that the division of this problem between the
1676 TOE and its operational environment is clearly defined.

1677 **9.5.2 Component levelling**

1678 The components in this family are levelled on whether they prescribe only security objectives for the
1679 operational environment (ASE_OBJ.1), or also security objectives for the TOE (ASE_OBJ.2).

1680 **9.5.3 ASE_OBJ.1 Direct rationale Security objectives for the operational environment**

1681 Dependencies: No dependencies

1682 **Developer action elements**

1683 **ASE_OBJ.1.1D**

1684 **The developer shall provide a statement of security objectives.**

1685 **ASE_OBJ.1.2D**

1686 **The developer shall provide a security objectives rationale.**

1687 **Content and presentation elements**

1688 **ASE_OBJ.1.1C**

1689 **The statement of security objectives shall describe the security objectives for the operational**
1690 **environment.**

1691 **ASE_OBJ.1.2C**

1692 **The security objectives rationale shall trace each security objective for the operational**
1693 **environment back to threats countered by that security objective, OSPs enforced by that**
1694 **security objective, and assumptions upheld by that security objective.**

1695 **ASE_OBJ.1.3C**

1696 **The security objectives rationale shall demonstrate that the security objectives for the**
1697 **operational environment uphold all assumptions.**

1698 **Evaluator action elements**

1699 **ASE_OBJ.1.1E**

1700 **The evaluator shall confirm that the information provided meets all requirements for content**
1701 **and presentation of evidence.**

1702 **9.5.4 ASE_OBJ.2 Security objectives**

1703 Dependencies: ASE_SPD.1 Security problem definition

1704 **Developer action elements**

1705 **ASE_OBJ.2.1D**

1706 **The developer shall provide a statement of security objectives.**

1707 **ASE_OBJ.2.2D**

1708 The developer shall provide a security objectives rationale.

1709 **Content and presentation elements**1710 **ASE_OBJ.2.1C**

1711 **The statement of security objectives shall describe the security objectives for the TOE and the**
 1712 **security objectives for the operational environment.**

1713 **ASE_OBJ.2.2C**

1714 **The security objectives rationale shall trace each security objective for the TOE back to threats**
 1715 **countered by that security objective and OSPs enforced by that security objective.**

1716 **ASE_OBJ.2.3C**

1717 **The security objectives rationale shall trace each security objective for the operational**
 1718 **environment back to threats countered by that security objective, OSPs enforced by that**
 1719 **security objective, and assumptions upheld by that security objective.**

1720 **ASE_OBJ.2.4C**1721 The security objectives **rationale** shall **demonstrate that** the security objectives **counter all threats.**1722 **ASE_OBJ.2.5C**

1723 **The security objectives rationale shall demonstrate that the security objectives enforce all**
 1724 **OSP.**

1725 **ASE_OBJ.2.6C**

1726 **The security objectives rationale shall demonstrate that the security objectives for the**
 1727 **operational environment uphold all assumptions.**

1728 **Evaluator action elements**1729 **ASE_OBJ.2.1E**

1730 **The evaluator shall confirm that the information provided meets all requirements for content**
 1731 **and presentation of evidence.**

1732 **9.6 Extended components definition (ASE_ECD)**1733 **9.6.1 Objectives**

1734 Extended security requirements are requirements that are not based on components from ISO/IEC
 1735 15408-2 or this document, but are based on extended components: components defined by the ST
 1736 author.

1737 Evaluation of the definition of extended components is necessary to determine that they are clear and
 1738 unambiguous, and that they are necessary, i.e. they may not be clearly expressed using existing
 1739 ISO/IEC 15408-2 or this document components.

1740 **9.6.2 ASE_ECD.1 Extended components definition**

1741 Dependencies: No dependencies.

1742 **Developer action elements**

1743 **ASE_ECD.1.1D**

1744 **The developer shall provide a statement of security requirements.**

1745 **ASE_ECD.1.2D**

1746 **The developer shall provide an extended components definition.**

1747 **Content and presentation elements**

1748 **ASE_ECD.1.1C**

1749 **The statement of security requirements shall identify all extended security requirements.**

1750 **ASE_ECD.1.2C**

1751 **The extended components definition shall define an extended component for each extended**
 1752 **security requirement.**

1753 **ASE_ECD.1.3C**

1754 **The extended components definition shall describe how each extended component is related to**
 1755 **the existing ISO/IEC 15408 series components, families, and classes.**

1756 **ASE_ECD.1.4C**

1757 **The extended components definition shall use the existing ISO/IEC 15408 series components,**
 1758 **families, classes, and methodology as a model for presentation.**

1759 **ASE_ECD.1.5C**

1760 **The extended components shall consist of measurable and objective elements such that**
 1761 **conformance or nonconformance to these elements may be demonstrated.**

1762 **Evaluator action elements**

1763 **ASE_ECD.1.1E**

1764 **The evaluator shall confirm that the information provided meets all requirements for content**
 1765 **and presentation of evidence.**

1766 **ASE_ECD.1.2E**

1767 **The evaluator shall confirm that no extended component may be clearly expressed using**
 1768 **existing components.**

1769 **9.7 Security requirements (ASE_REQ)**

1770 **9.7.1 Objectives**

1771 The SFRs form a clear, unambiguous and well-defined description of the expected security behaviour
1772 of the TOE. The SARs form a clear, unambiguous and canonical description of the expected activities
1773 that will be undertaken to gain assurance in the TOE.

1774 Evaluation of the security requirements is required to ensure that they are clear, unambiguous and
1775 well-defined.

1776 **9.7.2 Component levelling**

1777 The components in this family are levelled on whether they are stated as is (see ASE_REQ.1), or
1778 whether the SFRs are derived from security objectives for the TOE (see ASE_REQ.2.).

1779 **9.7.3 ASE_REQ.1 Direct Rationale security requirements**

1780 Dependencies: ASE_ECD.1 Extended components definition

1781 **Developer action elements**

1782 **ASE_REQ.1.1D**

1783 **The developer shall provide a statement of security requirements.**

1784 **ASE_REQ.1.2D**

1785 **The developer shall provide a security requirements rationale.**

1786 **Content and presentation elements**

1787 **ASE_REQ.1.1C**

1788 **The statement of security requirements shall describe the SFRs and the SARs.**

1789 **ASE_REQ.1.2C**

1790 **For a single-assurance ST, the statement of security requirements shall define the global set of**
1791 **SARs that apply to the entire TOE. The sets of SARs shall be consistent with the PPs or PP-**
1792 **Configuration to which the ST claims conformance.**

1793 **ASE_REQ.1.3C**

1794 **For a multi-assurance ST, the statement of security requirements shall define the global set of**
1795 **SARs that apply to the entire TOE and the sets of SARs that apply to each sub-TSF. The sets of**
1796 **SARs shall be consistent with the multi-assurance PP-Configuration to which the ST claims**
1797 **conformance.**

1798 **ASE_REQ.1.4C**

1799 **All subjects, objects, operations, security attributes, external entities and other terms that are**
1800 **used in the SFRs and the SARs shall be defined.**

1801 ASE_REQ.1.5C

1802 The statement of security requirements shall include a natural language description, part of
 1803 which describes how the SFRs combine together to provide security functionality in terms of
 1804 the architecture that is observable to Administrators and other users, or in terms of internal
 1805 features or properties.

1806 ASE_REQ.1.6C

1807 The statement of security requirements shall identify all operations on the security
 1808 requirements.

1809 ASE_REQ.1.7C

1810 All operations shall be performed correctly.

1811 ASE_REQ.1.8C

1812 Each dependency of the security requirements shall either be satisfied, or the security
 1813 requirements rationale shall justify the dependency not being satisfied.

1814 ASE_REQ.1.9C

1815 The security requirements rationale shall demonstrate that the SFRs (in conjunction with the
 1816 security objectives for the environment) counter all threats for the TOE.

1817 ASE_REQ.1.10C

1818 The security requirements rationale shall demonstrate that the SFRs (in conjunction with the
 1819 security objectives for the environment) enforce all OSPs.

1820 ASE_REQ.1.11C

1821 The security requirements rationale shall explain why the SARs were chosen.

1822 ASE_REQ.1.12C

1823 The statement of security requirements shall be internally consistent.

1824 ASE_REQ.1.13C

1825 If the ST defines sets of SARs that expand the sets of SARs of the PPs or the PP-Configuration it
 1826 claims conformance to, the security requirements rationale shall include an assurance
 1827 rationale that justifies the consistency of the extension and provides a rationale for the
 1828 disposition of any Evaluation Methods and Evaluation Activities identified in the conformance
 1829 statement that are affected by the extension of the sets of SARs.

1830 Evaluator action elements

1831 ASE_REQ.1.1E

1832 The evaluator shall confirm that the information provided meets all requirements for content
 1833 and presentation of evidence.

1834 **9.7.4 ASE_REQ.2 Derived security requirements**

1835 Dependencies: ASE_OBJ.2 Security objectives

1836 ASE_ECD.1 Extended components definition

1837 **Developer action elements**

1838 **ASE_REQ.2.1D**

1839 The developer shall provide a statement of security requirements.

1840 **ASE_REQ.2.2D**

1841 The developer shall provide a security requirements rationale.

1842 **Content and presentation elements**

1843 **ASE_REQ.2.1C**

1844 The statement of security requirements shall describe the SFRs and the SARs.

1845 **ASE_REQ.2.2C**

1846 For a single-assurance ST, the statement of security requirements shall define the global set of SARs
1847 that apply to the entire TOE. The sets of SARs shall be consistent with the PPs or PP-Configuration to
1848 which the ST claims conformance.

1849 **ASE_REQ.2.3C**

1850 For a multi-assurance ST, the statement of security requirements shall define the global set of SARs
1851 that apply to the entire TOE and the sets of SARs that apply to each sub-TSF. The sets of SARs shall be
1852 consistent with the multi-assurance PP-Configuration to which the ST claims conformance.

1853 **ASE_REQ.2.4C**

1854 All subjects, objects, operations, security attributes, external entities and other terms that are used in
1855 the SFRs and the SARs shall be defined.

1856 **ASE_REQ.2.5C**

1857 The statement of security requirements shall identify all operations on the security requirements.

1858 **ASE_REQ.2.6C**

1859 All operations shall be performed correctly.

1860 **ASE_REQ.2.7C**

1861 Each dependency of the security requirements shall either be satisfied, or the security requirements
1862 rationale shall justify the dependency not being satisfied.

1863 **ASE_REQ.2.8C**

1864 **The security requirements rationale shall demonstrate that the SFRs meet all security**
1865 **objectives for the TOE.**

1866 **ASE_REQ.2.9C**

1867 The security requirements rationale shall explain why the SARs were chosen.

1868 **ASE_REQ.2.10C**

1869 The statement of security requirements shall be internally consistent.

1870 **ASE_REQ.2.11C**

1871 **If the ST defines sets of SARs that expand the sets of SARs of the PPs or PP-Configuration it**
 1872 **claims conformance to, the security requirements rationale shall include an assurance**
 1873 **rationale that justifies the consistency of the extension and provides a rationale for the**
 1874 **disposition of any Evaluation Methods and Evaluation Activities identified in the conformance**
 1875 **statement that are affected by the extension of the sets of SARs.**

1876 **Evaluator action elements**1877 **ASE_REQ.2.1E**

1878 The evaluator shall confirm that the information provided meets all requirements for content and
 1879 presentation of evidence.

1880 **9.8 TOE summary specification (ASE_TSS)**1881 **9.8.1 Objectives**

1882 The TOE summary specification enables evaluators and potential consumers to gain a general
 1883 understanding of how the TOE is implemented.

1884 Evaluation of the TOE summary specification is necessary to determine whether it is adequately
 1885 described how the TOE:

- 1886 • meets its SFRs;
- 1887 • protects itself against interference, logical tampering and bypass;

1888 and whether the TOE summary specification is consistent with other narrative descriptions of the
 1889 TOE.

1890 **9.8.2 Component levelling**

1891 The components in this family are levelled on whether the TOE summary specification only needs to
 1892 describe how the TOE meets the SFRs, or whether the TOE summary specification also needs to
 1893 describe how the TOE protects itself against logical tampering and bypass. This additional description
 1894 may be used in special circumstances where there might be a specific concern regarding the TOE
 1895 security architecture.

1896 **9.8.3 ASE_TSS.1 TOE summary specification**

1897 Dependencies: ASE_INT.1 ST introduction

1898 ASE_REQ.1 Direct Rationale security requirements

1899 ADV_FSP.1 Basic functional specification

- 1900 **Developer action elements**
- 1901 **ASE_TSS.1.1D**
- 1902 **The developer shall provide a TOE summary specification.**
- 1903 **Content and presentation elements**
- 1904 **ASE_TSS.1.1C**
- 1905 **The TOE summary specification shall describe how the TOE meets each SFR.**
- 1906 **Evaluator action elements**
- 1907 **ASE_TSS.1.1E**
- 1908 **The evaluator shall confirm that the information provided meets all requirements for content**
 1909 **and presentation of evidence.**
- 1910 **ASE_TSS.1.2E**
- 1911 **The evaluator shall confirm that the TOE summary specification is consistent with the TOE**
 1912 **overview and the TOE description.**
- 1913 **9.8.4 ASE_TSS.2 TOE summary specification with architectural design summary**
- 1914 Dependencies: ASE_INT.1 ST introduction
- 1915 ASE_REQ.1 Direct Rationale security requirements
- 1916 ADV_ARC.1 Security architecture description
- 1917 **Developer action elements**
- 1918 **ASE_TSS.2.1D**
- 1919 **The developer shall provide a TOE summary specification.**
- 1920 **Content and presentation elements**
- 1921 **ASE_TSS.2.1C**
- 1922 **The TOE summary specification shall describe how the TOE meets each SFR.**
- 1923 **ASE_TSS.2.2C**
- 1924 **The TOE summary specification shall describe how the TOE protects itself against interference**
 1925 **and logical tampering.**
- 1926 **ASE_TSS.2.3C**
- 1927 **The TOE summary specification shall describe how the TOE protects itself against bypass.**

1928 **Evaluator action elements**

1929 **ASE_TSS.2.1E**

1930 The evaluator shall confirm that the information provided meets all requirements for content and
1931 presentation of evidence.

1932 **ASE_TSS.2.2E**

1933 The evaluator shall confirm that the TOE summary specification is consistent with the TOE overview
1934 and the TOE description.

1935 **9.9 Consistency of composite product Security Target (ASE_COMP)**

1936 **9.9.1 Objectives**

1937 The aim of this family is to determine whether the Security Target of the composite product⁴ does not
1938 contradict the Security Target of the related base component^{5,6}.

1939 **9.9.1 Component levelling**

1940 This family contains only one component.

1941 **9.9.2 ASE_COMP.1 Consistency of Security Target**

1942 Dependencies: No dependencies

1943 **9.9.3 Application notes**

1944 A Security Target for the composite product has to be written and evaluated.

1945 The composite product evaluator has to examine that the Security Target of the composite product
1946 does not contradict the Security Target of the related base component. In particular, it means that the
1947 composite product evaluator has to examine the composite product Security Target and the base
1948 component Security Target for any conflicting assumptions, compatibility of security objectives,
1949 security requirements and security functionality needed by the dependent component.

1950 The composite product evaluation sponsor shall ensure that the Security Target of the base
1951 component is available for the dependent component developer, for the composite product evaluator
1952 and for the composite product evaluation authority. The information available in the public version of
1953 the base component Security Target may not be sufficient.

1954 These application notes aid the developer to create as well as the evaluator to analyse a composite
1955 Security Target and describe a general methodology for it.

1956 In order to create a composite product Security Target, the developer should perform the following
1957 steps:

⁴ denoted by composite product Security Target or composite-ST in the following

⁵ denoted by base component Security Target or base-ST in the following

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

- 1958 Step 1: The developer formulates a preliminary Security Target for the composite product (the
 1959 composite-ST) using the standard code of practice. The composite-ST can be formulated
 1960 independently of the Security Target of the composite product's related base component (the base-ST)
 1961 – at least as long as there are no formal PP conformance claims.
- 1962 Step 2: The developer determines the overlap between the base-ST and the composite-ST through
 1963 analysing and comparing their respective TOE Security Functionality (TSF) ⁷⁸.
- 1964 Step 3: The developer determines under which conditions he can trust in and rely on the base
 1965 component-TSF being used by the Composite-ST without a new examination.
- 1966 Having undertaken these steps the developer completes the preliminary Security Target for the
 1967 composite product.
- 1968 It is not mandatory that the composite product and its related base component are being evaluated
 1969 according to the same edition of ISO/IEC 15408. It is due to the fact that the dependent component of
 1970 the composite product can rely on some security services of the base component, if (i) the assurance
 1971 level of the base component covers the intended assurance level of the composite product and (ii) the
 1972 base component evaluation is valid (i.e. accepted by the base component evaluation authority) and up-
 1973 to-date. Equivalence of single assurance components (and, hence, of assurance levels) belonging to
 1974 different ISO/IEC 15408 series editions have to be established / acknowledged by the composite
 1975 product evaluation authority.
- 1976 If conformance to a PP is claimed, e.g. a composite product Security Target claims conformance to a PP
 1977 (that possibly claims conformance to a further PP), the consistency check can be reduced to the
 1978 elements of the Security Target having not already been covered by these PPs. However, in general the
 1979 fact of compliance to a PP is not sufficient to avoid inconsistencies. Assume the following situation,
 1980 where → stands for “complies with”:
- 1981 composite-ST → PP 1 → PP 2 ← base-ST
- 1982 PP 1 may require any kind of conformance⁹, but this does not affect the ‘additional elements’ that the
 1983 base-ST may introduce beyond PP 2. In conclusion, these additions are not necessarily consistent with
 1984 the composite-ST's additions chosen beyond PP 1. There is no scenario that ensures their consistency
 1985 ‘by construction’.
- 1986 Note that consistency may be no direct matching: Objectives for the base component's environment
 1987 may become objectives for the composite TOE.
- 1988 **9.9.4 ASE_COMP.1 Consistency of Security Target**
- 1989 Dependencies: No dependencies

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

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

⁹ e.g. “strict”, “exact” or “demonstrable” according to ISO/IEC 15408

1990	9.9.4.1 Developer action elements
1991	9.9.4.1.1 ASE_COMP.1.1D
1992	The developer shall provide a statement of compatibility between the composite product Security Target and the base component Security Target. This statement may be provided within the composite product Security Target.
1993	
1994	
1995	9.9.4.2 Content and presentation elements
1996	9.9.4.2.1 ASE_COMP.1.1C
1997	The statement of compatibility shall describe the separation of the base component-TSF into relevant base component-TSF being used by the composite product Security Target and others.
1998	
1999	9.9.4.2.2 ASE_COMP.1.2C
2000	The statement of compatibility between the composite product Security Target and the base component Security Target shall show (e.g. in form of a mapping) that the Security Targets of the composite product and of the related base component match, i.e. that there is no conflict between security environments, security objectives, and security requirements of the composite product Security Target and the base component Security Target. It may be provided by indicating the concerned elements directly in the composite product Security Target followed by explanatory text, if necessary.
2001	
2002	
2003	
2004	
2005	
2006	
2007	9.9.4.3 Evaluator action elements
2008	9.9.4.3.1 ASE_COMP.1.1E
2009	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
2010	

2011 **10 Class ADV: Development**

2012 **10.1 Introduction**

2013 The requirements of the Development class provide information about the TOE. The knowledge
 2014 obtained by this information is used as the basis for conducting vulnerability analysis and testing upon
 2015 the TOE, as described in the AVA and ATE classes.

2016 The Development class encompasses seven families of requirements for structuring and representing
 2017 the TSF at various levels and varying forms of abstraction. These families include:

- 2018 • requirements for the description (at the various levels of abstraction) of the design and
 2019 implementation of the SFRs (ADV_FSP, ADV_TDS, ADV_IMP and ADV_COMP)
- 2020 • requirements for the description of the architecture-oriented features of domain separation, TSF
 2021 self-protection and non-bypassability of the security functionality (ADV_ARC)
- 2022 • requirements for a security policy model and for correspondence mappings between security
 2023 policy model and the functional specification (ADV_SPM)
- 2024 • requirements on the internal structure of the TSF, which covers aspects such as modularity,
 2025 layering, and minimisation of complexity (ADV_INT)

2026 When documenting the security functionality of a TOE, there are two properties that need to be
 2027 demonstrated. The first property is that the security functionality works correctly; that is, it performs
 2028 as specified. The second property, and one that is arguably harder to demonstrate, is that the TOE
 2029 cannot be used in a way such that the security functionality can be corrupted or bypassed. These two
 2030 properties require somewhat different approaches in analysis, and so the families in ADV are
 2031 structured to support these different approaches. The families Functional specification (ADV_FSP),
 2032 TOE design (ADV_TDS), Implementation representation (ADV_IMP), and Security policy modelling
 2033 (ADV_SPM) deal with the first property: the specification of the security functionality. The families
 2034 Security Architecture (ADV_ARC) and TSF internals (ADV_INT) deal with the second property: the
 2035 specification of the design of the TOE demonstrating the security functionality cannot be corrupted or
 2036 bypassed. It should be noted that both properties need to be realised: the more confidence one has
 2037 that the properties are satisfied, the more trustworthy the TOE is. The TSF of a composite product are
 2038 represented at various levels of abstraction in the families of the development class ADV. The family
 2039 Composite design compliance (ADV_COMP) determines whether the requirements on the dependent
 2040 component, imposed by the related base componen, are fulfilled in a composite product. Due to the
 2041 distribution of the TSF of a composite product to various levels in the families of the class ADV, this
 2042 family is not represented in Figure 7. The components in the families are designed so that more
 2043 assurance can be gained as the components hierarchically increase.

2044 The paradigm for the families targeted at the first property is one of design decomposition. At the
 2045 highest level, there is a functional specification of the TSF in terms of its interfaces (describing *what*
 2046 the TSF does in terms of requests to the TSF for services and resulting responses), decomposing the
 2047 TSF into smaller units (dependent on the assurance desired and the complexity of the TOE) and
 2048 describing *how* the TSF accomplishes its functions (to a level of detail commensurate with the
 2049 assurance level), and showing the implementation of the TSF. A formal model of the security
 2050 behaviour also may be given. All levels of decomposition are used in determining the completeness
 2051 and accuracy of all other levels, ensuring that the levels are mutually supportive. The requirements for
 2052 the various TSF representations are separated into different families, to allow the PP/ST author to
 2053 specify which TSF representations are required. The level chosen will dictate the assurance
 2054 desired/gained.

Figure 7 indicates the relationships among the various TSF representations of the ADV class, as well as their relationships with other classes. As the figure indicates, the APE and ASE classes define the requirements for the correspondence between the SFRs and the security objectives for the TOE. Class ASE also defines requirements for the correspondence between both the security objectives and SFRs, and for the TOE summary specification which explains how the TOE meets its SFRs. The activities of ALC_CMC.5.2E include the verification that the TSF that is tested under the ATE and AVA classes is in fact the one described by all of the ADV decomposition levels.

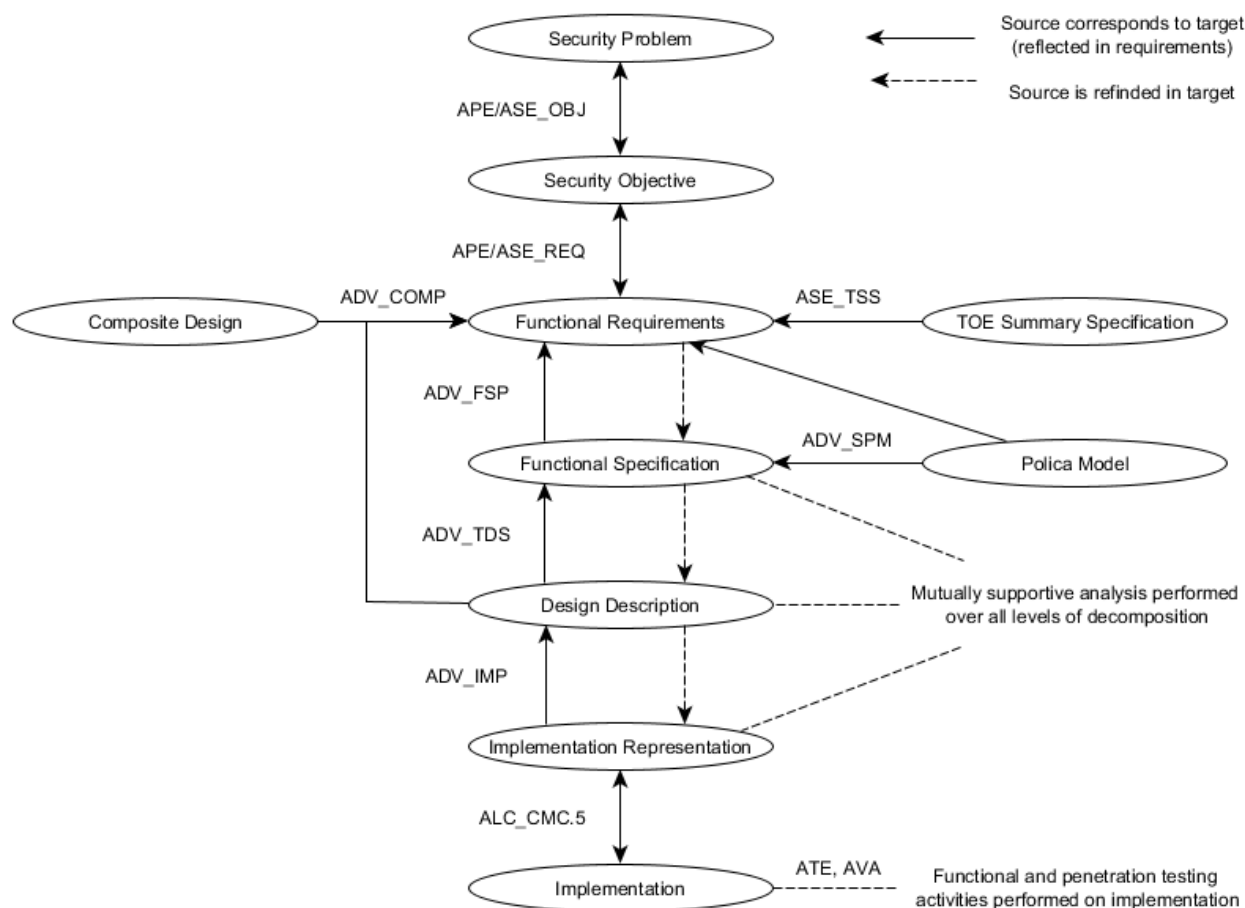


Figure 7 — Relationships of ADV constructs to one another and to other families

The requirements for all other correspondence shown in Figure 7 are defined in the ADV class for the TOE. The Security policy modelling (ADV_SPM) family defines the requirements for formally modelling selected SFRs, and providing correspondence between the functional specification and the formal model. Each assurance family specific to a TSF representation (i.e., Functional specification (ADV_FSP), TOE design (ADV_TDS) and Implementation representation (ADV_IMP)) defines requirements relating that TSF representation to the SFRs. All decompositions must accurately reflect all other decompositions (i.e., be mutually supportive); the developer supplies the tracings in the last .C elements of the components. Assurance relating to this factor is obtained during the analysis for each of the levels of decomposition by referring to other levels of decomposition (in a recursive fashion) while the analysis of a particular level of decomposition is being performed; the evaluator verifies the correspondence as part of the second E element. The understanding gained from these levels of decomposition form the basis of the functional and penetration testing efforts.

2076 The ADV_INT family is not represented in this figure, as it is related to the internal structure of the TSF,
 2077 and is only indirectly related to the process of refinement of the TSF representations. Similarly, the
 2078 ADV_ARC family is not represented in the figure because it relates to the architectural soundness,
 2079 rather than representation, of the TSF. Both ADV_INT and ADV_ARC relate to the analysis of the
 2080 property that the TOE cannot be made to circumvent or corrupt its security functionality.

2081 The TOE security functionality (TSF) consists of all parts of the TOE that have to be relied upon for
 2082 enforcement of the SFRs. The TSF includes both functionality that directly enforces the SFRs, as well as
 2083 functionality that, while not directly enforcing the SFRs, contributes to their enforcement in a more
 2084 indirect manner, including functionality with the capability to cause the SFRs to be violated. This
 2085 includes portions of the TOE that are invoked on start-up that are responsible for putting the TSF into
 2086 its initial secure state.

2087 Several important concepts were used in the development of the components of the ADV families.
 2088 These concepts, while introduced briefly here, are explained more fully in the application notes for the
 2089 families.

2090 One over-riding notion is that, as more information becomes available, greater assurance can be
 2091 obtained that the security functionality 1) is correctly implemented; 2) cannot be corrupted; and 3)
 2092 cannot be bypassed. This is done through the verification that the documentation is correct and
 2093 consistent with other documentation, and by providing information that can be used to ensure that the
 2094 testing activities (both functional and penetration testing) are comprehensive. This is reflected in the
 2095 levelling of the components of the families. In general, components are levelled based on the amount of
 2096 information that is to be provided (and subsequently analysed).

2097 While not true for all TOEs, it is generally the case that the TSF is sufficiently complex that there are
 2098 portions of the TSF that deserve more intense examination than other portions of the TSF.
 2099 Determining those portions is unfortunately somewhat subjective, thus terminology and components
 2100 have been defined such that as the level of assurance increases, the responsibility for determining
 2101 what portions of the TSF need to be examined in detail shifts from the developer to the evaluator. To
 2102 aid in expressing this concept, the following terminology is introduced. It should be noted that in the
 2103 families of the class, this terminology is used when expressing SFR-related portions of the TOE (that is,
 2104 elements and work units embodied in the Functional specification (ADV_FSP), TOE design (ADV_TDS),
 2105 and Implementation representation (ADV_IMP) families). While the general concept (that some
 2106 portions of the TOE are more *interesting* than others) applies to other families, the criteria are
 2107 expressed differently in order to obtain the assurance required.

2108 All portions of the TSF are *security relevant*, meaning that they must preserve the security of the TOE
 2109 as expressed by the SFRs and requirements for domain separation and non-bypassability. One aspect
 2110 of security relevance is the degree to which a portion of the TSF enforces a security requirement. Since
 2111 different portions of the TOE play different roles (or no apparent role at all) in enforcing security
 2112 requirements, this creates a continuum of SFR relevance: at one end of this continuum are portions of
 2113 the TOE that are termed *SFR-enforcing*. Such portions play a direct role in implementing any SFR on
 2114 the TOE. Such SFRs refer to any functionality provided by one of the SFRs contained in the ST. It should
 2115 be noted that the definition of *plays a role in* for SFR-enforcing functionality is impossible to express
 2116 quantitatively. For example, in the implementation of a Discretionary Access Control (DAC)
 2117 mechanism, a very narrow view of *SFR-enforcing* might be the several lines of code that actually
 2118 perform the check of a subject's attributes against the object's attributes. A broader view would
 2119 include the software entity (e.g., C function) that contained the several lines of code. A broader view
 2120 still would include callers of the C function, since they would be responsible for enforcing the decision
 2121 returned by the attribute check. A still broader view would include any code in the call tree (or
 2122 programming equivalent for the implementation language used) for that C function (e.g., a sort
 2123 function that sorted access control list entries in a first-match algorithm implementation). At some

point, the component is not so much *enforcing* the security policy but rather plays a *supporting* role; such components are termed *SFR supporting*. One of the characteristics of SFR-supporting functionality is that it is trusted to preserve the correctness of the SFR implementation by operating without error. Such functionality may be depended on by SFR-enforcing functionality, but the dependence is generally at a functional level; for example, memory management, buffer management, etc. Further down on the security relevance continuum is functionality termed *SFR non-interfering*. Such functionality has no role in implementing the SFRs, and is likely part of the TSF because of its environment; for example, any code running in a privileged hardware mode on an operating system. It needs to be considered part of the TSF because, if compromised (or replaced by malicious code), it could compromise the correct operation of an SFR by virtue of its operating in the privileged hardware mode. An example of SFR non-interfering functionality might be a set of mathematical floating point operations implemented in kernel mode for speed considerations.

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

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

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

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

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

The difference between semiformal and informal documents is only a matter of formatting or presentation: a semiformal notation includes 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*.

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

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

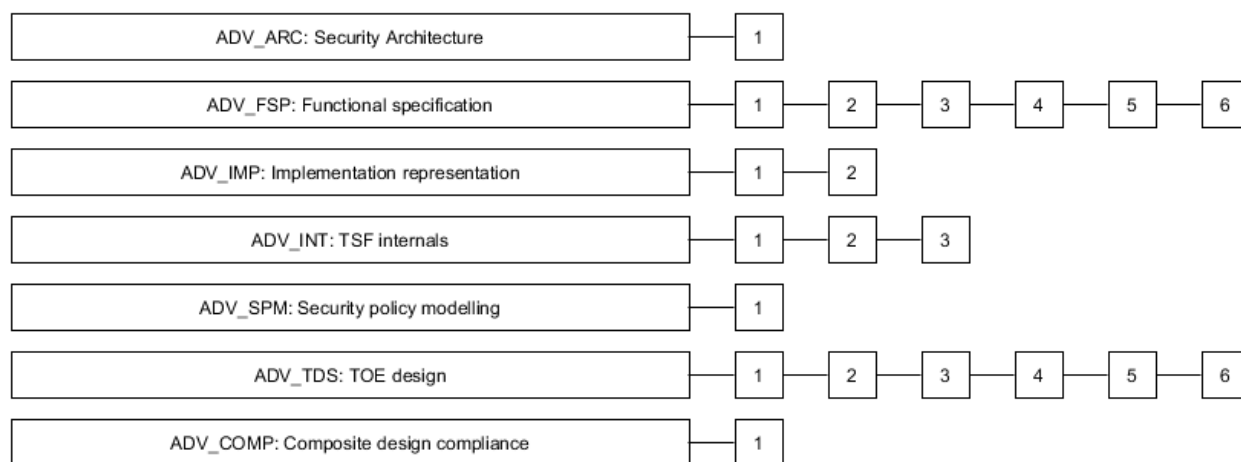


Figure 8 — ADV: Development class decomposition

In case of a **multi-assurance evaluation** the requirements for the description (at the various levels of abstraction) of the design and implementation of the SFRs (ADV_FSP, ADV_TDS, ADV_IMP and ADV_COMP) will be presented for the **sub-TSF** of the TOE. The architecture family (Security Architecture (ADV_ARC)) provides for requirements and analysis of the TOE based on properties of domain separation, self-protection, and non-bypassability which also may hold for boundaries between the **sub-TSF**.

2199 10.2 Security Architecture (ADV_ARC)

2200 10.2.1 Objectives

2201 The objective of this family is for the developer to provide a description of the security architecture of
 2202 the TSF. This will allow analysis of the information that, when coupled with the other evidence
 2203 presented for the TSF, will confirm the TSF achieves the desired properties. The security architecture
 2204 descriptions supports the implicit claim that security analysis of the TOE can be achieved by
 2205 examining the TSF; without a sound architecture, the entire TOE functionality would have to be
 2206 examined.

2207 10.2.2 Component levelling

2208 This family contains only one component.

2209 10.2.3 Application notes

2210 The properties of self-protection, domain separation, and non-bypassability are distinct from security
 2211 functionality expressed by **ISO/IEC 15408-2** SFRs because self-protection and non-bypassability
 2212 largely have no directly observable interface at the TSF. Rather, they are properties of the TSF that are
 2213 achieved through the design of the TOE and TSF, and enforced by the correct implementation of that
 2214 design.

2215 The approach used in this family is for the developer to design and provide a TSF that exhibits the
 2216 above-mentioned properties, and to provide evidence (in the form of documentation) that explains
 2217 these properties of the TSF. This explanation is provided at the same level of detail as the description
 2218 of the SFR-enforcing elements of the TOE in the TOE design document. The evaluator has the
 2219 responsibility for looking at the evidence and, coupled with other evidence delivered for the TOE and
 2220 TSF, determining that the properties are achieved.

2221 Specification of security functionality implementing the SFRs (in the Functional specification
 2222 (ADV_FSP) and TOE design (ADV_TDS)) will not necessarily describe mechanisms employed in
 2223 implementing self-protection and non-bypassability (e.g. memory management mechanisms).
 2224 Therefore, the material needed to provide the assurance that these requirements are being achieved is
 2225 better suited to a presentation separate from the design decomposition of the TSF as embodied in
 2226 ADV_FSP and ADV_TDS. This is not to imply that the security architecture description called for by this
 2227 component cannot reference or make use of the design decomposition material; but it is likely that
 2228 much of the detail present in the decomposition documentation will not be relevant to the argument
 2229 being provided for the security architecture description document.

2230 The description of architectural soundness can be thought of as a developer's vulnerability analysis, in
 2231 that it provides the justification for why the TSF is sound and enforces all of its SFRs. Where the
 2232 soundness is achieved through specific security mechanisms, these will be tested as part of the Depth
 2233 (ATE_DPT) requirements; where the soundness is achieved solely through the architecture, the
 2234 behaviour will be tested as part of the AVA: Vulnerability assessment requirements.

2235 This family consists of requirements for a security architecture description that describes the self-
 2236 protection, domain separation, non-bypassability principles, including a description of how these
 2237 principles are supported by the parts of the TOE that are used for TSF initialisation.

2238 In case of a **multi-assurance evaluation** the properties of self-protection, domain separation, and
 2239 non-bypassability may also be described for boundaries between the **sub-TSF**.

2240 Additional information on the security architecture properties of self-protection, domain separation,
 2241 and non-bypassability can be found in Annex A.1, ADV_ARC: Supplementary material on security
 2242 architectures.

2243 **10.2.4 ADV_ARC.1 Security architecture description**

2244 Dependencies: ADV_FSP.1 Basic functional specification

2245 ADV_TDS.1 Basic design

2246 **Developer action elements**

2247 **ADV_ARC.1.1D**

2248 **The developer shall design and implement the TOE so that the security features of the TSF**
 2249 **cannot be bypassed.**

2250 **ADV_ARC.1.2D**

2251 **The developer shall design and implement the TSF so that it is able to protect itself from**
 2252 **tampering by untrusted active entities.**

2253 **ADV_ARC.1.3D**

2254 **The developer shall provide a security architecture description of the TSF.**

2255 **Content and presentation elements**

2256 **ADV_ARC.1.1C**

2257 **The security architecture description shall be at a level of detail commensurate with the**
 2258 **description of the SFR-enforcing abstractions described in the TOE design document.**

2259 **ADV_ARC.1.2C**

2260 **The security architecture description shall describe the security domains maintained by the**
 2261 **TSF consistently with the SFRs.**

2262 **ADV_ARC.1.3C**

2263 **The security architecture description shall describe how the TSF initialisation process is**
 2264 **secure.**

2265 **ADV_ARC.1.4C**

2266 **The security architecture description shall demonstrate that the TSF protects itself from**
 2267 **tampering.**

2268 **ADV_ARC.1.5C**

2269 **The security architecture description shall demonstrate that the TSF prevents bypass of the**
 2270 **SFR-enforcing functionality.**

2271 **Evaluator action elements**

2272 **ADV_ARC.1.1E**

2273 **The evaluator shall confirm that the information provided meets all requirements for content**
 2274 **and presentation of evidence.**

2275 **10.3 Functional specification (ADV_FSP)**

2276 **10.3.1 Objectives**

2277 This family levies requirements upon the functional specification, which describes the TSF interfaces
 2278 (TSFIs). The TSFI consists of all means by which external entities (or subjects in the TOE but outside of
 2279 the TSF) supply data to the TSF, receive data from the TSF and invoke services from the TSF. It does
 2280 *not* describe how the TSF processes those service requests, nor does it describe the communication
 2281 when the TSF invokes services from its operational environment; this information is addressed by the
 2282 TOE design (ADV_TDS) and Reliance of dependent component (ACO_REL) families, respectively.

2283 This family provides assurance directly by allowing the evaluator to understand how the TSF meets
 2284 the claimed SFRs. It also provides assurance indirectly, as input to other assurance families and
 2285 classes:

- 2286 • ADV_ARC, where the description of the TSFIs may be used to gain better understanding of how the
 2287 TSF is protected against corruption (i.e. subversion of self-protection or domain separation)
 2288 and/or bypass;
- 2289 • ATE, where the description of the TSFIs is an important input for both developer and evaluator
 2290 testing;
- 2291 • AVA, where the description of the TSFIs is used to search for vulnerabilities.

2292 **10.3.2 Component levelling**

2293 The components in this family are levelled on the degree of detail required of the description of the
 2294 TSFIs, and the degree of formalism required of the description of the TSFIs.

2295 **10.3.3 Application notes**

2296 Once the TSFIs are determined (see A.2.2 for guidance and examples of determining TSFI), they are
 2297 described. At lower-level components, developers focus their documentation (and evaluators focus
 2298 their analysis) on the more security-relevant aspects of the TOE. Three categories of TSFIs are defined,
 2299 based upon the relevance the services available through them have to the SFRs being claimed:

- 2300 • If a service available through an interface can be traced to one of the SFRs levied on the TSF,
 2301 then that interface is termed *SFR-enforcing*. Note that it is possible that an interface may have
 2302 various services and results, some of which may be SFR-enforcing and some of which may not.
- 2303 • Interfaces to (or services available through an interface relating to) services that SFR-enforcing
 2304 functionality depends upon, but need only to function correctly in order for the security
 2305 policies of the TOE to be preserved, are termed *SFR-supporting*.
- 2306 • Interfaces to services on which SFR-enforcing functionality has no dependence are termed *SFR*
 2307 *non-interfering*.

2308 It should be noted that in order for an interface to be SFR-supporting or SFR non-interfering it must
 2309 have *no* SFR-enforcing services or results. In contrast, an SFR-enforcing interface may have SFR-
 2310 supporting services (for example, the ability to set the system clock may be an SFR-enforcing service of
 2311 an interface, but if that same interface is used to display the system date that service may be only SFR-
 2312 supporting). An example of a purely SFR-supporting interface is a system call interface that is used
 2313 both by users and by a portion of the TSF that is running on behalf of users.

2314 As more information about the TSFIs becomes available, the greater the assurance that can be gained
 2315 that the interfaces are correctly categorised/analysed. The requirements are structured such that, at
 2316 the lowest level, the information required for SFR non-interfering interfaces is the minimum necessary
 2317 in order for the evaluator to make this determination in an effective manner. At higher levels, more
 2318 information becomes available so that the evaluator has greater confidence in the designation.

2319 The purpose in defining these labels (SFR-enforcing, SFR-supporting, and SFR-non-interfering) and for
 2320 levying different requirements upon each (at the lower assurance components) is to provide a first
 2321 approximation of where to focus the analysis and the evidence upon which that analysis is performed.
 2322 If the developer's documentation of the TSF interfaces describes all of the interfaces to the degree
 2323 specified in the requirements for the SFR-enforcing interfaces (that is, if the documentation exceeds
 2324 the requirements), there is no need for the developer to create new evidence to match the
 2325 requirements. Similarly, because the labels are merely a means of differentiating the interface types
 2326 within the requirements, there is no need for the developer to update the evidence solely to label the
 2327 interfaces as SFR-enforcing, SFR-supporting, and SFR-non-interfering. The primary purpose of this
 2328 labelling is to allow developers with less mature development methodologies (and associated
 2329 artefacts, such as detailed interface and design documentation) to provide only the necessary evidence
 2330 without undue cost.

2331 The last C element of each component within this family provides a direct correspondence between
 2332 the SFRs and the functional specification; that is, an indication of which interfaces are used to invoke
 2333 each of the claimed SFRs. In the cases where the ST contains such functional requirements as 15408-2,
 2334 whose functionality may not manifest itself at the TSFIs, the functional specification and/or the tracing
 2335 is expected to identify these SFRs; including them in the functional specification helps to ensure that
 2336 they are not lost at lower levels of decomposition, where they will be relevant.

2337 **Detail about the Interfaces**

2338 The requirements define collections of details about TSFI to be provided. For the purposes of the
 2339 requirements, interfaces are specified (in varying degrees of detail) in terms of their purpose, method
 2340 of use, parameters, parameter descriptions, and error messages.

2341 The *purpose* of an interface is a high-level description of the general goal of the interface (e.g. process
 2342 GUI commands, receive network packets, provide printer output, etc.).

2343 The interface's *method of use* describes how the interface is supposed to be used. This description
 2344 should be built around the various interactions available at that interface. For instance, if the interface
 2345 were a Unix command shell, *ls*, *mv* and *cp* would be interactions for that interface. For each interaction
 2346 the method of use describes what the interaction does, both for behaviour seen at the interface (e.g.
 2347 the programmer calling the API, the Windows users changing a setting in the registry, etc.) as well as
 2348 behaviour at other interfaces (e.g. generating an audit record).

2349 *Parameters* are explicit inputs to and outputs from an interface that control the behaviour of that
 2350 interface. For example, parameters are the arguments supplied to an API; the various fields in a packet
 2351 for a given network protocol; the individual key values in the Windows Registry; the signals across a

2352 set of pins on a chip; the flags that can be set for the *ls*, etc. The parameters are “identified” with a
 2353 simple list of what they are.

2354 A *parameter description* tells what the parameter is in some meaningful way. For instance, an
 2355 acceptable parameter description for interface *foo(i)* would be “parameter *i* is an integer that indicates
 2356 the number of users currently logged in to the system”. A description such as “parameter *i* is an
 2357 integer” is not an acceptable.

2358 The description of an interface's *actions* describes what the interface does. This is more detailed than
 2359 the purpose in that, while the “purpose” reveals why one might want to use it, the “actions” reveals
 2360 everything that it does. These actions might be related to the SFRs or not. In cases where the
 2361 interface's action is not related to SFRs, its description is said to be *summarised*, meaning the
 2362 description merely makes clear that it is indeed not SFR-related.

2363 The *error message description* identifies the condition that generated it, what the message is, and the
 2364 meaning of any error codes. An error message is generated by the TSF to signify that a problem or
 2365 irregularity of some degree has been encountered. The requirements in this family refer to different
 2366 kinds of error messages:

- 2367 • a “direct” error message is a security-relevant response through a specific TSFI invocation.
- 2368 • an “indirect” error cannot be tied to a specific TSFI invocation because it results from system-wide
 2369 conditions (e.g. resource exhaustion, connectivity interruptions, etc.). Error messages that are not
 2370 security-relevant are also considered “indirect”.
- 2371 • “remaining” errors are any other errors, such as those that might be referenced within the code.
 2372 For example, the use of condition-checking code that checks for conditions that would not logically
 2373 occur (e.g. a final “else” after a list of “case” statements), would provide for generating a catch-all
 2374 error message; in an operational TOE, these error messages should never be seen.

2375 An example functional specification is provided in A.2.4.

2376 **Components of this Family**

2377 Increasing assurance through increased completeness and accuracy in the interface specification is
 2378 reflected in the documentation required from the developer as detailed in the various hierarchical
 2379 components of this family.

2380 At ADV_FSP.1 Basic functional specification, the only documentation required is a characterisation of
 2381 all TSFIs and a high level description of SFR-enforcing and SFR-supporting TSFIs. To provide some
 2382 assurance that the “important” aspects of the TSF have been correctly characterised at the TSFIs, the
 2383 developer is required to provide the purpose and method of use, parameters for the SFR-enforcing and
 2384 SFR-supporting TSFIs.

2385 At ADV_FSP.2 Security-enforcing functional specification, the developer is required to provide the
 2386 purpose, method of use, parameters, and parameter descriptions for all TSFIs. Additionally, for the
 2387 SFR-enforcing TSFIs the developer has to describe the SFR-enforcing actions and direct error
 2388 messages.

2389 At ADV_FSP.3 Functional specification with complete summary, the developer must now, in addition to
 2390 the information required at ADV_FSP.2, provide enough information about the SFR-supporting and
 2391 SFR-non-interfering actions to show that they are not SFR-enforcing. Further, the developer must now
 2392 document all of the direct error messages resulting from the invocation of SFR-enforcing TSFIs.

- 2393 At ADV_FSP.4 Complete functional specification, all TSFIs - whether SFR-enforcing, SFR-supporting or
2394 SFR-non-interfering - must be described to the same degree, including all of the direct error messages.
- 2395 At ADV_FSP.5 Complete semi-formal functional specification with additional error information, the
2396 TSFIs descriptions also include error messages that do not result from an invocation of a TSFI.
- 2397 At ADV_FSP.6 Complete semi-formal functional specification with additional formal specification, in
2398 addition to the information required by ADV_FSP.5, all remaining error messages are included. The
2399 developer must also provide a formal description of the TSFI. This provides an alternative view of the
2400 TSFI that may expose inconsistencies or incomplete specification.
- 2401 **10.3.4 ADV_FSP.1 Basic functional specification**
- 2402 Dependencies: No dependencies.
- 2403 **Developer action elements**
- 2404 **ADV_FSP.1.1D**
- 2405 **The developer shall provide a functional specification.**
- 2406 **ADV_FSP.1.2D**
- 2407 **The developer shall provide a tracing from the functional specification to the SFRs.**
- 2408 **Content and presentation elements**
- 2409 **ADV_FSP.1.1C**
- 2410 **The functional specification shall describe the purpose and method of use for each SFR-**
2411 **enforcing and SFR-supporting TSFI.**
- 2412 **ADV_FSP.1.2C**
- 2413 **The functional specification shall identify all parameters associated with each SFR-enforcing**
2414 **and SFR-supporting TSFI.**
- 2415 **ADV_FSP.1.3C**
- 2416 **The functional specification shall provide rationale for the implicit categorisation of interfaces**
2417 **as SFR-non-interfering.**
- 2418 **ADV_FSP.1.4C**
- 2419 **The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.**
- 2420 **Evaluator action elements**
- 2421 **ADV_FSP.1.1E**
- 2422 **The evaluator shall confirm that the information provided meets all requirements for content**
2423 **and presentation of evidence.**

- 2424 **ADV_FSP.1.2E**
- 2425 **The evaluator shall determine that the functional specification is an accurate and complete**
 2426 **instantiation of the SFRs.**
- 2427 **10.3.5 ADV_FSP.2 Security-enforcing functional specification**
- 2428 Dependencies: ADV_TDS.1 Basic design
- 2429 **Developer action elements**
- 2430 **ADV_FSP.2.1D**
- 2431 The developer shall provide a functional specification.
- 2432 **ADV_FSP.2.2D**
- 2433 The developer shall provide a tracing from the functional specification to the SFRs.
- 2434 **Content and presentation elements**
- 2435 **ADV_FSP.2.1C**
- 2436 **The functional specification shall completely represent the TSF.**
- 2437 **ADV_FSP.2.2C**
- 2438 The functional specification shall describe the purpose and method of use for **all** TSFI.
- 2439 **ADV_FSP.2.3C**
- 2440 The functional specification shall identify **and describe** all parameters associated with each TSFI.
- 2441 **ADV_FSP.2.4C**
- 2442 **For each SFR-enforcing TSFI, the functional specification shall describe the SFR-enforcing actions**
 2443 **associated with the TSFI.**
- 2444 **ADV_FSP.2.5C**
- 2445 **For each SFR-enforcing TSFI, the functional specification shall describe direct error messages**
 2446 **resulting from processing associated with the SFR-enforcing actions.**
- 2447 **ADV_FSP.2.6C**
- 2448 The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.
- 2449 **Evaluator action elements**
- 2450 **ADV_FSP.2.1E**
- 2451 The evaluator shall confirm that the information provided meets all requirements for content and
 2452 presentation of evidence.

2453 **ADV_FSP.2.2E**

2454 The evaluator shall determine that the functional specification is an accurate and complete
 2455 instantiation of the SFRs.

2456 **10.3.6 ADV_FSP.3 Functional specification with complete summary**

2457 Dependencies: ADV_TDS.1 Basic design

2458 **Developer action elements**2459 **ADV_FSP.3.1D**

2460 The developer shall provide a functional specification.

2461 **ADV_FSP.3.2D**

2462 The developer shall provide a tracing from the functional specification to the SFRs.

2463 **Content and presentation elements**2464 **ADV_FSP.3.1C**

2465 The functional specification shall completely represent the TSF.

2466 **ADV_FSP.3.2C**

2467 The functional specification shall describe the purpose and method of use for all TSFI.

2468 **ADV_FSP.3.3C**

2469 The functional specification shall identify and describe all parameters associated with each TSFI.

2470 **ADV_FSP.3.4C**

2471 For each SFR-enforcing TSFI, the functional specification shall describe the SFR-enforcing actions
 2472 associated with the TSFI.

2473 **ADV_FSP.3.5C**

2474 For each SFR-enforcing TSFI, the functional specification shall describe direct error messages resulting
 2475 from **SFR-enforcing actions and exceptions** associated with **invocation of the TSFI**.

2476 **ADV_FSP.3.6C**

2477 **The functional specification shall summarise the SFR-supporting and SFR-non-interfering**
 2478 **actions associated with each TSFI.**

2479 **ADV_FSP.3.7C**

2480 The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

2481 **Evaluator action elements**

2482 **ADV_FSP.3.1E**

2483 The evaluator shall confirm that the information provided meets all requirements for content and
2484 presentation of evidence.

2485 **ADV_FSP.3.2E**

2486 The evaluator shall determine that the functional specification is an accurate and complete
2487 instantiation of the SFRs.

2488 **10.3.7 ADV_FSP.4 Complete functional specification**

2489 Dependencies: ADV_TDS.1 Basic design

2490 **Developer action elements**

2491 **ADV_FSP.4.1D**

2492 The developer shall provide a functional specification.

2493 **ADV_FSP.4.2D**

2494 The developer shall provide a tracing from the functional specification to the SFRs.

2495 **Content and presentation elements**

2496 **ADV_FSP.4.1C**

2497 The functional specification shall completely represent the TSF.

2498 **ADV_FSP.4.2C**

2499 The functional specification shall describe the purpose and method of use for all TSFI.

2500 **ADV_FSP.4.3C**

2501 The functional specification shall identify and describe all parameters associated with each TSFI.

2502 **ADV_FSP.4.4C**

2503 The functional specification shall **describe all** actions associated with each TSFI.

2504 **ADV_FSP.4.5C**

2505 **The functional specification shall describe all direct error messages that may result from an**
2506 **invocation of each TSFI.**

2507 **ADV_FSP.4.6C**

2508 The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

2509 **Evaluator action elements**

2510 **ADV_FSP.4.1E**

2511 The evaluator shall confirm that the information provided meets all requirements for content and
2512 presentation of evidence.

2513 **ADV_FSP.4.2E**

2514 The evaluator shall determine that the functional specification is an accurate and complete
2515 instantiation of the SFRs.

2516 **10.3.8 ADV_FSP.5 Complete semi-formal functional specification with additional error**
2517 **information**

2518 Dependencies: ADV_TDS.1 Basic design

2519 ADV_IMP.1 Implementation representation of the TSF

2520 **Developer action elements**

2521 **ADV_FSP.5.1D**

2522 The developer shall provide a functional specification.

2523 **ADV_FSP.5.2D**

2524 The developer shall provide a tracing from the functional specification to the SFRs.

2525 **Content and presentation elements**

2526 **ADV_FSP.5.1C**

2527 The functional specification shall completely represent the TSF.

2528 **ADV_FSP.5.2C**

2529 **The functional specification shall describe the TSFI using a semi-formal style.**

2530 **ADV_FSP.5.3C**

2531 The functional specification shall describe the purpose and method of use for all TSFI.

2532 **ADV_FSP.5.4C**

2533 The functional specification shall identify and describe all parameters associated with each TSFI.

2534 **ADV_FSP.5.5C**

2535 The functional specification shall describe all actions associated with each TSFI.

2536 **ADV_FSP.5.6C**

2537 The functional specification shall describe all direct error messages that may result from an invocation
2538 of each TSFI.

2539 **ADV_FSP.5.7C**

2540 **The functional specification shall describe all error messages that do not result from an**
 2541 **invocation of a TSFI.**

2542 **ADV_FSP.5.8C**

2543 **The functional specification shall provide a rationale for each error message contained in the**
 2544 **TSF implementation yet does not result from an invocation of a TSFI.**

2545 **ADV_FSP.5.9C**

2546 The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

2547 **Evaluator action elements**2548 **ADV_FSP.5.1E**

2549 The evaluator shall confirm that the information provided meets all requirements for content and
 2550 presentation of evidence.

2551 **ADV_FSP.5.2E**

2552 The evaluator shall determine that the functional specification is an accurate and complete
 2553 instantiation of the SFRs.

2554 **10.3.9 ADV_FSP.6 Complete semi-formal functional specification with additional formal**
 2555 **specification**

2556 Dependencies: ADV_TDS.1 Basic design

2557 ADV_IMP.1 Implementation representation of the TSF

2558 **Developer action elements**2559 **ADV_FSP.6.1D**

2560 The developer shall provide a functional specification.

2561 **ADV_FSP.6.2D**

2562 **The developer shall provide a formal presentation of the functional specification of the TSF.**

2563 **ADV_FSP.6.3D**

2564 The developer shall provide a tracing from the functional specification to the SFRs.

2565 **Content and presentation elements**2566 **ADV_FSP.6.1C**

2567 The functional specification shall completely represent the TSF.

2568 **ADV_FSP.6.2C**

2569 The functional specification shall describe the TSFI using a **formal** style.

2570 **ADV_FSP.6.3C**

2571 The functional specification shall describe the purpose and method of use for all TSFI.

2572 **ADV_FSP.6.4C**

2573 The functional specification shall identify and describe all parameters associated with each TSFI.

2574 **ADV_FSP.6.5C**

2575 The functional specification shall describe all actions associated with each TSFI.

2576 **ADV_FSP.6.6C**

2577 The functional specification shall describe all direct error messages that may result from an invocation
2578 of each TSFI.

2579 **ADV_FSP.6.7C**

2580 The functional specification shall describe all error messages **contained in the TSF implementation**
2581 **representation.**

2582 **ADV_FSP.6.8C**

2583 The functional specification shall provide a rationale for each error message contained in the TSF
2584 implementation **that is not otherwise described in the functional specification justifying why it is**
2585 **not associated with** a TSFI.

2586 **ADV_FSP.6.9C**

2587 **The formal presentation of the functional specification of the TSF shall describe the TSFI using**
2588 **a formal style, supported by informal, explanatory text where appropriate.**

2589 **ADV_FSP.6.10C**

2590 The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

2591 **Evaluator action elements**2592 **ADV_FSP.6.1E**

2593 The evaluator shall confirm that the information provided meets all requirements for content and
2594 presentation of evidence.

2595 **ADV_FSP.6.2E**

2596 The evaluator shall determine that the functional specification is an accurate and complete
2597 instantiation of the SFRs.

2598 **10.4 Implementation representation (ADV_IMP)**2599 **10.4.1 Objectives**

2600 The function of the Implementation representation (ADV_IMP) family is for the developer to make
2601 available the implementation representation (and, at higher levels, the implementation itself) of the

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

10.4.2 Component levelling

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

10.4.3 Application notes

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

The entire implementation representation is made available to ensure that analysis activities are not curtailed due to lack of information. This does not, however, imply that all of the representation is examined when the analysis activities are being performed. This is likely impractical in almost all cases, in addition to the fact that it most likely will not result in a higher-assurance TOE vs. targeted sampling of the implementation representation. The implementation representation is made available to allow analysis of other TOE design decompositions (e.g., functional specification, TOE design), and to gain confidence that the security functionality described at a higher level in the design actually appear to be implemented in the TOE. Conventions in some forms of the implementation representation may make it difficult or impossible to determine from just the implementation representation itself what the actual result of the compilation or run-time interpretation will be. For example, compiler directives for C language compilers will cause the compiler to exclude or include entire portions of the code. For this reason, it is important that such "extra" information or related tools (scripts, compilers, etc.) be provided so that the implementation representation can be accurately determined.

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

The implementation representation is manipulated by the developer in a form that is suitable for transformation to the actual implementation. For instance, the developer may work with files containing source code, which is eventually compiled to become part of the TSF. The developer makes available the implementation representation in the form used by the developer, so that the evaluator may use automated techniques in the analysis. This also increases the confidence that the implementation representation examined is actually the one used in the production of the TSF (as opposed to the case where it is supplied in an alternate presentation format, such as a word processor

document). It should be noted that other forms of the implementation representation may also be used by the developer; these forms are supplied as well. The overall goal is to supply the evaluator with the information that will maximise the effectiveness of the evaluator's analysis efforts.

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

10.4.4 ADV_IMP.1 Implementation representation of the TSF

Dependencies: ADV_TDS.3 Basic modular design

ALC_TAT.1 Well-defined development tools

Developer action elements

ADV_IMP.1.1D

The developer shall make available the implementation representation for the entire TSF.

ADV_IMP.1.2D

The developer shall provide a mapping between the TOE design description and the sample of the implementation representation.

Content and presentation elements

ADV_IMP.1.1C

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

ADV_IMP.1.2C

The implementation representation shall be in the form used by the development personnel.

ADV_IMP.1.3C

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

2681 **Evaluator action elements**

2682 **ADV_IMP.1.1E**

2683 **The evaluator shall confirm that, for the selected sample of the implementation representation,**
 2684 **the information provided meets all requirements for content and presentation of evidence.**

2685 **10.4.5 ADV_IMP.2 Complete mapping of the implementation representation of the TSF**

2686 Dependencies: ADV_TDS.3 Basic modular design

2687 ALC_TAT.1 Well-defined development tools

2688 ALC_CMC.5 Advanced support

2689 **Developer action elements**

2690 **ADV_IMP.2.1D**

2691 The developer shall make available the implementation representation for the entire TSF.

2692 **ADV_IMP.2.2D**

2693 The developer shall provide a mapping between the TOE design description and the **entire**
 2694 implementation representation.

2695 **Content and presentation elements**

2696 **ADV_IMP.2.1C**

2697 The implementation representation shall define the TSF to a level of detail such that the TSF may be
 2698 generated without further design decisions.

2699 **ADV_IMP.2.2C**

2700 The implementation representation shall be in the form used by the development personnel.

2701 **ADV_IMP.2.3C**

2702 The mapping between the TOE design description and the **entire** implementation representation shall
 2703 demonstrate their correspondence.

2704 **Evaluator action elements**

2705 **ADV_IMP.2.1E**

2706 The evaluator shall confirm that the information provided meets all requirements for content and
 2707 presentation of evidence.

2708 **10.5 TSF internals (ADV_INT)**

2709 **10.5.1 Objectives**

2710 This family addresses the assessment of the internal structure of the TSF. A TSF whose internals are
 2711 well-structured is easier to implement and less likely to contain flaws that could lead to
 2712 vulnerabilities; it is also easier to maintain without the introduction of flaws.

2713 10.5.2 Component levelling

2714 The components in this family are levelled on the basis of the amount of structure and minimisation of
 2715 complexity required. ADV_INT.1 Well-structured subset of TSF internals places requirements for well-
 2716 structured internals on only selected parts of the TSF. This component is not included in an EAL
 2717 because this component is viewed for use in special circumstances (e.g., the sponsor has a specific
 2718 concern regarding a cryptographic module, which is isolated from the rest of the TSF) and would not
 2719 be widely applicable.

2720 At the next level, the requirements for well-structured internals are placed on the entire TSF. Finally,
 2721 minimisation of complexity is introduced in the highest component.

2722 10.5.3 Application notes

2723 These requirements, when applied to the internal structure of the TSF, typically result in
 2724 improvements that aid both the developer and the evaluator in understanding the TSF, and also
 2725 provide the basis for designing and evaluating test suites. Further, improving understandability of the
 2726 TSF should assist the developer in simplifying its maintainability.

2727 The requirements in this family are presented at a fairly abstract level. The wide variety of TOEs
 2728 makes it impossible to codify anything more specific than “well-structured” or “minimum complexity”.
 2729 Judgements on structure and complexity are expected to be derived from the specific technologies
 2730 used in the TOE. For example, software is likely to be considered well-structured if it exhibits the
 2731 characteristics cited in the software engineering disciplines. The components within this family call for
 2732 identifying the standards for measuring the characteristic of being well-structured and not overly-
 2733 complex.

2734 10.5.4 ADV_INT.1 Well-structured subset of TSF internals

2735 Dependencies: ADV_IMP.1 Implementation representation of the TSF

2736 ADV_TDS.3 Basic modular design

2737 ALC_TAT.1 Well-defined development tools

2738 Objectives

2739 The objective of this component is to provide a means for requiring specific portions of the TSF to be
 2740 well-structured. The intent is that the entire TSF has been designed and implemented using sound
 2741 engineering principles, but the analysis is performed upon only a specific subset.

2742 Application notes

2743 This component requires the PP or ST author to fill in an assignment with the subset of the TSF. This
 2744 subset may be identified in terms of the internals of the TSF at any layer of abstraction. For example:

2745 a) the structural elements of the TSF as identified in the TOE design (e.g. “The developer shall design
 2746 and implement *the audit subsystem* such that it has well-structured internals.”)

2747 b) the implementation (e.g. “The developer shall design and implement *the encrypt.c and decrypt.c*
 2748 *files* such that it has well-structured internals.” or “The developer shall design and implement *the*
 2749 *6227 IC chip* such that it has well-structured internals.”)

2750 It is likely this would not be readily accomplished by referencing the claimed SFRs (e.g. “The developer
2751 shall design and implement *the portion of the TSF that provide anonymity as defined in FPR_ANO.2* such
2752 that it has well-structured internals.”) because this does not indicate where to focus the analysis.

2753 This component has limited value and would be suitable in cases where potentially-malicious
2754 users/subjects have limited or strictly controlled access to the TSFIs or where there is another means
2755 of protection (e.g., domain separation) that ensures the chosen subset of the TSF cannot be adversely
2756 affected by the rest of the TSF (e.g., the cryptographic functionality, which is isolated from the rest of
2757 the TSF, is well-structured).

2758 **Developer action elements**

2759 **ADV_INT.1.1D**

2760 **The developer shall design and implement [assignment: *subset of the TSF*] such that it has well-**
2761 **structured internals.**

2762 **ADV_INT.1.2D**

2763 **The developer shall provide an internals description and justification.**

2764 **Content and presentation elements**

2765 **ADV_INT.1.1C**

2766 **The justification shall explain the characteristics used to judge the meaning of “well-**
2767 **structured”.**

2768 **ADV_INT.1.2C**

2769 **The TSF internals description shall demonstrate that the assigned subset of the TSF is well-**
2770 **structured.**

2771 **Evaluator action elements**

2772 **ADV_INT.1.1E**

2773 **The evaluator shall confirm that the information provided meets all requirements for content**
2774 **and presentation of evidence.**

2775 **ADV_INT.1.2E**

2776 **The evaluator shall perform an internals analysis on the assigned subset of the TSF.**

2777 **10.5.5 ADV_INT.2 Well-structured internals**

2778 Dependencies: ADV_IMP.1 Implementation representation of the TSF

2779 ADV_TDS.3 Basic modular design

2780 ALC_TAT.1 Well-defined development tools

2781 **Objectives**

2782 The objective of this component is to provide a means for requiring the TSF to be well-structured. The
2783 intent is that the entire TSF has been designed and implemented using sound engineering principles.

2784 **Application notes**

2785 Judgements on the adequacy of the structure are expected to be derived from the specific technologies
2786 used in the TOE. This component calls for identifying the standards for measuring the characteristic of
2787 being well-structured.

2788 **Developer action elements**

2789 **ADV_INT.2.1D**

2790 The developer shall design and implement the **entire TSF** such that it has well-structured internals.

2791 **ADV_INT.2.2D**

2792 The developer shall provide an internals description and justification.

2793 **Content and presentation elements**

2794 **ADV_INT.2.1C**

2795 The justification shall **describe** the characteristics used to judge the meaning of “well-structured”.

2796 **ADV_INT.2.2C**

2797 The TSF internals description shall demonstrate that the **entire TSF** is well-structured.

2798 **Evaluator action elements**

2799 **ADV_INT.2.1E**

2800 The evaluator shall confirm that the information provided meets all requirements for content and
2801 presentation of evidence.

2802 **ADV_INT.2.2E**

2803 The evaluator shall perform an internals analysis on the TSF.

2804 **10.5.6 ADV_INT.3 Minimally complex internals**

2805 Dependencies: ADV_IMP.1 Implementation representation of the TSF

2806 ADV_TDS.3 Basic modular design

2807 ALC_TAT.1 Well-defined development tools

2808 **Objectives**

2809 The objective of this component is to provide a means for requiring the TSF to be well-structured and
2810 of minimal complexity. The intent is that the entire TSF has been designed and implemented using
2811 sound engineering principles.

2812 **Application notes**

2813 Judgements on the adequacy of the structure and complexity are expected to be derived from the
 2814 specific technologies used in the TOE. This component calls for identifying the standards for
 2815 measuring the structure and complexity.

2816 **Developer action elements**

2817 **ADV_INT.3.1D**

2818 The developer shall design and implement the entire TSF such that it has well-structured internals.

2819 **ADV_INT.3.2D**

2820 The developer shall provide an internals description and justification.

2821 **Content and presentation elements**

2822 **ADV_INT.3.1C**

2823 The justification shall describe the characteristics used to judge the meaning of “well-structured” **and**
 2824 “**complex**”.

2825 **ADV_INT.3.2C**

2826 The TSF internals description shall demonstrate that the entire TSF is well-structured **and is not**
 2827 **overly complex**.

2828 **Evaluator action elements**

2829 **ADV_INT.3.1E**

2830 The evaluator shall confirm that the information provided meets all requirements for content and
 2831 presentation of evidence.

2832 **ADV_INT.3.2E**

2833 The evaluator shall perform an internals analysis on the **entire** TSF.

2834 **10.6 Security policy modelling (ADV_SPM)**

2835 **10.6.1 Objectives**

2836 It is the objective of this family to provide additional assurance from the development of a formal
 2837 *security policy model* of the TSFI behaviour of the TSF, and establishing a correspondence between the
 2838 functional specification and this security policy model. Preserving internal consistency the security
 2839 policy model is expected to formally establish the security principles from its characteristics by means
 2840 of a mathematical proof.

2841 **10.6.2 Component levelling**

2842 This family contains only one component.

2843 10.6.3 Application notes

2844 Inadequacies in a TOE can result either from a failure in understanding the security requirements or
 2845 from a flawed implementation of those security requirements. Defining the security requirements
 2846 adequately to ensure their understanding may be problematic because the definition must be
 2847 sufficiently precise to prevent undesired results or subtle flaws during implementation of the TOE.
 2848 Throughout the design, implementation, and review processes, the modelled security requirements
 2849 may and should be used as precise design and implementation guidance, thereby providing increased
 2850 assurance that the modelled security requirements modelled via the TSFI behaviour are satisfied by
 2851 the TOE. The precision of the model and resulting guidance is significantly improved by casting the
 2852 model in a formal language and verifying the security requirements by automated formal proof
 2853 techniques.

2854 The creation of a formal security policy model helps to identify and eliminate ambiguous, inconsistent,
 2855 incomplete, contradictory, or unenforceable security policy elements. Once the TOE has been built, the
 2856 formal model serves the evaluation effort by contributing to the evaluator's judgement of how well the
 2857 developer has understood the security functionality being implemented and whether there are
 2858 inconsistencies between the security requirements and the TOE design. The confidence in the model is
 2859 accompanied by a correspondence analysis for model elements and the functional specification, and a
 2860 proof that the model contains no inconsistencies.

2861 A formal security policy model is a precise formal presentation of the important aspects of security
 2862 and their relationship to the behaviour of the TOE; it identifies the set of rules and practises that
 2863 regulates how the TSF manages, protects, and otherwise controls the system resources. The model
 2864 includes the set of restrictions and properties that specify how information and computing resources
 2865 are prevented from being used to violate the SFRs, accompanied by a persuasive set of engineering
 2866 arguments showing that these restrictions and properties play a key role in the enforcement of the
 2867 SFRs. It consists both of the formalisms that express the security functionality, as well as ancillary text
 2868 to explain the model and to provide it with context. The security behaviour of the TSF is modelled both
 2869 in terms of external behaviour (i.e. how the TSF interacts with the rest of the TOE and with its
 2870 operational environment), as well as its internal behaviour.

2871 The security policy model of the TOE is informally abstracted from its realisation by considering the
 2872 TSFI behaviour defined in the functional specification, which is strongly connected to the SFRs and
 2873 security policies expressed in the ST. The purpose of formal methods lies within the enhancement of
 2874 the rigour of enforcement. Informal arguments are always prone to fallacies; especially if relationships
 2875 among subjects, objects and operations get more and more involved. In order to minimise the risk of
 2876 insecure state reachability the rules and characteristics of the security policy model are mapped to
 2877 respective properties and features within some formal system, whose rigour and strength can
 2878 afterwards be used to obtain the security properties by means of theorems and formal proof.

2879 While the term "formal security policy model" is used in academic circles, ISO/IEC 15408's approach
 2880 has no fixed definition of "security"; it would equate to whatever SFRs are being claimed. Therefore,
 2881 the formal security policy model is merely a formal representation of the set of SFRs being claimed by
 2882 the TOE.

2883 The term *security policy* has traditionally been associated with only access control policies, whether
 2884 label-based (mandatory access control) or user-based (discretionary access control). However, a
 2885 security policy is not limited to access control; there are also audit policies, identification policies,
 2886 authentication policies, encryption policies, management policies, and any other security policies that
 2887 are enforced by the TOE, as described in the PP/ST.

2888 **10.6.4 ADV_SPM.1 Formal TOE security policy model**

2889 Dependencies: ADV_FSP.5 Complete semi-formal functional specification with additional error
2890 information

2891 ADV_FSP.6 Complete semi-formal functional specification with additional formal
2892 specification

2893 **Developer action elements**

2894 **ADV_SPM.1.1D**

2895 **The developer shall provide a formal security policy model for the TSFI behaviour of the TOE.**

2896 **ADV_SPM.1.2D**

2897 **The developer shall determine all TSFIs and analyze for each TSFI whether its behaviour can be**
2898 **modelled by the formal security policy model. If a TSFI cannot be modelled, for example caused**
2899 **by technical limitations, the developer shall analyze the impact of not modelling the TSFI**
2900 **behaviour on the security of the TOE. If parts of the TSFI behaviour cannot be modelled due to**
2901 **technical limitations, the remaining parts shall nevertheless be covered by the formal model.**

2902 **ADV_SPM.1.3D**

2903 **The formal security policy model shall identify the modelled TSFIs. For each TSFI covered by**
2904 **the formal security policy model, the model shall identify the related SFRs and security policies**
2905 **in the ST. For each SFR covered by the formal security policy model, the model shall identify the**
2906 **relevant portions of the statement of SFRs.**

2907 **ADV_SPM.1.4D**

2908 **For all TSFIs that are not modelled by the formal security policy model, the developer shall**
2909 **identify the affected SFRs and security policies in the ST.**

2910 **ADV_SPM.1.5D**

2911 **The developer shall provide a formal proof of correspondence between the model and any**
2912 **formal functional specification. The proof of correspondence shall relate model elements and**
2913 **TSFIs. With ADV_SPM.1.3D, the given proof of correspondence thereby implicitly provides a**
2914 **correspondence between model elements and SFRs, as well as model elements and security**
2915 **policies. The developer defines a structured process for identifying and presenting**
2916 **corresponding items formally.**

2917 **ADV_SPM.1.6D**

2918 **The developer shall provide a demonstration of correspondence between the model and the**
2919 **functional specification. This item shall demonstrate the correspondence between model**
2920 **elements and TSFIs.**

2921 **Content and presentation elements**

2922 **ADV_SPM.1.1C**

2923 **The model shall define security for the TOE and provide a formal proof that the TOE cannot**
2924 **reach a state that is not secure.**

2925 **ADV_SPM.1.2C**

2926 **The developer shall provide an analysis why the chosen modelling formalism is appropriate.**

2927 **ADV_SPM.1.3C**

2928 **If tool support is used, the developer shall identify the tool chain used to verify the formal**
 2929 **security policy model, including environments and version numbers. The developer shall**
 2930 **provide arguments why the tool chain is suited and trustworthy.**

2931 **ADV_SPM.1.4C**

2932 **The developer shall define how the formal analysis of the formal security policy model may be**
 2933 **reproduced (for example, applying an interactive theorem prover to prove correctness of the**
 2934 **formal security policy model).**

2935 **ADV_SPM.1.5C**

2936 **The model shall be in a formal style, supported by explanatory text as required, and identify**
 2937 **the TSFIs that are modelled. Additionally, the SFRs and security policies of the TSF that are**
 2938 **modelled via the TSFI behaviour shall be presented. The model shall identify all TSFIs that are**
 2939 **not modelled (compare ADV_SPM.1.2D) and present the affected SFRs and security policies. The**
 2940 **model shall explain the reason for not modelling TSFIs and provide an impact analysis which**
 2941 **shows that correctness of the formal model is not affected.**

2942 **ADV_SPM.1.6C**

2943 **The correspondence between the model and the functional specification shall be at the correct**
 2944 **level of formality. The developer shall describe the correspondence analysis process and define**
 2945 **the applied understanding of correspondence. If a semi-formal functional specification is**
 2946 **provided, the correspondence must be shown semi-formally. If a formal functional**
 2947 **specification is provided, the correspondence must be shown formally.**

2948 **ADV_SPM.1.7C**

2949 **The correspondence shall show that the model is consistent and complete with respect to the**
 2950 **functional specification.**

2951 **Evaluator action elements**

2952 **ADV_SPM.1.1E**

2953 **The evaluator shall confirm that the information provided meets all requirements for content**
 2954 **and presentation of evidence.**

2955 **10.7 TOE design (ADV_TDS)**

2956 **10.7.1 Objectives**

2957 **The design description of a TOE provides both context for a description of the TSF, and a thorough**
 2958 **description of the TSF. As assurance needs increase, the level of detail provided in the description also**
 2959 **increases. As the size and complexity of the TSF increase, multiple levels of decomposition are**
 2960 **appropriate. The design requirements are intended to provide information (commensurate with the**
 2961 **given assurance level) so that a determination can be made that the security functional requirements**
 2962 **are realised.**

2963 10.7.2 Component levelling

2964 The components in this family are levelled on the basis of the amount of information that is required
 2965 to be presented with respect to the TSF, and on the degree of formalism required of the design
 2966 description.

2967 10.7.3 Application notes

2968 The goal of design documentation is to provide sufficient information to determine the TSF boundary,
 2969 and to describe *how* the TSF implements the Security Functional Requirements. The amount and
 2970 structure of the design documentation will depend on the complexity of the TOE and the number of
 2971 SFRs; in general, a very complex TOE with a large number of SFRs will require more design
 2972 documentation than a very simple TOE implementing only a few SFRs. Very complex TOEs will benefit
 2973 (in terms of the assurance provided) from the production of differing levels of decomposition in
 2974 describing the design, while very simple TOEs do not require both high-level and low-level
 2975 descriptions of its implementation.

2976 This family uses two levels of decomposition: the *subsystem* and the *module*. A module is the most
 2977 specific description of functionality: it is a description of the implementation. A developer should be
 2978 able to implement the part of the TOE described by the module with no further design decisions. A
 2979 subsystem is a description of the design of the TOE; it helps to provide a high-level description of what
 2980 a portion of the TOE is doing and how. As such, a subsystem may be further divided into lower-level
 2981 subsystems, or into modules. Very complex TOEs might require several levels of subsystems in order
 2982 to adequately convey a useful description of how the TOE works. Very simple TOEs, in contrast, might
 2983 not require a subsystem level of description; the module might clearly describe how the TOE works.

2984 The general approach adopted for design documentation is that, as the level of assurance increases,
 2985 the emphasis of description shifts from the general (subsystem level) to more (module level) detail. In
 2986 cases where a module-level of abstraction is appropriate because the TOE is simple enough to be
 2987 described at the module level, yet the level of assurance calls for a subsystem level of description, the
 2988 module-level description alone will suffice. For complex TOEs, however, this is not the case: an
 2989 enormous amount of (module-level) detail would be incomprehensible without an accompanying
 2990 subsystem level of description.

2991 This approach follows the general paradigm that providing additional detail about the implementation
 2992 of the TSF will result in greater assurance that the SFRs are implemented correctly, and provide
 2993 information that can be used to demonstrate this in testing (ATE: Tests).

2994 In the requirements for this family, the term *interface* is used as the means of communication
 2995 (between two subsystems or modules). It describes how the communication is invoked; this is similar
 2996 to the details of TSFI (see Functional specification (ADV_FSP)). The term *interaction* is used to identify
 2997 the purpose for communication; it identifies why two subsystems or modules are communicating.

2998 Detail about the Subsystems and Modules

2999 The requirements define collections of details about subsystems and modules to be provided:

- 3000 a) The subsystems and modules are *identified* with a simple list of what they are.
- 3001 b) Subsystems and modules may be *categorised* (either implicitly or explicitly) as “SFR-enforcing”,
 3002 “SFR-supporting”, or “SFR-non-interfering”; these terms are used the same as they are used in
 3003 Functional specification (ADV_FSP).

- 3004 c) A subsystem's *behaviour* is what it does. The behaviour may also be categorised as SFR-enforcing,
 3005 SFR-supporting, or SFR-non-interfering. The behaviour of the subsystem is never categorised as
 3006 more SFR-relevant than the category of the subsystem itself. For example, an SFR-enforcing
 3007 subsystem can have SFR-enforcing behaviour as well as SFR-supporting or SFR-non-interfering
 3008 behaviour.
- 3009 d) A *behaviour summary* of a subsystem is an overview of the actions it performs (e.g. "The TCP
 3010 subsystem assembles IP datagrams into reliable byte streams").
- 3011 e) A *behaviour description* of a subsystem is an explanation of everything it does. This description
 3012 should be at a level of detail that one can readily determine whether the behaviour has any
 3013 relevance to the enforcement of the SFRs.
- 3014 f) A *description of interactions* among or between subsystems or modules identifies the reason that
 3015 subsystems or modules communicate, and characterises the information that is passed. It need not
 3016 define the information to the same level of detail as an interface specification. For example, it
 3017 would be sufficient to say "subsystem X requests a block of memory from the memory manager,
 3018 which responds with the location of the allocated memory.
- 3019 g) A *description of interfaces* provides the details of how the interactions among modules are
 3020 achieved. Rather than describing the reason the modules are communicating or the purpose of
 3021 their communication (that is, the description of interactions), the description of interfaces
 3022 describes the details of how that communication is accomplished, in terms of the structure and
 3023 contents of the messages, semaphores, internal process communications, etc.
- 3024 h) The *purpose* describes how a module provides their functionality. It provides sufficient detail that
 3025 no further design decisions are needed. The correspondence between the implementation
 3026 representation that implements the module, and the purpose of the module should be readily
 3027 apparent.
- 3028 i) A module is otherwise *described* in terms of whatever is identified in the element.
- 3029 Subsystems and modules, and "SFR-enforcing", etc. are all further explained in greater detail in A.4,
 3030 ADV_TDS: Subsystems and Modules.

3031 **10.7.4 ADV_TDS.1 Basic design**

3032 Dependencies: ADV_FSP.2 Security-enforcing functional specification

3033 **Developer action elements**

3034 **ADV_TDS.1.1D**

3035 **The developer shall provide the design of the TOE.**

3036 **ADV_TDS.1.2D**

3037 **The developer shall provide a mapping from the TSFI of the functional specification to the**
 3038 **lowest level of decomposition available in the TOE design.**

3039 **Content and presentation elements**

3040 **ADV_TDS.1.1C**

3041 **The design shall describe the structure of the TOE in terms of subsystems.**

3042 **ADV_TDS.1.2C**

3043 **The design shall identify all subsystems of the TSF.**

3044 **ADV_TDS.1.3C**

3045 **The design shall provide the behaviour summary of each SFR-supporting or SFR-non-**
 3046 **interfering TSF subsystem.**

3047 **ADV_TDS.1.4C**

3048 **The design shall summarise the SFR-enforcing behaviour of the SFR-enforcing subsystems.**

3049 **ADV_TDS.1.5C**

3050 **The design shall provide a description of the interactions among SFR-enforcing subsystems of**
 3051 **the TSF, and between the SFR-enforcing subsystems of the TSF and other subsystems of the TSF.**

3052 **ADV_TDS.1.6C**

3053 **The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE**
 3054 **design that they invoke.**

3055 **Evaluator action elements**

3056 **ADV_TDS.1.1E**

3057 **The evaluator shall confirm that the information provided meets all requirements for content**
 3058 **and presentation of evidence.**

3059 **ADV_TDS.1.2E**

3060 **The evaluator shall determine that the design is an accurate and complete instantiation of all**
 3061 **security functional requirements.**

3062 **10.7.5 ADV_TDS.2 Architectural design**

3063 Dependencies: ADV_FSP.3 Functional specification with complete summary

3064 **Developer action elements**

3065 **ADV_TDS.2.1D**

3066 **The developer shall provide the design of the TOE.**

3067 **ADV_TDS.2.2D**

3068 **The developer shall provide a mapping from the TSFI of the functional specification to the lowest level**
 3069 **of decomposition available in the TOE design.**

3070 **Content and presentation elements**

3071 **ADV_TDS.2.1C**

3072 The design shall describe the structure of the TOE in terms of subsystems.

3073 **ADV_TDS.2.2C**

3074 The design shall identify all subsystems of the TSF.

3075 **ADV_TDS.2.3C**

3076 **The design shall provide the behaviour summary of each SFR non-interfering subsystem of the**
3077 **TSF.**

3078 **ADV_TDS.2.4C**

3079 The design shall **describe** the SFR-enforcing behaviour of the SFR-enforcing subsystems.

3080 **ADV_TDS.2.5C**

3081 The design shall summarise the **SFR-supporting and SFR-non-interfering** behaviour of the SFR-
3082 enforcing subsystems.

3083 **10.7.5.1.1 ADV_TDS.2.6C**

3084 The design shall summarise the behaviour of the **SFR-supporting** subsystems.

3085 **ADV_TDS.2.7C**

3086 **The design shall provide a description of the interactions among all subsystems of the TSF.**

3087 **ADV_TDS.2.8C**

3088 The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE design that
3089 they invoke.

3090 **Evaluator action elements**

3091 **ADV_TDS.2.1E**

3092 The evaluator shall confirm that the information provided meets all requirements for content and
3093 presentation of evidence.

3094 **ADV_TDS.2.2E**

3095 The evaluator shall determine that the design is an accurate and complete instantiation of all security
3096 functional requirements.

3097 **10.7.6 ADV_TDS.3 Basic modular design**

3098 Dependencies: ADV_FSP.4 Complete functional specification

3099 **Developer action elements**

3100 **ADV_TDS.3.1D**

3101 The developer shall provide the design of the TOE.

3102 **ADV_TDS.3.2D**

3103 The developer shall provide a mapping from the TSFI of the functional specification to the lowest level
3104 of decomposition available in the TOE design.

3105 **Content and presentation elements**

3106 **ADV_TDS.3.1C**

3107 The design shall describe the structure of the TOE in terms of subsystems.

3108 **ADV_TDS.3.2C**

3109 **The design shall describe the TSF in terms of modules.**

3110 **ADV_TDS.3.3C**

3111 The design shall identify all subsystems of the TSF.

3112 **ADV_TDS.3.4C**

3113 The design shall **provide a description of each subsystem of the TSF.**

3114 **ADV_TDS.3.5C**

3115 The design shall provide a description of the interactions among all subsystems of the TSF.

3116 **ADV_TDS.3.6C**

3117 **The design shall provide a mapping from the subsystems of the TSF to the modules of the TSF.**

3118 **ADV_TDS.3.7C**

3119 The design shall describe each **SFR-enforcing module in terms of its purpose and relationship**
3120 **with other modules.**

3121 **ADV_TDS.3.8C**

3122 **The design shall describe each SFR-enforcing module in terms of its SFR-related interfaces,**
3123 **return values from those interfaces, interaction with other modules and called SFR-related**
3124 **interfaces to other SFR-enforcing modules.**

3125 **ADV_TDS.3.9C**

3126 The design shall describe each **SFR-supporting and SFR-non-interfering module in terms of its**
3127 **purpose and interaction with other modules.**

3128 **ADV_TDS.3.10C**

3129 The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE design that
3130 they invoke.

3131 **Evaluator action elements**3132 **ADV_TDS.3.1E**

3133 The evaluator shall confirm that the information provided meets all requirements for content and
3134 presentation of evidence.

3135 **ADV_TDS.3.2E**

3136 The evaluator shall determine that the design is an accurate and complete instantiation of all security
3137 functional requirements.

3138 **10.7.7 ADV_TDS.4 Semiformal modular design**

3139 Dependencies: ADV_FSP.5 Complete semi-formal functional specification with additional error
3140 information

3141 **Developer action elements**3142 **ADV_TDS.4.1D**

3143 The developer shall provide the design of the TOE.

3144 **ADV_TDS.4.2D**

3145 The developer shall provide a mapping from the TSFI of the functional specification to the lowest level
3146 of decomposition available in the TOE design.

3147 **Content and presentation elements**3148 **ADV_TDS.4.1C**

3149 The design shall describe the structure of the TOE in terms of subsystems.

3150 **ADV_TDS.4.2C**

3151 The design shall describe the TSF in terms of modules, **designating each module as SFR-enforcing,**
3152 **SFR-supporting, or SFR-non-interfering.**

3153 **ADV_TDS.4.3C**

3154 The design shall identify all subsystems of the TSF.

3155 **ADV_TDS.4.4C**

3156 The design shall provide a **semiformal** description of each subsystem of the TSF, **supported by**
3157 **informal, explanatory text where appropriate.**

3158 **ADV_TDS.4.5C**

3159 The design shall provide a description of the interactions among all subsystems of the TSF.

- 3160 **ADV_TDS.4.6C**
- 3161 The design shall provide a mapping from the subsystems of the TSF to the modules of the TSF.
- 3162 **10.7.7.1.1 ADV_TDS.4.7C**
- 3163 The design shall describe each SFR-enforcing **and SFR-supporting** module in terms of its purpose and
3164 relationship with other modules.
- 3165 **10.7.7.1.2 ADV_TDS.4.8C**
- 3166 The design shall describe each SFR-enforcing **and SFR-supporting** module in terms of its SFR-related
3167 interfaces, return values from those interfaces, interaction with other modules and called SFR-related
3168 interfaces to other SFR-enforcing **or SFR-supporting** modules.
- 3169 **ADV_TDS.4.9C**
- 3170 The design shall describe each SFR-non-interfering module in terms of its purpose and interaction
3171 with other modules.
- 3172 **ADV_TDS.4.10C**
- 3173 The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE design that
3174 they invoke.
- 3175 **Evaluator action elements**
- 3176 **ADV_TDS.4.1E**
- 3177 The evaluator shall confirm that the information provided meets all requirements for content and
3178 presentation of evidence.
- 3179 **ADV_TDS.4.2E**
- 3180 The evaluator shall determine that the design is an accurate and complete instantiation of all security
3181 functional requirements.
- 3182 **10.7.8 ADV_TDS.5 Complete semiformal modular design**
- 3183 Dependencies: ADV_FSP.5 Complete semi-formal functional specification with additional error
3184 information
- 3185 **Developer action elements**
- 3186 **ADV_TDS.5.1D**
- 3187 The developer shall provide the design of the TOE.
- 3188 **ADV_TDS.5.2D**
- 3189 The developer shall provide a mapping from the TSFI of the functional specification to the lowest level
3190 of decomposition available in the TOE design.

3191 **Content and presentation elements**

3192 **ADV_TDS.5.1C**

3193 The design shall describe the structure of the TOE in terms of subsystems.

3194 **ADV_TDS.5.2C**

3195 The design shall describe the TSF in terms of modules, designating each module as SFR-enforcing, SFR-
3196 supporting, or SFR-non-interfering.

3197 **ADV_TDS.5.3C**

3198 The design shall identify all subsystems of the TSF.

3199 **ADV_TDS.5.4C**

3200 The design shall provide a semiformal description of each subsystem of the TSF, supported by
3201 informal, explanatory text where appropriate.

3202 **ADV_TDS.5.5C**

3203 The design shall provide a description of the interactions among all subsystems of the TSF.

3204 **ADV_TDS.5.6C**

3205 The design shall provide a mapping from the subsystems of the TSF to the modules of the TSF.

3206 **ADV_TDS.5.7C**

3207 The design shall **provide a semiformal description of** each module in terms of its **purpose,**
3208 **interaction,** interfaces, return values from those interfaces, and called interfaces to other modules,
3209 **supported by informal, explanatory text where appropriate.**

3210 **ADV_TDS.5.8C**

3211 The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE design that
3212 they invoke.

3213 **Evaluator action elements**

3214 **ADV_TDS.5.1E**

3215 The evaluator shall confirm that the information provided meets all requirements for content and
3216 presentation of evidence.

3217 **ADV_TDS.5.2E**

3218 The evaluator shall determine that the design is an accurate and complete instantiation of all security
3219 functional requirements.

3220 **10.7.9 ADV_TDS.6 Complete semiformal modular design with formal high-level design**
3221 **presentation**

3222 Dependencies: ADV_FSP.6 Complete semi-formal functional specification with additional formal
3223 specification

3224 **Developer action elements**

3225 **ADV_TDS.6.1D**

3226 The developer shall provide the design of the TOE.

3227 **ADV_TDS.6.2D**

3228 The developer shall provide a mapping from the TSFI of the functional specification to the lowest level
3229 of decomposition available in the TOE design.

3230 **ADV_TDS.6.3D**

3231 **The developer shall provide a formal specification of the TSF subsystems.**

3232 **ADV_TDS.6.4D**

3233 **The developer shall provide a proof of correspondence between the formal specifications of the**
3234 **TSF subsystems and of the functional specification.**

3235 **Content and presentation elements**

3236 **ADV_TDS.6.1C**

3237 The design shall describe the structure of the TOE in terms of subsystems.

3238 **ADV_TDS.6.2C**

3239 The design shall describe the TSF in terms of modules, designating each module as SFR-enforcing, SFR-
3240 supporting, or SFR-non-interfering.

3241 **ADV_TDS.6.3C**

3242 The design shall identify all subsystems of the TSF.

3243 **ADV_TDS.6.4C**

3244 The design shall provide a semiformal description of each subsystem of the TSF, supported by
3245 informal, explanatory text where appropriate.

3246 **ADV_TDS.6.5C**

3247 The design shall provide a description of the interactions among all subsystems of the TSF.

3248 **ADV_TDS.6.6C**

3249 The design shall provide a mapping from the subsystems of the TSF to the modules of the TSF.

3250 **ADV_TDS.6.7C**

3251 The design shall **describe** each module in **semiformal style in** terms of its purpose, interaction,
3252 interfaces, return values from those interfaces, and called interfaces to other modules, supported by
3253 informal, explanatory text where appropriate.

3254 **ADV_TDS.6.8C**

3255 **The formal specification of the TSF subsystems shall describe the TSF using a formal style,**
 3256 **supported by informal, explanatory text where appropriate.**

3257 **ADV_TDS.6.9C**

3258 The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE design that
 3259 they invoke.

3260 **ADV_TDS.6.10C**

3261 **The proof of correspondence between the formal specifications of the TSF subsystems and of**
 3262 **the functional specification shall demonstrate that all behaviour described in the TOE design is**
 3263 **a correct and complete refinement of the TSFI that invoked it.**

3264 **Evaluator action elements**3265 **ADV_TDS.6.1E**

3266 The evaluator shall confirm that the information provided meets all requirements for content and
 3267 presentation of evidence.

3268 **ADV_TDS.6.2E**

3269 The evaluator shall determine that the design is an accurate and complete instantiation of all security
 3270 functional requirements.

3271 **10.8 Composite design compliance (ADV_COMP)**3272 **10.8.1 Objectives**

3273 The aim of this family is to determine whether the requirements on the dependent component,
 3274 imposed by the related base component, are fulfilled in the composite product.

3275 **10.8.2 Component levelling**

3276 This family contains only one component.

3277 **10.8.3 Application notes**

3278 The requirements on the dependent component, imposed by the related base component, can be
 3279 formulated in the relevant base component-related user guidance, *ETR for composite evaluation* (in
 3280 form of observations and recommendations) and report of the base component evaluation authority
 3281 (e.g. in form of constraints and recommendations). The developer of the dependent component shall
 3282 regard each of these sources, if available (cf. Table 2, Clause 14 in ISO/IEC 15408-1), and implement
 3283 the dependent component in such a way that the applicable requirements are fulfilled. The composite
 3284 product evaluator shall verify that all stipulations for the dependent component that are imposed by
 3285 the base component and provided in its evaluation related documentation are fulfilled by the
 3286 composite product, i.e. have been taken into account by the dependent component developer.

3287 The composite product evaluation sponsor shall ensure that the following is made available for the
 3288 composite product evaluator:

- 3289 • the base component-related user guidance,

- 3290 • the base component-related *ETR for composite evaluation* prepared by the base component
3291 evaluator,
- 3292 • the report of the base component evaluation authority,
- 3293 • a rationale for secure composite product implementation including evidence prepared by the
3294 dependent component developer.

3295 The TSF of the composite product are represented at various levels of abstraction in the families of the
3296 development class ADV. From experience, the appropriate levels of design representation for
3297 examining, whether the requirements of the base component are fulfilled by the composite product,
3298 are the TOE design (ADV_TDS), security architecture (ADV_ARC) and the implementation (ADV_IMP).
3299 In case that, these design representation levels are not available (e.g. due to the assurance package
3300 chosen is EAL1), the current family is not applicable (see the next paragraph for the reason).

3301 Due to the definition of the composite product (cf. Subclause 13.3.2.2 in ISO/IEC 15408-1) the
3302 interface between its base component and dependent component is the internal one, hence, a
3303 functional specification (ADV_FSP) as representation level is not appropriate for analysing the design
3304 compliance.

3305 Security architecture ADV_ARC as assurance family is dedicated to ensure that integrative security
3306 services like domain separation, self-protection and non-bypassability properly work. It is impossible
3307 and not the sense of the composite evaluation to have an insight into the architectural internals of the
3308 related base component (it is a matter of the base component evaluation). What the composite product
3309 evaluator has to do in the context of ADV_ARC is

- 3310 i. to determine whether the application uses services of the underlying platform within its own
3311 Composite-ST to provide domain separation, self-protection, non-bypassability and protected
3312 start-up; if no, there is no further composite activities for ADV_ARC; if yes, then
- 3313 ii. the evaluator has to determine, whether the dependent component uses these services of the base
3314 component in an appropriate/secure way.

3315 As consistency of the composite product security policy has already been considered in the context of
3316 the Security Target in the assurance family ASE_COMP, there is no necessity to consider non-
3317 contradictoriness of the security policy model (ADV_SPM) of the composite product and the security
3318 policy model of its related base component.

3319 **10.8.4 ADV_COMP.1 Design compliance with the base component-related user guidance, ETR for** 3320 **composite evaluation and report of the base component evaluation authority**

3321 Dependencies: No dependencies

3322 **10.8.4.1 Developer action elements**

3323 **10.8.4.1.1 ADV_COMP.1.1D**

3324 **The developer shall provide a design compliance justification**

3325 **10.8.4.2 Content and presentation elements**

3326 **10.8.4.2.1 ADV_COMP.1.1C**

3327 **The design compliance justification shall provide a rationale for design compliance – on an**
3328 **appropriate representation level – of how the requirements on the dependent component that**
3329 **are imposed by the related base component are fulfilled in the composite product.**

3330 10.8.4.3 Evaluator action elements

3331 10.8.4.3.1 ADV_COMP.1.1E

3332 The evaluator shall confirm that the rationale for design compliance is complete, coherent, and
3333 internally consistent.

3334 11 Class AGD: Guidance documents

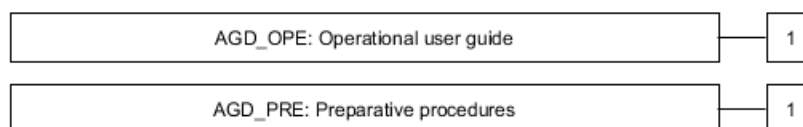
3335 11.1 Introduction

3336 The guidance documents class provides the requirements for guidance documentation for all user
3337 roles. For the secure preparation and operation of the TOE it is necessary to describe all relevant
3338 aspects for the secure handling of the TOE. The class also addresses the possibility of unintended
3339 incorrect configuration or handling of the TOE.

3340 In many cases it may be appropriate that guidance is provided in separate documents for preparation
3341 and operation of the TOE, or even separate for different user roles as end-users, administrators,
3342 application programmers using software or hardware interfaces, etc.

3343 The guidance documents class is subdivided into two families which are concerned with the
3344 preparative user guidance (what has to be done to transform the delivered TOE into its evaluated
3345 configuration in the operational environment as described in the ST) and with the operational user
3346 guidance (what has to be done during the operation of the TOE in its evaluated configuration).

3347 Figure 9 shows the families within this class, and the hierarchy of components within the families.



3348

3349 **Figure 9 — AGD: Guidance documents class decomposition**

3350 11.2 Operational user guidance (AGD_OPE)

3351 11.2.1 Objectives

3352 Operational user guidance refers to written material that is intended to be used by all types of users of
3353 the TOE in its evaluated configuration: end-users, persons responsible for maintaining and
3354 administering the TOE in a correct manner for maximum security, and by others (e.g. programmers)
3355 using the TOE's external interfaces. Operational user guidance describes the security functionality
3356 provided by the TSF, provides instructions and guidelines (including warnings), helps to understand
3357 the TSF and includes the security-critical information, and the security-critical actions required, for its
3358 secure use. Misleading and unreasonable guidance should be absent from the guidance
3359 documentation, and secure procedures for all modes of operation should be addressed. Insecure states
3360 should be easy to detect.

3361 The operational user guidance provides a measure of confidence that non-malicious users,
3362 administrators, application providers and others exercising the external interfaces of the TOE will
3363 understand the secure operation of the TOE and will use it as intended. The evaluation of the user
3364 guidance includes investigating whether the TOE can be used in a manner that is insecure but that the

3365 user of the TOE would reasonably believe to be secure. The objective is to minimise the risk of human
 3366 or other errors in operation that may deactivate, disable, or fail to activate security functionality,
 3367 resulting in an undetected insecure state.

3368 **11.2.2 Component levelling**

3369 This family contains only one component.

3370 **11.2.3 Application notes**

3371 There may be different user roles or groups that are recognised by the TOE and that can interact with
 3372 the TSF. These user roles and groups should be taken into consideration by the operational user
 3373 guidance. They may be roughly grouped into administrators and non-administrative users, or more
 3374 specifically grouped into persons responsible for receiving, accepting, installing and maintaining the
 3375 TOE, application programmers, revisors, auditors, daily-management, end-users. Each role can
 3376 encompass an extensive set of capabilities, or can be a single one.

3377 The requirement AGD_OPE.1.1C encompasses the aspect that any warnings to the users during
 3378 operation of a TOE with regard to the security problem definition and the security objectives for the
 3379 operational environment described in the PP/ST are appropriately covered in the user guidance.

3380 The concept of secure values, as employed in AGD_OPE.1.3C, has relevance where a user has control
 3381 over security parameters. Guidance needs to be provided on secure and insecure settings for such
 3382 parameters.

3383 AGD_OPE.1.4C requires that the user guidance describes the appropriate reactions to all security-
 3384 relevant events. Although many security-relevant events are the result of performing functions, this
 3385 need not always be the case (e.g. the audit log fills up, an intrusion is detected). Furthermore, a
 3386 security-relevant event may happen as a result of a specific chain of functions or, conversely, several
 3387 security-relevant events may be triggered by one function.

3388 AGD_OPE.1.7C requires that the user guidance is clear and reasonable. Misleading or unreasonable
 3389 guidance may result in a user of the TOE believing that the TOE is secure when it is not.

3390 An example of misleading guidance would be the description of a single guidance instruction that
 3391 could be parsed in more than one way, one of which may result in an insecure state.

3392 An example of unreasonable guidance would be a recommendation to follow a procedure that is so
 3393 complicated that it cannot reasonably be expected that users will follow this guidance.

3394 **11.2.4 AGD_OPE.1 Operational user guidance**

3395 Dependencies: ADV_FSP.1 Basic functional specification

3396 **Developer action elements**

3397 **AGD_OPE.1.1D**

3398 **The developer shall provide operational user guidance.**

3399 **Content and presentation elements**

3400 **AGD_OPE.1.1C**

3401 **The operational user guidance shall describe, for each user role, the user-accessible functions**
 3402 **and privileges that should be controlled in a secure processing environment, including**
 3403 **appropriate warnings.**

3404 **AGD_OPE.1.2C**

3405 **The operational user guidance shall describe, for each user role, how to use the available**
 3406 **interfaces provided by the TOE in a secure manner.**

3407 **AGD_OPE.1.3C**

3408 **The operational user guidance shall describe, for each user role, the available functions and**
 3409 **interfaces, in particular all security parameters under the control of the user, indicating secure**
 3410 **values as appropriate.**

3411 **AGD_OPE.1.4C**

3412 **The operational user guidance shall, for each user role, clearly present each type of security-**
 3413 **relevant event relative to the user-accessible functions that need to be performed, including**
 3414 **changing the security characteristics of entities under the control of the TSF.**

3415 **AGD_OPE.1.5C**

3416 **The operational user guidance shall identify all possible modes of operation of the TOE**
 3417 **(including operation following failure or operational error), their consequences and**
 3418 **implications for maintaining secure operation.**

3419 **AGD_OPE.1.6C**

3420 **The operational user guidance shall, for each user role, describe the security controls to be**
 3421 **followed in order to fulfil the security objectives for the operational environment as described**
 3422 **in the ST.**

3423 **AGD_OPE.1.7C**

3424 **The operational user guidance shall be clear and reasonable.**

3425 **Evaluator action elements**

3426 **AGD_OPE.1.1E**

3427 **The evaluator shall confirm that the information provided meets all requirements for content**
 3428 **and presentation of evidence.**

3429 **11.3 Preparative procedures (AGD_PRE)**

3430 **11.3.1 Objectives**

3431 **Preparative procedures are useful for ensuring that the TOE has been received and installed in a**
 3432 **secure manner as intended by the developer. The requirements for preparation call for a secure**
 3433 **transition from the delivered TOE to its initial operational environment. This includes investigating**

3434 whether the TOE can be configured or installed in a manner that is insecure but that the user of the
 3435 TOE would reasonably believe to be secure.

3436 **11.3.2 Component levelling**

3437 This family contains only one component.

3438 **11.3.3 Application notes**

3439 It is recognised that the application of these requirements will vary depending on aspects such as
 3440 whether the TOE is delivered in an operational state, or whether it has to be installed at the TOE
 3441 owner's site, etc.

3442 The first process covered by the preparative procedures is the consumer's secure acceptance of the
 3443 received TOE in accordance with the developer's delivery procedures. If the developer has not defined
 3444 delivery procedures, security of the acceptance has to be ensured otherwise.

3445 Installation of the TOE includes transforming its operational environment into a state that conforms to
 3446 the security objectives for the operational environment provided in the ST.

3447 It might also be the case that no installation is necessary, for example a smart card. In this case it may
 3448 be inappropriate to require and analyse installation procedures.

3449 The requirements in this assurance family are presented separately from those in the Operational user
 3450 guidance (AGD_OPE) family, due to the infrequent, possibly one-time use of the preparative
 3451 procedures.

3452 **11.3.4 AGD_PRE.1 Preparative procedures**

3453 Dependencies: No dependencies.

3454 **Developer action elements**

3455 **AGD_PRE.1.1D**

3456 **The developer shall provide the TOE including its preparative procedures.**

3457 **Content and presentation elements**

3458 **AGD_PRE.1.1C**

3459 **The preparative procedures shall describe all the steps necessary for secure acceptance of the**
 3460 **delivered TOE in accordance with the developer's delivery procedures.**

3461 **AGD_PRE.1.2C**

3462 **The preparative procedures shall describe all the steps necessary for secure installation of the**
 3463 **TOE and for the secure preparation of the operational environment in accordance with the**
 3464 **security objectives for the operational environment as described in the ST.**

3465 **Evaluator action elements**

3466 **AGD_PRE.1.1E**

3467 **The evaluator shall confirm that the information provided meets all requirements for content**
 3468 **and presentation of evidence.**

3469 **AGD_PRE.1.2E**

3470 **The evaluator shall apply the preparative procedures to confirm that the TOE can be prepared**
 3471 **securely for operation.**

3472 **12 Class ALC: Life-cycle support**

3473 **12.1 Introduction**

3474 Life-cycle support is an aspect of establishing appropriate security controls in the development,
 3475 production, delivery and maintenance of the TOE. Confidence in the correspondence between the TOE
 3476 security requirements and the TOE is greater if security analysis and the production of the evidence
 3477 are done on a regular basis as an integral part of the development, production, delivery and
 3478 maintenance activities.

3479 During the life-cycle of the TOE it is distinguished whether the TOE is under the responsibility of the
 3480 TOE developer or the user rather than whether it is located in the development or the user
 3481 environment. The point of transition is when the TOE is accepted by the user. User in this context
 3482 relates to the end-user as well as product- and system integrators.

3483 The ALC class consists of nine families:

- 3484 • Development Life-cycle definition (ALC_LCD) provides requirements for the developer's
 3485 description of the life-cycle model used in the development, production, delivery and
 3486 maintenance life-cycle of the TOE;
- 3487 • CM capabilities (ALC_CMC) provides requirements for the management of the configuration
 3488 items;
- 3489 • CM scope (ALC_CMS) requires a minimum set of configuration items to be managed in the
 3490 defined way;
- 3491 • Developer environment security (ALC_DVS) is concerned with the developer's physical, logical,
 3492 procedural, personnel, and other security controls;
- 3493 • Tools and techniques (ALC_TAT) provides requirements for the development tools and
 3494 implementation standards used by the developer;
- 3495 • Flaw remediation (ALC_FLR) provides requirements for the handling of security flaws.
- 3496 • Delivery (ALC_DEL) provides requirements for the procedures used for the delivery of the TOE
 3497 to the downstream user. Delivery processes occurring during the development of the TOE are
 3498 denoted rather as transfers, and are handled in the context of integration and acceptance
 3499 procedures in other families of this class.
- 3500 • ALC_TDA is concerned with the generation of certain artefacts during the development
 3501 process.

- ALC_COMP is concerned with the integration of composition parts and a consistency check of delivery procedures.

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

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

Figure 10 — ALC: Life-cycle support class decomposition

12.2 CM capabilities (ALC_CMC)

12.2.1 Objectives

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

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

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

The objectives of this family include the following:

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

12.2.2 Component levelling

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

12.2.3 Application notes

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

3536 For developer organizations that specify more than one CM application, or include different instances
 3537 of a CM application within the scope of the TOEs design, development, production and maintenance, it
 3538 is required to document all of them. For evaluation purposes, the set of CM applications should be
 3539 regarded as parts of an overall CM system, applicable to the TOE, which is addressed in the criteria.

3540 The overall CM system should address any aspects of integration between component CM applications.

3541 Several elements of this family refer to configuration items. These elements identify CM requirements
 3542 to be imposed on all items identified in the configuration list, but leave the contents of the list to the
 3543 discretion of the developer. CM scope (ALC_CMS) can be used to narrow this discretion by identifying
 3544 specific items that must be included in the configuration list, and hence within the scope of the overall
 3545 CM system.

3546 ALC_CMC.2.3C introduces a requirement that the CM system uniquely identify all configuration items.
 3547 This also requires that modifications to configuration items result in a new, unique identifier being
 3548 assigned to the configuration item.

3549 ALC_CMC.3.8C introduces the requirement that the evidence shall demonstrate that the CM system
 3550 operates in accordance with the CM plan. Examples of such evidence might be documentation such as
 3551 screen snapshots or audit trail output from the CM system, or a detailed demonstration of the CM
 3552 system by the developer. The evaluator is responsible for determining that this evidence is sufficient
 3553 to show that the CM system operates in accordance with the CM plan.

3554 ALC_CMC.4.5C introduces a requirement that the CM system provide an automated means to support
 3555 the production of the TOE. This requires that the CM system provide an automated means to assist in
 3556 determining that the correct configuration items are used in generating the TOE.

3557 ALC_CMC.5.10C introduces a requirement that the CM system provide an automated means to
 3558 ascertain the changes between the TOE and its preceding version. If no previous version of the TOE
 3559 exists, the developer still needs to provide an automated means to ascertain the changes between the
 3560 TOE and a future version of the TOE.

3561 **12.2.4 ALC_CMC.1 Labelling of the TOE**

3562 Dependencies: ALC_CMS.1 TOE CM coverage

3563 **Objectives**

3564 A unique reference is required to ensure that there is no ambiguity in terms of which instance of the
 3565 TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be
 3566 aware of which instance of the TOE they are using.

3567 **Developer action elements**

3568 **ALC_CMC.1.1D**

3569 **The developer shall provide the TOE and a unique reference for the TOE.**

3570 **Content and presentation elements**

3571 **ALC_CMC.1.1C**

3572 **The TOE shall be labelled with its unique reference.**

3573 **Evaluator action elements**

3574 **ALC_CMC.1.1E**

3575 **The evaluator shall confirm that the information provided meets all requirements for content**
 3576 **and presentation of evidence.**

3577 **12.2.5 ALC_CMC.2 Use of the CM system**

3578 Dependencies: ALC_CMS.1 TOE CM coverage

3579 **Objectives**

3580 A unique reference is required to ensure that there is no ambiguity in terms of which instance of the
 3581 TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be
 3582 aware of which instance of the TOE they are using.

3583 Unique identification of the configuration items leads to a clearer understanding of the composition of
 3584 the TOE, which in turn helps to determine those items which are subject to the evaluation
 3585 requirements for the TOE.

3586 The use of a CM system increases assurance that the configuration items are maintained in a
 3587 controlled manner.

3588 **Developer action elements**

3589 **ALC_CMC.2.1D**

3590 The developer shall provide the TOE and a unique reference for the TOE.

3591 **ALC_CMC.2.2D**

3592 **The developer shall provide the CM documentation.**

3593 **ALC_CMC.2.3D**

3594 **The developer shall use a CM system.**

3595 **Content and presentation elements**

3596 **ALC_CMC.2.1C**

3597 The TOE shall be labelled with its unique reference.

3598 **ALC_CMC.2.2C**

3599 **The CM documentation shall describe the method used to uniquely identify the configuration**
 3600 **items.**

3601 **ALC_CMC.2.3C**

3602 **The CM system shall uniquely identify all configuration items.**

3603 **Evaluator action elements**

3604 **ALC_CMC.2.1E**

3605 The evaluator shall confirm that the information provided meets all requirements for content and
3606 presentation of evidence.

3607 **12.2.6 ALC_CMC.3 Authorisation controls**

3608 Dependencies: ALC_CMS.1 TOE CM coverage

3609 ALC_DVS.1 Identification of security measures

3610 A life-cycle model encompasses the procedures, tools and techniques used to develop
3611 and maintain the TOE. Aspects of the process that may be covered by such a model include design
3612 methods, review procedures, project management controls, change control procedures, test methods
3613 and acceptance procedures. An effective life-cycle model will address these aspects of the
3614 development and maintenance process within an overall management structure that assigns
3615 responsibilities and monitors progress.

3616 There are different types of acceptance situations that are dealt with at different locations in the
3617 criteria:

- 3618 • acceptance of parts delivered by subcontractors (“integration”) should be treated in this family
- 3619 • Development Life-cycle definition (ALC_LCD),
- 3620 • acceptance subsequent to internal transportations in Developer environment security
3621 (ALC_DVS),
- 3622 • acceptance of parts into the CM system in CM capabilities (ALC_CMC), and
- 3623 • acceptance of the delivered TOE by the consumer in Delivery (ALC_DEL).

3624 The first three types may overlap.

3625 Although life-cycle definition deals with the maintenance of the TOE and hence with aspects becoming
3626 relevant after the completion of the evaluation, its evaluation adds assurance through an analysis of
3627 the life-cycle information for the TOE provided at the time of the evaluation.

3628 A life-cycle model provides for the necessary control over the development and maintenance of the
3629 TOE, if the model enables sufficient minimisation of the danger that the TOE will not meet its security
3630 requirement.

3631 A measurable life-cycle model is a model using some quantitative valuation (arithmetic parameters
3632 and/or metrics) of the managed product in order to measure development properties of the product.
3633 Typical metrics are source code complexity metrics, defect density (errors per size of code) or mean
3634 time to failure. For the security evaluation all those metrics are of relevance, which are used to
3635 increase quality by decreasing the probability of faults and thereby in turn increasing assurance in the
3636 security of the TOE.

3637 One should take into account that there exist standardised life-cycle models on the one hand (like the
3638 waterfall model) and standardised metrics on the other hand (like error density), which may be

3639 combined. The ISO/IEC 15408 series does not require the life-cycle to follow exactly one standard
3640 defining both aspects.

3641 ALC_LCD.1 Developer defined life-cycle processes

3642 **Objectives**

3643 A unique reference is required to ensure that there is no ambiguity in terms of which instance of the
3644 TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be
3645 aware of which instance of the TOE they are using.

3646 Unique identification of the configuration items leads to a clearer understanding of the composition of
3647 the TOE, which in turn helps to determine those items which are subject to the evaluation
3648 requirements for the TOE.

3649 The use of a CM system increases assurance that the configuration items are maintained in a
3650 controlled manner.

3651 Providing controls to ensure that unauthorised modifications are not made to the TOE (“CM access
3652 control”), and ensuring proper functionality and use of the CM system, helps to maintain the integrity
3653 of the TOE.

3654 **Developer action elements**

3655 **ALC_CMC.3.1D**

3656 The developer shall provide the TOE and a unique reference for the TOE.

3657 **ALC_CMC.3.2D**

3658 The developer shall provide the CM documentation.

3659 **ALC_CMC.3.3D**

3660 The developer shall use a CM system.

3661 **Content and presentation elements**

3662 **ALC_CMC.3.1C**

3663 The TOE shall be labelled with its unique reference.

3664 **ALC_CMC.3.2C**

3665 The CM documentation shall describe the method used to uniquely identify the configuration items.

3666 **ALC_CMC.3.3C**

3667 The CM system shall uniquely identify all configuration items.

3668 **ALC_CMC.3.4C**

3669 **The CM system shall provide controls such that only authorised changes are made to the**
3670 **configuration items.**

3671 **ALC_CMC.3.5C**

3672 **The CM documentation shall include a CM plan.**

3673 **ALC_CMC.3.6C**

3674 **The CM plan shall describe how the CM system is used for the development of the TOE.**

3675 **ALC_CMC.3.7C**

3676 **The evidence shall demonstrate that all configuration items are being maintained under the CM**
3677 **system.**

3678 **ALC_CMC.3.8C**

3679 **The evidence shall demonstrate that the CM system is being operated in accordance with the**
3680 **CM plan.**

3681 **Evaluator action elements**

3682 **ALC_CMC.3.1E**

3683 The evaluator shall confirm that the information provided meets all requirements for content and
3684 presentation of evidence.

3685 **12.2.7 ALC_CMC.4 Production support, acceptance procedures and automation**

3686 Dependencies: ALC_CMS.1 TOE CM coverage

3687 ALC_DVS.1 Identification of security measures

3688 A life-cycle model encompasses the procedures, tools and techniques used to develop
3689 and maintain the TOE. Aspects of the process that may be covered by such a model include design
3690 methods, review procedures, project management controls, change control procedures, test methods
3691 and acceptance procedures. An effective life-cycle model will address these aspects of the
3692 development and maintenance process within an overall management structure that assigns
3693 responsibilities and monitors progress.

3694 There are different types of acceptance situations that are dealt with at different locations in the
3695 criteria:

- 3696 • acceptance of parts delivered by subcontractors ("integration") should be treated in this family
- 3697 • Development Life-cycle definition (ALC_LCD),
- 3698 • acceptance subsequent to internal transportations in Developer environment security
3699 (ALC_DVS),
- 3700 • acceptance of parts into the CM system in CM capabilities (ALC_CMC), and
- 3701 • acceptance of the delivered TOE by the consumer in Delivery (ALC_DEL).

3702 The first three types may overlap.

3703 Although life-cycle definition deals with the maintenance of the TOE and hence with aspects becoming
 3704 relevant after the completion of the evaluation, its evaluation adds assurance through an analysis of
 3705 the life-cycle information for the TOE provided at the time of the evaluation.

3706 A life-cycle model provides for the necessary control over the development and maintenance of the
 3707 TOE, if the model enables sufficient minimisation of the danger that the TOE will not meet its security
 3708 requirement.

3709 A measurable life-cycle model is a model using some quantitative valuation (arithmetic parameters
 3710 and/or metrics) of the managed product in order to measure development properties of the product.
 3711 Typical metrics are source code complexity metrics, defect density (errors per size of code) or mean
 3712 time to failure. For the security evaluation all those metrics are of relevance, which are used to
 3713 increase quality by decreasing the probability of faults and thereby in turn increasing assurance in the
 3714 security of the TOE.

3715 One should take into account that there exist standardised life-cycle models on the one hand (like the
 3716 waterfall model) and standardised metrics on the other hand (like error density), which may be
 3717 combined. The ISO/IEC 15408 series does not require the life-cycle to follow exactly one standard
 3718 defining both aspects.

3719 ALC_LCD.1 Developer defined life-cycle processes

3720 Objectives

3721 A unique reference is required to ensure that there is no ambiguity in terms of which instance of the
 3722 TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be
 3723 aware of which instance of the TOE they are using.

3724 Unique identification of the configuration items leads to a clearer understanding of the composition of
 3725 the TOE, which in turn helps to determine those items which are subject to the evaluation
 3726 requirements for the TOE.

3727 The use of a CM system increases assurance that the configuration items are maintained in a
 3728 controlled manner.

3729 Providing access controls to help ensure that unauthorised modifications are not made to the TOE
 3730 ("CM access control"), and ensuring proper functionality and use of the CM system, helps to maintain
 3731 the integrity of the TOE.

3732 The purpose of the acceptance procedures is to ensure that the parts of the TOE are of adequate
 3733 quality and to confirm that any creation or modification of configuration items is authorised.
 3734 Acceptance procedures are an essential element in integration processes and in the life-cycle
 3735 management of the TOE.

3736 In a CM system where the quantity and organization of configuration items is complex, it is difficult to
 3737 control changes without the support of automated tools. In particular, these automated tools need to
 3738 be able to support the numerous changes that occur during development and ensure that those
 3739 changes are authorised. It is an objective of this component to ensure that the configuration items are
 3740 controlled through automated means. In the case where the overall CM system includes more than one
 3741 CM application then automated tools can also support integration between the CM applications and of
 3742 the TOE.

3743 Production support procedures help to ensure that the generation of the TOE from a managed set of
 3744 configuration items is correctly performed in an authorised manner, particularly in the case when
 3745 different developers are involved and integration processes have to be carried out.

3746 **Developer action elements**

3747 **ALC_CMC.4.1D**

3748 The developer shall provide the TOE and a unique reference for the TOE.

3749 **ALC_CMC.4.2D**

3750 The developer shall provide the CM documentation.

3751 **ALC_CMC.4.3D**

3752 The developer shall use a CM system.

3753 **Content and presentation elements**

3754 **ALC_CMC.4.1C**

3755 The TOE shall be labelled with its unique reference.

3756 **ALC_CMC.4.2C**

3757 The CM documentation shall describe the method or methods used to uniquely identify the
 3758 configuration items.

3759 **ALC_CMC.4.3C**

3760 The CM system shall uniquely identify all configuration items.

3761 **ALC_CMC.4.4C**

3762 The CM system shall provide **automated** controls such that only authorised changes are made to the
 3763 configuration items.

3764 **ALC_CMC.4.5C**

3765 **The CM system shall support the production of the TOE by automated means.**

3766 **ALC_CMC.4.6C**

3767 The CM documentation shall include a CM plan.

3768 **ALC_CMC.4.7C**

3769 The CM plan shall describe how the CM system is used for the development of the TOE.

3770 **ALC_CMC.4.8C**

3771 **The CM plan shall describe the procedures used to accept modified or newly created**
 3772 **configuration items as part of the TOE.**

3773 **ALC_CMC.4.9C**

3774 The evidence shall demonstrate that all configuration items are being maintained under the CM
3775 system.

3776 **ALC_CMC.4.10C**

3777 The evidence shall demonstrate that the CM system is being operated in accordance with the CM plan.

3778 **Evaluator action elements**3779 **ALC_CMC.4.1E**

3780 The evaluator shall confirm that the information provided meets all requirements for content and
3781 presentation of evidence.

3782 **12.2.8 ALC_CMC.5 Advanced support**

3783 Dependencies: ALC_CMS.1 TOE CM coverage

3784 ALC_DVS.2 Sufficiency of security measures

3785 A life-cycle model encompasses the procedures, tools and techniques used to develop
3786 and maintain the TOE. Aspects of the process that may be covered by such a model include design
3787 methods, review procedures, project management controls, change control procedures, test methods
3788 and acceptance procedures. An effective life-cycle model will address these aspects of the
3789 development and maintenance process within an overall management structure that assigns
3790 responsibilities and monitors progress.

3791 There are different types of acceptance situations that are dealt with at different locations in the
3792 criteria:

- 3793 • acceptance of parts delivered by subcontractors ("integration") should be treated in this family
- 3794 • Development Life-cycle definition (ALC_LCD),
- 3795 • acceptance subsequent to internal transportations in Developer environment security
3796 (ALC_DVS),
- 3797 • acceptance of parts into the CM system in CM capabilities (ALC_CMC), and
- 3798 • acceptance of the delivered TOE by the consumer in Delivery (ALC_DEL).

3799 The first three types may overlap.

3800 Although life-cycle definition deals with the maintenance of the TOE and hence with aspects becoming
3801 relevant after the completion of the evaluation, its evaluation adds assurance through an analysis of
3802 the life-cycle information for the TOE provided at the time of the evaluation.

3803 A life-cycle model provides for the necessary control over the development and maintenance of the
3804 TOE, if the model enables sufficient minimisation of the danger that the TOE will not meet its security
3805 requirement.

3806 A measurable life-cycle model is a model using some quantitative valuation (arithmetic parameters
 3807 and/or metrics) of the managed product in order to measure development properties of the product.
 3808 Typical metrics are source code complexity metrics, defect density (errors per size of code) or mean
 3809 time to failure. For the security evaluation all those metrics are of relevance, which are used to
 3810 increase quality by decreasing the probability of faults and thereby in turn increasing assurance in the
 3811 security of the TOE.

3812 One should take into account that there exist standardised life-cycle models on the one hand (like the
 3813 waterfall model) and standardised metrics on the other hand (like error density), which may be
 3814 combined. The ISO/IEC 15408 series does not require the life-cycle to follow exactly one standard
 3815 defining both aspects.

3816 ALC_LCD.1 Developer defined life-cycle processes

3817 **Objectives**

3818 A unique reference is required to ensure that there is no ambiguity in terms of which instance of the
 3819 TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be
 3820 aware of which instance of the TOE they are using.

3821 Unique identification of the configuration items leads to a clearer understanding of the composition of
 3822 the TOE, which in turn helps to determine those items which are subject to the evaluation
 3823 requirements for the TOE.

3824 The use of a CM system increases assurance that the configuration items are maintained in a
 3825 controlled manner.

3826 Providing controls to ensure that unauthorised modifications are not made to the TOE ("CM access
 3827 control"), and ensuring proper functionality and use of the CM system, helps to maintain the integrity
 3828 of the TOE.

3829 The purpose of the acceptance procedures is to ensure that the parts of the TOE meet defined criteria
 3830 in regard to the integrity of the TOE. Criteria for acceptance procedures may include code review,
 3831 checking for vulnerabilities, authenticity checking, and functional testing to confirm that any creation
 3832 or modification of configuration items is authorised. Acceptance procedures are an essential element
 3833 in integration processes and in the life-cycle management of the TOE.

3834 In development environments where the configuration items are complex, it is difficult to control
 3835 changes without the support of automated tools. In particular, these automated tools need to be able
 3836 to support the numerous changes that occur during development and ensure that those changes are
 3837 authorised. It is an objective of this component to ensure that the configuration items are controlled
 3838 through automated means. If the TOE is developed by multiple developers, i.e. integration has to take
 3839 place, the use of automatic tools is adequate.

3840 Production support procedures help to ensure that the generation of the TOE from a managed set of
 3841 configuration items is correctly performed in an authorised manner, particularly in the case when
 3842 different developers are involved and integration processes have to be carried out.

3843 Requiring that the CM system be able to identify the version of the implementation representation
 3844 from which the TOE is generated helps to ensure that the integrity of this material is preserved by the
 3845 appropriate technical, physical and procedural safeguards.

3846 Providing an automated means of ascertaining changes between versions of the TOE and identifying
 3847 which configuration items are affected by modifications to other configuration items assists in

3848 determining the impact of the changes between successive versions of the TOE. This in turn can
 3849 provide valuable information in determining whether changes to the TOE result in all configuration
 3850 items being consistent with one another.

3851 **Developer action elements**

3852 **ALC_CMC.5.1D**

3853 The developer shall provide the TOE and a unique reference for the TOE.

3854 **ALC_CMC.5.2D**

3855 The developer shall provide the CM documentation.

3856 **ALC_CMC.5.3D**

3857 The developer shall use a CM system.

3858 **Content and presentation elements**

3859 **ALC_CMC.5.1C**

3860 The TOE shall be labelled with its unique reference.

3861 **ALC_CMC.5.2C**

3862 The CM documentation shall describe the method used to uniquely identify the configuration items.

3863 **ALC_CMC.5.3C**

3864 **The CM documentation shall justify that the acceptance procedures provide for an adequate**
 3865 **and appropriate review of changes to all configuration items.**

3866 **ALC_CMC.5.4C**

3867 The CM system shall uniquely identify all configuration items.

3868 **ALC_CMC.5.5C**

3869 The CM system shall provide automated controls such that only authorised changes are made to the
 3870 configuration items.

3871 **ALC_CMC.5.6C**

3872 The CM system shall support the production of the TOE by automated means.

3873 **ALC_CMC.5.7C**

3874 **The CM system shall ensure that the person responsible for accepting a configuration item into**
 3875 **CM is not the person who developed it.**

3876 **ALC_CMC.5.8C**

3877 **The CM system shall identify the configuration items that comprise the TSF.**

3878 **ALC_CMC.5.9C**

3879 **The CM system shall support the audit of all changes to the TOE by automated means, including**
 3880 **the originator, date, and time in the audit trail.**

3881 **ALC_CMC.5.10C**

3882 **The CM system shall provide an automated means to identify all other configuration items that**
 3883 **are affected by the change of a given configuration item.**

3884 **ALC_CMC.5.11C**

3885 **The CM system shall be able to identify the version of the implementation representation from**
 3886 **which the TOE is generated.**

3887 **ALC_CMC.5.12C**

3888 The CM documentation shall include a CM plan.

3889 **ALC_CMC.5.13C**

3890 The CM plan shall describe how the CM system is used for the development of the TOE.

3891 **ALC_CMC.5.14C**

3892 The CM plan shall describe the procedures used to accept modified or newly created configuration
 3893 items as part of the TOE.

3894 **ALC_CMC.5.15C**

3895 The evidence shall demonstrate that all configuration items are being maintained under the CM
 3896 system.

3897 **ALC_CMC.5.16C**

3898 The evidence shall demonstrate that the CM system is being operated in accordance with the CM plan.

3899 **Evaluator action elements**

3900 **ALC_CMC.5.1E**

3901 The evaluator shall confirm that the information provided meets all requirements for content and
 3902 presentation of evidence.

3903 **ALC_CMC.5.2E**

3904 **The evaluator shall determine that the application of the production support procedures**
 3905 **results in a TOE as provided by the developer for testing activities.**

3906 **12.3 CM scope (ALC_CMS)**

3907 **12.3.1 Objectives**

3908 The objective of this family is to identify items to be included as configuration items and hence placed
 3909 under the CM requirements of CM capabilities (ALC_CMC). Applying configuration management to
 3910 these additional items provides additional assurance that the integrity of TOE is maintained.

3911 **12.3.2 Component levelling**

3912 The components in this family are levelled on the basis of which of the following are required to be
 3913 included as configuration items: the TOE and the evaluation evidence required by the SARs; the parts
 3914 of the TOE; the implementation representation; security flaws; and development tools and related
 3915 information.

3916 **12.3.3 Application notes**

3917 While CM scope (ALC_CMS) mandates a list of configuration items and that each item on this list be
 3918 under CM, CM capabilities (ALC_CMC) leaves the contents of the configuration list to the discretion of
 3919 the developer. CM scope (ALC_CMS) narrows this discretion by identifying items that must be included
 3920 in the configuration list, and hence come under the CM requirements of CM capabilities (ALC_CMC).

3921 **12.3.4 ALC_CMS.1 TOE CM coverage**

3922 Dependencies: No dependencies.

3923 **Objectives**

3924 A CM system can control changes only to those items that have been placed under CM (i.e., the
 3925 configuration items identified in the configuration list). Placing the TOE itself and the evaluation
 3926 evidence required by the other SARs in the ST under CM provides assurance that they have been
 3927 modified in a controlled manner with proper authorisations.

3928 **Application notes**

3929 ALC_CMS.1.1C introduces the requirement that the TOE itself and the evaluation evidence required by
 3930 the other SARs in the ST be included in the configuration list and hence be subject to the CM
 3931 requirements of CM capabilities (ALC_CMC).

3932 **Developer action elements**

3933 **ALC_CMS.1.1D**

3934 **The developer shall provide a configuration list for the TOE.**

3935 **Content and presentation elements**

3936 **ALC_CMS.1.1C**

3937 **The configuration list shall include the following: the TOE itself; and the evaluation evidence**
 3938 **required by the SARs.**

3939 **ALC_CMS.1.2C**

3940 **The configuration list shall uniquely identify the configuration items.**

3941 **Evaluator action elements**

3942 **ALC_CMS.1.1E**

3943 **The evaluator shall confirm that the information provided meets all requirements for content**
 3944 **and presentation of evidence.**

3945 **12.3.5 ALC_CMS.2 Parts of the TOE CM coverage**

3946 Dependencies: No dependencies.

3947 **Objectives**

3948 A CM system can control changes only to those items that have been placed under CM (i.e., the
 3949 configuration items identified in the configuration list). Placing the TOE itself, the parts that comprise
 3950 the TOE, and the evaluation evidence required by the other SARs under CM provides assurance that
 3951 they have been modified in a controlled manner with proper authorisations.

3952 **Application notes**

3953 ALC_CMS.2.1C introduces the requirement that the parts that comprise the TOE (all parts that are
 3954 delivered to the consumer, for example hardware parts or executable files) be included in the
 3955 configuration list and hence be subject to the CM requirements of CM capabilities (ALC_CMC).

3956 ALC_CMS.2.3C introduces the requirement that the configuration list indicate the developer of each
 3957 TSF relevant configuration item.

3958 **Developer action elements**

3959 **ALC_CMS.2.1D**

3960 The developer shall provide a configuration list for the TOE.

3961 **Content and presentation elements**

3962 **ALC_CMS.2.1C**

3963 The configuration list shall include the following: the TOE itself; the evaluation evidence required by
 3964 the SARs; **and the parts that comprise the TOE.**

3965 **ALC_CMS.2.2C**

3966 The configuration list shall uniquely identify the configuration items.

3967 **ALC_CMS.2.3C**

3968 **For each TSF relevant configuration item, the configuration list shall indicate the developer of**
 3969 **the item.**

3970 **Evaluator action elements**

3971 **ALC_CMS.2.1E**

3972 The evaluator shall confirm that the information provided meets all requirements for content and
 3973 presentation of evidence.

3974 **12.3.6 ALC_CMS.3 Implementation representation CM coverage**

3975 Dependencies: No dependencies.

3976 **Objectives**

3977 A CM system can control changes only to those items that have been placed under CM (i.e., the
3978 configuration items identified in the configuration list). Placing the TOE itself, the parts that comprise
3979 the TOE, the TOE implementation representation and the evaluation evidence required by the other
3980 SARs under CM provides assurance that they have been modified in a controlled manner with proper
3981 authorisations.

3982 **Application notes**

3983 ALC_CMS.3.1C introduces the requirement that the TOE implementation representation be included in
3984 the list of configuration items and hence be subject to the CM requirements of CM capabilities
3985 (ALC_CMC).

3986 **Developer action elements**

3987 **ALC_CMS.3.1D**

3988 The developer shall provide a configuration list for the TOE.

3989 **Content and presentation elements**

3990 **ALC_CMS.3.1C**

3991 The configuration list shall include the following: the TOE itself; the evaluation evidence required by
3992 the SARs; the parts that comprise the TOE; **and the implementation representation.**

3993 **ALC_CMS.3.2C**

3994 The configuration list shall uniquely identify the configuration items.

3995 **ALC_CMS.3.3C**

3996 For each TSF relevant configuration item, the configuration list shall indicate the developer of the item.

3997 **Evaluator action elements**

3998 **ALC_CMS.3.1E**

3999 The evaluator shall confirm that the information provided meets all requirements for content and
4000 presentation of evidence.

4001 **12.3.7 ALC_CMS.4 Problem tracking CM coverage**

4002 Dependencies: No dependencies.

4003 **Objectives**

4004 A CM system can control changes only to those items that have been placed under CM (i.e., the
4005 configuration items identified in the configuration list). Placing the TOE itself, the parts that comprise
4006 the TOE, the TOE implementation representation and the evaluation evidence required by the other

4007 SARs under CM provides assurance that they have been modified in a controlled manner with proper
4008 authorisations.

4009 Placing security flaw reports under CM ensures that the integrity of the reports is maintained and that
4010 access to them is managed, further, it may support developers in tracking security flaws to their
4011 resolution.

4012 **Application notes**

4013 ALC_CMS.4.1C introduces the requirement that reports of identified security flaws be included in the
4014 configuration list and hence be subject to the CM requirements of CM capabilities (ALC_CMC). This
4015 requires that information regarding previously identified security flaw reports and their resolution be
4016 maintained.

4017 **Developer action elements**

4018 **ALC_CMS.4.1D**

4019 The developer shall provide a configuration list for the TOE.

4020 **Content and presentation elements**

4021 **ALC_CMS.4.1C**

4022 The configuration list shall include the following: the TOE itself; the evaluation evidence required by
4023 the SARs; the parts that comprise the TOE; the implementation representation; **and security flaw**
4024 **reports and resolution status.**

4025 **ALC_CMS.4.2C**

4026 The configuration list shall uniquely identify the configuration items.

4027 **ALC_CMS.4.3C**

4028 For each TSF relevant configuration item, the configuration list shall indicate the developer of the item.

4029 **Evaluator action elements**

4030 **ALC_CMS.4.1E**

4031 The evaluator shall confirm that the information provided meets all requirements for content and
4032 presentation of evidence.

4033 **12.3.8 ALC_CMS.5 Development tools CM coverage**

4034 Dependencies: No dependencies.

4035 **Objectives**

4036 A CM system can control changes only to those items that have been placed under CM (i.e., the
4037 configuration items identified in the configuration list). Placing the TOE itself, the parts that comprise
4038 the TOE, the TOE implementation representation and the evaluation evidence required by the other
4039 SARs under CM provides assurance that they have been modified in a controlled manner with proper
4040 authorisations.

4041 Placing security flaw reports under CM ensures that the integrity of the reports is maintained and that
 4042 access to them is managed, further, it may support developers in tracking security flaws to their
 4043 resolution.

4044 Development tools play an important role in ensuring the production of a quality version of the TOE.
 4045 Therefore, it is important to control modifications to these tools.

4046 **Application notes**

4047 ALC_CMS.5.1C introduces the requirement that development tools and other related information be
 4048 included in the list of configuration items and hence be subject to the CM requirements of CM
 4049 capabilities (ALC_CMC). Examples of development tools are programming languages and compilers.
 4050 Information pertaining to TOE generation items (such as compiler options, generation options, and
 4051 build options) is an example of information relating to development tools.

4052 **Developer action elements**

4053 **ALC_CMS.5.1D**

4054 The developer shall provide a configuration list for the TOE.

4055 **Content and presentation elements**

4056 **ALC_CMS.5.1C**

4057 The configuration list shall include the following: the TOE itself; the evaluation evidence required by
 4058 the SARs; the parts that comprise the TOE; the implementation representation; security flaw reports
 4059 and resolution status; **and development tools and related information.**

4060 **ALC_CMS.5.2C**

4061 The configuration list shall uniquely identify the configuration items.

4062 **ALC_CMS.5.3C**

4063 For each TSF relevant configuration item, the configuration list shall indicate the developer of the item.

4064 **Evaluator action elements**

4065 **ALC_CMS.5.1E**

4066 The evaluator shall confirm that the information provided meets all requirements for content and
 4067 presentation of evidence.

4068 **12.4 Delivery (ALC_DEL)**

4069 **12.4.1 Objectives**

4070 The concern of this family is the secure transfer of the finished TOE from the development
 4071 environment into the responsibility of the user.

4072 The requirements for delivery call for system control and distribution facilities and procedures that
 4073 detail the controls necessary to provide assurance that the security of the TOE is maintained during
 4074 distribution of the TOE to the user. For a valid distribution of the TOE, the procedures used for the

4075 distribution of the TOE address the implied or identified objectives identified in the PP/ST relating to
4076 the security of the TOE during delivery.

4077 **12.4.2 Component levelling**

4078 This family contains only one component. An increasing level of protection for the TOE is established
4079 by requiring that the delivery procedures are commensurate with the assumed attack potential in the
4080 family Vulnerability analysis (AVA_VAN) specified in the ST.

4081 **12.4.3 Application notes**

4082 Transfers from subcontractors to the developer or between different development sites are not
4083 considered here, but in the family Developer environment security (ALC_DVS).

4084 The end of the delivery phase is marked by the acceptance of the transfer of the TOE into the
4085 responsibility of the downstream user.

4086 NOTE: This does not necessarily coincide with the arrival of the TOE at the downstream user's
4087 location.

4088 The delivery procedures should consider, if applicable, issues such as:

- 4089 a) ensuring that the TOE received by the consumer corresponds precisely to the evaluated version of
4090 the TOE;
- 4091 b) avoiding or detecting any tampering with the actual version of the TOE;
- 4092 c) preventing submission of a counterfeit version of the TOE;
- 4093 d) avoiding unwanted knowledge of distribution of the TOE to the consumer: there might be cases
4094 where potential attackers should not know when and how it is delivered;
- 4095 e) avoiding or detecting the TOE being intercepted during delivery; and
- 4096 f) avoiding the TOE being delayed or stopped during distribution.

4097 The delivery procedures should include the recipient's actions implied by these issues. The consistent
4098 description of these implied actions is examined in the Preparative procedures (AGD_PRE) family, if
4099 present.

4100 **12.4.4 ALC_DEL.1 Delivery procedures**

4101 Dependencies: No dependencies.

4102 **Developer action elements**

4103 **ALC_DEL.1.1D**

4104 **The developer shall document and provide procedures for delivery of the TOE or parts of it to**
4105 **the consumer.**

4106 **ALC_DEL.1.2D**

4107 **The developer shall use the delivery procedures.**

4108 **Content and presentation elements**

4109 **ALC_DEL.1.1C**

4110 **The delivery documentation shall describe all procedures that are necessary to maintain**
 4111 **security when distributing versions of the TOE to the consumer.**

4112 **Evaluator action elements**

4113 **ALC_DEL.1.1E**

4114 **The evaluator shall confirm that the information provided meets all requirements for content**
 4115 **and presentation of evidence.**

4116 **12.5 Developer environment security (ALC_DVS)**

4117 **12.5.1 Objectives**

4118 Development security is concerned with the determination and specification of security controls
 4119 relating to the developer provided environment.

4120 NOTE: Such controls include coverage of security relevant aspects of asset management, human
 4121 resources security, physical and environmental security, communications and operations
 4122 management, access control, information systems acquisition, development and maintenance,
 4123 information security incident management, and business continuity management.

4124 **12.5.2 Component levelling**

4125 The components in this family are levelled on the basis of whether justification of the sufficiency of the
 4126 security controls is required.

4127 **12.5.3 Application notes**

4128 This family deals with controls to remove or reduce threads and security risks existing at the
 4129 developer's site.

4130 The evaluator should visit the site(s) in order to assess evidence for development security. This may
 4131 include sites of subcontractors involved in the TOE development and production. Any decision not to
 4132 visit shall be agreed with the evaluation authority.

4133 Although development security deals with the maintenance of the TOE and hence with aspects
 4134 becoming relevant after the completion of the evaluation, the Developer environment security
 4135 (ALC_DVS) requirements specify only that the development security controls be in place at the time of
 4136 evaluation. Furthermore, Developer environment security (ALC_DVS) does not contain any
 4137 requirements related to the sponsor's intention to apply the development security controls in the
 4138 future, after completion of the evaluation.

4139 It is recognised that confidentiality may not always be an issue for the protection of the TOE in its
 4140 development environment. The use of the word "necessary" allows for the selection of appropriate
 4141 safeguards.

4142 **12.5.4 ALC_DVS.1 Identification of security controls**

4143 Dependencies: No dependencies.

4144 **Developer action elements**

4145 **ALC_DVS.1.1D**

4146 **The developer shall produce and provide development security documentation.**

4147 **Content and presentation elements**

4148 **ALC_DVS.1.1C**

4149 **The development security documentation shall describe all the physical, logical, procedural,**
 4150 **personnel, and other security controls that are necessary to protect the confidentiality and**
 4151 **integrity of the TOE design and implementation in its development environment.**

4152 **Evaluator action elements**

4153 **ALC_DVS.1.1E**

4154 **The evaluator shall confirm that the information provided meets all requirements for content**
 4155 **and presentation of evidence.**

4156 **ALC_DVS.1.2E**

4157 **The evaluator shall confirm that the security controls are being applied.**

4158 **12.5.5 ALC_DVS.2 Sufficiency of security controls**

4159 Dependencies: No dependencies.

4160 **Developer action elements**

4161 **ALC_DVS.2.1D**

4162 **The developer shall produce and provide development security documentation.**

4163 **Content and presentation elements**

4164 **ALC_DVS.2.1C**

4165 **The development security documentation shall describe all the physical, procedural, personnel, and**
 4166 **other security controls that are necessary to protect the confidentiality and integrity of the TOE design**
 4167 **and implementation in its development environment.**

4168 **ALC_DVS.2.2C**

4169 **The development security documentation shall justify that the security controls provide the**
 4170 **necessary level of protection to maintain the confidentiality and integrity of the TOE.**

4171 **Evaluator action elements**

4172 **ALC_DVS.2.1E**

4173 **The evaluator shall confirm that the information provided meets all requirements for content and**
 4174 **presentation of evidence.**

4175 **ALC_DVS.2.2E**

4176 The evaluator shall confirm that the security controls are being applied.

4177 **12.6 Flaw remediation (ALC_FLR)**4178 **12.6.1 Objectives**

4179 Flaw remediation requires that discovered security flaws be tracked and corrected by the developer.
 4180 Although future compliance with flaw remediation procedures cannot be determined at the time of the
 4181 TOE evaluation, it is possible to evaluate the policies and procedures that a developer has in place to
 4182 track and correct flaws, and to distribute the flaw information and corrections.

4183 **12.6.2 Component levelling**

4184 The components in this family are levelled on the basis of the increasing extent in scope of the flaw
 4185 remediation procedures and the rigour of the flaw remediation policies.

4186 **12.6.3 Application notes**

4187 This family provides assurance that the TOE will be maintained and supported in the future, requiring
 4188 the TOE developer to track and correct flaws in the TOE. Additionally, requirements are included for
 4189 the distribution of flaw corrections. However, this family does not impose evaluation requirements
 4190 beyond the current evaluation.

4191 The TOE user is considered to be the focal point in the user organisation that is responsible for
 4192 receiving and implementing fixes to security flaws. This is not necessarily an individual user, but may
 4193 be an organisational representative who is responsible for the handling of security flaws. The use of
 4194 the term TOE user recognises that different organisations have different procedures for handling flaw
 4195 reporting, which may be done either by an individual user, or by a central administrative body.

4196 The flaw remediation procedures should describe the methods for dealing with all types of flaws
 4197 encountered. These flaws may be reported by the developer, by users of the TOE, or by other parties
 4198 with familiarity with the TOE. Some flaws may not be reparable immediately. There may be some
 4199 occasions where a flaw cannot be fixed and other (e.g. procedural) controls must be taken. The
 4200 documentation provided should cover the procedures for providing the operational sites with fixes,
 4201 and providing information on flaws where fixes are delayed (and what to do in the interim) or when
 4202 fixes are not possible.

4203 Changes applied to a TOE after its release render it unevaluated; although some information from the
 4204 original evaluation may still apply. The phrase “release of the TOE” used in this family therefore refers
 4205 to a version of a product that is a release of a certified TOE, to which changes have been applied.

4206 **12.6.4 ALC_FLR.1 Basic flaw remediation**

4207 Dependencies: No dependencies.

4208 **Developer action elements**4209 **ALC_FLR.1.1D**

4210 **The developer shall document and provide flaw remediation procedures addressed to TOE**
 4211 **developers.**

4212 **Content and presentation elements**

4213 **ALC_FLR.1.1C**

4214 **The flaw remediation procedures documentation shall describe the procedures used to track**
4215 **all reported security flaws in each release of the TOE.**

4216 **ALC_FLR.1.2C**

4217 **The flaw remediation procedures shall require that a description of the nature and effect of**
4218 **each security flaw be provided, as well as the status of finding a correction to that flaw.**

4219 **ALC_FLR.1.3C**

4220 **The flaw remediation procedures shall require that corrective actions be identified for each of**
4221 **the security flaws.**

4222 **ALC_FLR.1.4C**

4223 **The flaw remediation procedures documentation shall describe the methods used to provide**
4224 **flaw information, corrections and guidance on corrective actions to TOE users.**

4225 **Evaluator action elements**

4226 **ALC_FLR.1.1E**

4227 **The evaluator shall confirm that the information provided meets all requirements for content**
4228 **and presentation of evidence.**

4229 **12.6.5 ALC_FLR.2 Flaw reporting procedures**

4230 Dependencies: No dependencies.

4231 **Objectives**

4232 In order for the developer to be able to act appropriately upon security flaw reports from TOE users,
4233 and to know to whom to send corrective fixes, TOE users need to understand how to submit security
4234 flaw reports to the developer. Flaw remediation guidance from the developer to the TOE user ensures
4235 that TOE users are aware of this important information.

4236 **Developer action elements**

4237 **ALC_FLR.2.1D**

4238 The developer shall document and provide flaw remediation procedures addressed to TOE developers.

4239 **ALC_FLR.2.2D**

4240 **The developer shall establish a procedure for accepting and acting upon all reports of security**
4241 **flaws and requests for corrections to those flaws.**

4242 **ALC_FLR.2.3D**

4243 **The developer shall provide flaw remediation guidance addressed to TOE users.**

4244 **Content and presentation elements**

4245 **ALC_FLR.2.1C**

4246 The flaw remediation procedures documentation shall describe the procedures used to track all
4247 reported security flaws in each release of the TOE.

4248 **ALC_FLR.2.2C**

4249 The flaw remediation procedures shall require that a description of the nature and effect of each
4250 security flaw be provided, as well as the status of finding a correction to that flaw.

4251 **ALC_FLR.2.3C**

4252 The flaw remediation procedures shall require that corrective actions be identified for each of the
4253 security flaws.

4254 **ALC_FLR.2.4C**

4255 The flaw remediation procedures documentation shall describe the methods used to provide flaw
4256 information, corrections and guidance on corrective actions to TOE users.

4257 **ALC_FLR.2.5C**

4258 **The flaw remediation procedures shall describe a means by which the developer receives from**
4259 **TOE users reports and enquiries of suspected security flaws in the TOE.**

4260 **ALC_FLR.2.6C**

4261 **The procedures for processing reported security flaws shall ensure that any reported flaws are**
4262 **remediated and the remediation procedures issued to TOE users.**

4263 **ALC_FLR.2.7C**

4264 **The procedures for processing reported security flaws shall provide safeguards that any**
4265 **corrections to these security flaws do not introduce any new flaws.**

4266 **ALC_FLR.2.8C**

4267 **The flaw remediation guidance shall describe a means by which TOE users report to the**
4268 **developer any suspected security flaws in the TOE.**

4269 **Evaluator action elements**

4270 **ALC_FLR.2.1E**

4271 The evaluator shall confirm that the information provided meets all requirements for content and
4272 presentation of evidence.

4273 **12.6.6 ALC_FLR.3 Systematic flaw remediation**

4274 Dependencies: No dependencies.

4275 **Objectives**

4276 In order for the developer to be able to act appropriately upon security flaw reports from TOE users,
 4277 and to know to whom to send corrective fixes, TOE users need to understand how to submit security
 4278 flaw reports to the developer, and how to register themselves with the developer so that they may
 4279 receive these corrective fixes. Flaw remediation guidance from the developer to the TOE user ensures
 4280 that TOE users are aware of this important information.

4281 **Developer action elements**

4282 **ALC_FLR.3.1D**

4283 The developer shall document and provide flaw remediation procedures addressed to TOE developers.

4284 **ALC_FLR.3.2D**

4285 The developer shall establish a procedure for accepting and acting upon all reports of security flaws
 4286 and requests for corrections to those flaws.

4287 **ALC_FLR.3.3D**

4288 The developer shall provide flaw remediation guidance addressed to TOE users.

4289 **Content and presentation elements**

4290 **ALC_FLR.3.1C**

4291 The flaw remediation procedures documentation shall describe the procedures used to track all
 4292 reported security flaws in each release of the TOE.

4293 **ALC_FLR.3.2C**

4294 The flaw remediation procedures shall require that a description of the nature and effect of each
 4295 security flaw be provided, as well as the status of finding a correction to that flaw.

4296 **ALC_FLR.3.3C**

4297 The flaw remediation procedures shall require that corrective actions be identified for each of the
 4298 security flaws.

4299 **ALC_FLR.3.4C**

4300 The flaw remediation procedures documentation shall describe the methods used to provide flaw
 4301 information, corrections and guidance on corrective actions to TOE users.

4302 **ALC_FLR.3.5C**

4303 The flaw remediation procedures shall describe a means by which the developer receives from TOE
 4304 users reports and enquiries of suspected security flaws in the TOE.

4305 **ALC_FLR.3.6C**

4306 **The flaw remediation procedures shall include a procedure requiring timely response and the**
 4307 **automatic distribution of security flaw reports and the associated corrections to registered**
 4308 **users who might be affected by the security flaw.**

4309 **ALC_FLR.3.7C**

4310 The procedures for processing reported security flaws shall ensure that any reported flaws are
 4311 remediated and the remediation procedures issued to TOE users.

4312 **ALC_FLR.3.8C**

4313 The procedures for processing reported security flaws shall provide safeguards that any corrections to
 4314 these security flaws do not introduce any new flaws.

4315 **ALC_FLR.3.9C**

4316 The flaw remediation guidance shall describe a means by which TOE users report to the developer any
 4317 suspected security flaws in the TOE.

4318 **ALC_FLR.3.10C**

4319 **The flaw remediation guidance shall describe a means by which TOE users may register with**
 4320 **the developer, to be eligible to receive security flaw reports and corrections.**

4321 **ALC_FLR.3.11C**

4322 **The flaw remediation guidance shall identify the specific points of contact for all reports and**
 4323 **enquiries about security issues involving the TOE.**

4324 **Evaluator action elements**4325 **ALC_FLR.3.1E**

4326 The evaluator shall confirm that the information provided meets all requirements for content and
 4327 presentation of evidence.

4328 **12.7 Development Life-cycle definition (ALC_LCD)**4329 **12.7.1 Objectives**

4330 Poorly defined or uncontrolled processes applied during the development, production and
 4331 maintenance of the TOE can result in a TOE that does not meet all of its security objectives. Therefore,
 4332 it is important that well defined and controlled processes be established as early as possible in the
 4333 TOE's life-cycle.

4334 Defining and implementing such processes does not guarantee that the TOE meets all of its SFRs. It is
 4335 possible that the processes will be insufficient or inadequate.

4336 Adopting a life-cycle model, or models that meets the needs of the developer's organization will
 4337 improve the likelihood that the development, production and maintenance processes applied to TOE
 4338 support the correct design and implementation of a TOE that meets the specified SFRs.

4339 The determination of appropriate process controls in order to support process improvement is a long
 4340 established best practice.

4341 **12.7.2 Component levelling**

4342 The components in this family are levelled on the basis of increasing requirements for measurability of
 4343 the life-cycle model, and for compliance with that model.

4344 12.7.3 Application notes

4345 A life-cycle model encompasses the procedures, tools and techniques used to develop and maintain the
 4346 TOE. Aspects of the process that may be covered by such a model include design methods, review
 4347 procedures, project management controls, change control procedures, test methods and acceptance
 4348 procedures. An effective life-cycle model will address these aspects of the development and
 4349 maintenance process within an overall management structure that assigns responsibilities and
 4350 monitors progress.

4351 There are different types of acceptance situations that are dealt with at different locations in the
 4352 criteria:

- 4353 • acceptance of parts delivered by subcontractors (“integration”) should be treated in this family
- 4354 • Development Life-cycle definition (ALC_LCD),
- 4355 • acceptance subsequent to internal transportations in Developer environment security
 4356 (ALC_DVS),
- 4357 • acceptance of parts into the CM system in CM capabilities (ALC_CMC), and
- 4358 • acceptance of the delivered TOE by the consumer in Delivery (ALC_DEL).

4359 The first three types may overlap.

4360 Although life-cycle definition deals with the maintenance of the TOE and hence with aspects becoming
 4361 relevant after the completion of the evaluation, its evaluation adds assurance through an analysis of
 4362 the life-cycle information for the TOE provided at the time of the evaluation.

4363 A life-cycle model provides for the necessary control over the development and maintenance of the
 4364 TOE, if the model enables sufficient minimisation of the danger that the TOE will not meet its security
 4365 requirement.

4366 A measurable life-cycle model is a model using some quantitative valuation (arithmetic parameters
 4367 and/or metrics) of the managed product in order to measure development properties of the product.
 4368 Typical metrics are source code complexity metrics, defect density (errors per size of code) or mean
 4369 time to failure. For the security evaluation all those metrics are of relevance, which are used to
 4370 increase quality by decreasing the probability of faults and thereby in turn increasing assurance in the
 4371 security of the TOE.

4372 One should take into account that there exist standardised life-cycle models on the one hand (like the
 4373 waterfall model) and standardised metrics on the other hand (like error density), which may be
 4374 combined. The ISO/IEC 15408 series does not require the life-cycle to follow exactly one standard
 4375 defining both aspects.

4376 12.7.4 ALC_LCD.1 Developer defined life-cycle processes

4377 Dependencies: No dependencies.

4378 **Developer action elements**

4379 **ALC_LCD.1.1D**

4380 **The developer shall establish a life-cycle model to be used in the development and**
4381 **maintenance of the TOE.**

4382 **ALC_LCD.1.2D**

4383 **The developer shall provide life-cycle definition documentation.**

4384 **Content and presentation elements**

4385 **ALC_LCD.1.1C**

4386 **The life-cycle definition documentation shall describe the processes used to develop and**
4387 **maintain the TOE.**

4388 **ALC_LCD.1.2C**

4389 **The life-cycle model shall provide for the necessary control over the development and**
4390 **maintenance of the TOE.**

4391 **Evaluator action elements**

4392 **ALC_LCD.1.1E**

4393 **The evaluator shall confirm that the information provided meets all requirements for content**
4394 **and presentation of evidence.**

4395 **12.7.5 ALC_LCD.2 Measurable life-cycle model**

4396 Dependencies: No dependencies.

4397 **Developer action elements**

4398 **ALC_LCD.2.1D**

4399 **The developer shall establish a life-cycle model to be used in the development and maintenance of the**
4400 **TOE that is based on a measurable life-cycle model.**

4401 **ALC_LCD.2.2D**

4402 **The developer shall provide life-cycle definition documentation.**

4403 **ALC_LCD.2.3D**

4404 **The developer shall measure the TOE development using the measurable life-cycle model..**

4405 **ALC_LCD.2.4D**

4406 **The developer shall provide life-cycle output documentation.**

4407 **Content and presentation elements**

4408 **ALC_LCD.2.1C**

4409 The life-cycle definition documentation shall describe the model used to develop and maintain the TOE
 4410 **including the details of its arithmetic parameters and/or metrics used to measure the quality of**
 4411 **the TOE and/or its development.**

4412 **ALC_LCD.2.2C**

4413 The life-cycle model shall provide for the necessary control over the development and maintenance of
 4414 the TOE.

4415 **ALC_LCD.2.3C**

4416 **The life-cycle output documentation shall provide the results of the measurements of the TOE**
 4417 **development using the measurable life-cycle model.**

4418 **Evaluator action elements**

4419 **ALC_LCD.2.1E**

4420 The evaluator shall confirm that the information provided meets all requirements for content and
 4421 presentation of evidence.

4422 **ALC_LCD.2.2E**

4423 **The evaluator shall confirm that the measurements of the TOE development processes and**
 4424 **security relevant properties of the TOE support improvements in the development processes**
 4425 **and/or the TOE itself.**

4426 **12.8 TOE Development Artifacts (ALC_TDA)**

4427 **12.8.1 Objectives**

4428 This family aims to add trust to the development process or a development. It focuses on the
 4429 generation of certain artifacts in the development process. These artifacts are used at a later point in
 4430 time to assess the degree to which the development process is trustable. This trust is realized through
 4431 the validation of the generated artifacts for confirming them as sufficient evidence for trustable
 4432 development.

4433 This family introduces developer practices within the development process to generate the required
 4434 artifacts for realizing trustable development. If a requirement in this family does not explicitly specify
 4435 the use of automation to generate the required artifacts, the developer is free to undertake the
 4436 corresponding practice manually, or to use some integrated automation in the development process,
 4437 or to use a hybrid method of both. It is expected that the degree of trust in the development process is
 4438 proportional to the degree of automation adoption to implement the corresponding practice in the
 4439 development process.

4440 This family also has a relationship with the ALC_TAT family. As ALC_TAT focuses on the tools and
 4441 techniques aspect for developing, analysing, and implementing the TOE, it provides the necessary
 4442 context when describing the practices of this family being introduced into the development process.

4443 **12.8.2 Component levelling**

4444 The components in this family are levelled on the basis of increasing cross-checking for consistency
4445 with relevant evidence from components of other families of other security assurance classes.

4446 **12.8.3 Application notes**

4447 The requirements in ALC_TDA.1 provide a degree of trust in the developer's ability to identify the set
4448 of implementation representation which actually has been used during the TOE generation time. This
4449 degree of trust helps to positively answer the question "is that really the source code for this software"
4450 or "is that really the register-transfer level (RTL) design or description for this integrated circuit
4451 hardware"" or "is that really the set of implementation representation for this TOE", which is
4452 potentially relevant in an evaluation. Such degree of trust is built on

4453 a) the timing of when the set of implementation representation identifiers is recorded or logged,

4454 b) the integrity and authenticity of the record of implementation representation identifiers, and

4455 c) the traceability of implementation representation identifiers from the TOE.

4456 In the case where some implementation representation elements are also covered in the configuration
4457 list due to ALC_CMS.3, the requirements in ALC_TDA.2 make sure that these implementation
4458 representation elements actually are identifiable through the use of the implementation
4459 representation identifiers of ALC_TDA.1.

4460 With the accurate recording or logging of the actual implementation representation being used by the
4461 development tools under the scope of ALC_TAT, it provides an additional evidence to convince a third
4462 party that a regeneration of the TOE is functionally equivalent to the original TOE.

4463 The requirements in ALC_TDA.3 provide the developer an opportunity to testify the absence of
4464 functional differences between the two possibly visibly different TOEs which have been independently
4465 generated from the identical set of implementation representation.

4466 **12.8.4 ALC_TDA.1 Uniquely identifying implementation representation**

4467 Dependencies: No dependencies.

4468 **Developer action elements**

4469 **ALC_TDA.1.1D**

4470 **The developer shall identify individual elements of the TOE implementation representation to**
4471 **record the list of unique TOE implementation representation identifiers, as the development**
4472 **tool generates the TOE.**

4473 **ALC_TDA.1.2D**

4474 **The developer shall use the current date and time to timestamp the list of unique TOE**
4475 **implementation representation identifiers as recorded during the TOE generation time.**

4476 **ALC_TDA.1.3D**

4477 **The developer shall maintain the integrity of the list of unique TOE implementation**
4478 **representation identifiers as recorded during the TOE generation time.**

4479 **ALC_TDA.1.4D**

4480 **The developer shall ensure the authenticity of the list of unique TOE implementation**
 4481 **representation identifiers as recorded during the TOE generation time, with the maintenance**
 4482 **of the (author) origination information.**

4483 **ALC_TDA.1.5D**

4484 **The developer shall be able to trace from the TOE to the list of unique TOE implementation**
 4485 **representation identifiers as recorded during the TOE generation time.**

4486 **ALC_TDA.1.6D**

4487 **The developer shall produce and provide documentation describing**

4488 **a) the developer's creation of the list of unique TOE implementation representation**
 4489 **identifiers as recorded during the TOE generation time;**

4490 **b) the developer's timestamp being applied to the list of unique TOE implementation**
 4491 **representation identifiers as recorded during the TOE generation time;**

4492 **c) the maintenance of the (author) origination information of the list of unique TOE**
 4493 **implementation representation identifiers as recorded during the TOE generation time;**

4494 **d) the maintenance of the integrity of the list of unique TOE implementation**
 4495 **representation identifiers as recorded during the TOE generation time and its**
 4496 **associated timestamp and (author) origination information;**

4497 **e) the developer's mechanism to trace from the TOE to the list of unique TOE**
 4498 **implementation representation identifiers as recorded during the TOE generation time.**

4499 **Content and presentation elements**4500 **ALC_TDA.1.1C**

4501 **The list of unique TOE implementation representation identifiers as recorded during the TOE**
 4502 **generation time shall demonstrate the correspondence between the TOE implementation**
 4503 **representation element identifiers and the TOE implementation representation element**
 4504 **names.**

4505 **ALC_TDA.1.2C**

4506 **The TOE implementation representation element names shall be in the same form as used or**
 4507 **referenced by the development tool to generate the TOE.**

4508 **ALC_TDA.1.3C**

4509 **The timestamp of the list of unique TOE implementation representation identifiers as recorded**
 4510 **during the TOE generation time shall be consistent with the creation time of the TOE.**

4511 **ALC_TDA.1.4C**

4512 **The (author) origination information of the list of unique TOE implementation representation**
 4513 **identifiers as recorded during the TOE generation time shall be consistent with the (author)**

- 4514 **origination information of the TOE. The author origination information may be the name of an**
 4515 **affiliate of an organization.**
- 4516 **Evaluator action elements**
- 4517 **ALC_TDA.1.1E**
- 4518 **The evaluator shall confirm that the information provided meets all requirements for content**
 4519 **and presentation of evidence.**
- 4520 **ALC_TDA.1.2E**
- 4521 **The evaluator shall confirm that the development tool for generating the TOE is capable to use**
 4522 **or reference the implementation representation element names.**
- 4523 **ALC_TDA.1.3E**
- 4524 **The evaluator shall confirm that the list of unique TOE implementation representation**
 4525 **identifiers as recorded during the TOE generation time is consistent with the creation time of**
 4526 **the TOE.**
- 4527 **ALC_TDA.1.4E**
- 4528 **The evaluator shall confirm that the (author) origination information of the list of unique TOE**
 4529 **implementation representation identifiers as recorded during the TOE generation time is**
 4530 **consistent with the (author) origination information of the TOE.**
- 4531 **ALC_TDA.1.5E**
- 4532 **The evaluator shall check the integrity of the list of unique TOE implementation representation**
 4533 **identifiers as recorded during the TOE generation time and its associated timestamp and**
 4534 **(author) origination information.**
- 4535 **ALC_TDA.1.6E**
- 4536 **The evaluator shall confirm the developer's ability to trace from the TOE to the list of unique**
 4537 **TOE implementation representation identifiers as recorded during the TOE generation time.**
- 4538 **12.8.5 ALC_TDA.2 Matching CMS scope of implementation representation**
- 4539 Dependencies: ALC_CMS.3 Implementation representation CM coverage
- 4540 **Developer action elements**
- 4541 **ALC_TDA.2.1D**
- 4542 **The developer shall identify individual elements of the TOE implementation representation to record**
 4543 **the list of unique TOE implementation representation identifiers, as the development tool generates**
 4544 **the TOE.**
- 4545 **ALC_TDA.2.2D**
- 4546 **The developer shall use the current date and time to timestamp the list of unique TOE implementation**
 4547 **representation identifiers as recorded during the TOE generation time.**

4548 **ALC_TDA.2.3D**

4549 The developer shall maintain the integrity of the list of unique TOE implementation representation
4550 identifiers as recorded during the TOE generation time.

4551 **ALC_TDA.2.4D**

4552 The developer shall ensure the authenticity of the list of unique TOE implementation representation
4553 identifiers as recorded during the TOE generation time, with the maintenance of the (author)
4554 origination information.

4555 **ALC_TDA.2.5D**

4556 The developer shall be able to trace from the TOE to the list of unique TOE implementation
4557 representation identifiers as recorded during the TOE generation time.

4558 **ALC_TDA.2.6D**

4559 The developer shall produce and provide documentation describing

- 4560 a) the developer's creation of the list of unique TOE implementation representation identifiers as
4561 recorded during the TOE generation time;
- 4562 b) the developer's timestamp being applied to the list of unique TOE implementation
4563 representation identifiers as recorded during the TOE generation time;
- 4564 c) the maintenance of the (author) origination information of the list of unique TOE
4565 implementation representation identifiers as recorded during the TOE generation time;
- 4566 d) the maintenance of the integrity of the list of unique TOE implementation representation
4567 identifiers as recorded during the TOE generation time and its associated timestamp and
4568 (author) origination information;
- 4569 e) the developer's mechanism to trace from the TOE to the list of unique TOE implementation
4570 representation identifiers as recorded during the TOE generation time.

4571 **ALC_TDA.2.7D**

4572 **The developer shall provide evidence that the elements of implementation representation**
4573 **under the configuration scope of ALC_CMS.3 are identified by the list of unique TOE**
4574 **implementation representation identifiers as recorded during the TOE generation time.**

4575 **Content and presentation elements**4576 **ALC_TDA.2.1C**

4577 The list of unique TOE implementation representation identifiers as recorded during the TOE
4578 generation time shall demonstrate the correspondence between the TOE implementation
4579 representation element identifiers and the TOE implementation representation element names.

4580 **ALC_TDA.2.2C**

4581 The TOE implementation representation element names shall be in the same form as used or
4582 referenced by the development tool to generate the TOE.

4583 **ALC_TDA.2.3C**

4584 The timestamp of the list of unique TOE implementation representation identifiers as recorded during
4585 the TOE generation time shall be consistent with the creation time of the TOE.

4586 **ALC_TDA.2.4C**

4587 The (author) origination information of the list of unique TOE implementation representation
 4588 identifiers as recorded during the TOE generation time shall be consistent with the (author)
 4589 origination information of the TOE. The author origination information may be the name of an affiliate
 4590 of an organization.

4591 **ALC_TDA.2.5C**

4592 **The list of identifiers of the elements of implementation representation under the**
 4593 **configuration scope of ALC_CMS.3 shall match with the list of unique TOE implementation**
 4594 **representation identifiers as recorded during the TOE generation time.**

4595 **Evaluator action elements**4596 **ALC_TDA.2.1E**

4597 The evaluator shall confirm that the information provided meets all requirements for content and
 4598 presentation of evidence.

4599 **ALC_TDA.2.2E**

4600 The evaluator shall confirm that the development tool for generating the TOE is capable to use or
 4601 reference the implementation representation element names.

4602 **ALC_TDA.2.3E**

4603 The evaluator shall confirm that the list of unique TOE implementation representation identifiers as
 4604 recorded during the TOE generation time is consistent with the creation time of the TOE.

4605 **ALC_TDA.2.4E**

4606 The evaluator shall confirm that the (author) origination information of the list of unique TOE
 4607 implementation representation identifiers as recorded during the TOE generation time is consistent
 4608 with the (author) origination information of the TOE.

4609 **ALC_TDA.2.5E**

4610 The evaluator shall check the integrity of the list of unique TOE implementation representation
 4611 identifiers as recorded during the TOE generation time and its associated timestamp and (author)
 4612 origination information.

4613 **ALC_TDA.2.6E**

4614 The evaluator shall confirm the developer's ability to trace from the TOE to the list of unique TOE
 4615 implementation representation identifiers as recorded during the TOE generation time.

4616 **ALC_TDA.2.7E**

4617 **The evaluator shall confirm that the list of identifiers of the elements of implementation**
 4618 **representation under the configuration scope of ALC_CMS.3 matches with the list of unique TOE**
 4619 **implementation representation identifiers as recorded during the TOE generation time.**

4620 **12.8.6 ALC_TDA.3 Regenerate TOE with well-defined development tools**

4621 Dependencies: ALC_CMS.3 Implementation representation CM coverage

4622 ALC_TAT.1 Well-defined development tools and

4623 ADV_IMP.1 Implementation representation of the TSF

4624 **Developer action elements**

4625 **ALC_TDA.3.1D**

4626 The developer shall identify individual elements of the TOE implementation representation to record
4627 the list of unique TOE implementation representation identifiers, as the development tool generates
4628 the TOE.

4629 **ALC_TDA.3.2D**

4630 The developer shall use the current date and time to timestamp the list of unique TOE implementation
4631 representation identifiers as recorded during the TOE generation time.

4632 **ALC_TDA.3.3D**

4633 The developer shall maintain the integrity of the list of unique TOE implementation representation
4634 identifiers as recorded during the TOE generation time.

4635 **ALC_TDA.3.4D**

4636 The developer shall ensure the authenticity of the list of unique TOE implementation representation
4637 identifiers as recorded during the TOE generation time, with the maintenance of the (author)
4638 origination information.

4639 **ALC_TDA.3.5D**

4640 The developer shall be able to trace from the TOE to the list of unique TOE implementation
4641 representation identifiers as recorded during the TOE generation time.

4642 **ALC_TDA.3.6D**

4643 The developer shall produce and provide documentation describing

- 4644 a) the developer's creation of the list of unique TOE implementation representation identifiers as
4645 recorded during the TOE generation time;
- 4646 b) the developer's timestamp being applied to the list of unique TOE implementation
4647 representation identifiers as recorded during the TOE generation time;
- 4648 c) the maintenance of the (author) origination information of the list of unique TOE
4649 implementation representation identifiers as recorded during the TOE generation time;
- 4650 d) the maintenance of the integrity of the list of unique TOE implementation representation
4651 identifiers as recorded during the TOE generation time and its associated timestamp and
4652 (author) origination information;
- 4653 e) the developer's mechanism to trace from the TOE to the list of unique TOE implementation
4654 representation identifiers as recorded during the TOE generation time.

4655 **ALC_TDA.3.7D**

4656 The developer shall provide evidence that the elements of implementation representation under the
4657 configuration scope of ALC_CMS.3 are identified by the list of unique TOE implementation
4658 representation identifiers as recorded during the TOE generation time.

4659 **ALC_TDA.3.8D**

4660 **After applying the development tools to another copy of the TOE implementation**
4661 **representation according to the list of unique TOE implementation representation identifiers**
4662 **to regenerate a TOE copy, the developer shall explain the functional differences, if any, between**
4663 **the TOE copy and the original TOE.**

4664 **ALC_TDA.3.9D**

4665 **The developer shall produce and provide documentation explaining the functional differences,**
4666 **if any, between the regenerated TOE copy and the original TOE.**

4667 **Content and presentation elements**

4668 **ALC_TDA.3.1C**

4669 The list of unique TOE implementation representation identifiers as recorded during the TOE
4670 generation time shall demonstrate the correspondence between the TOE implementation
4671 representation element identifiers and the TOE implementation representation element names.

4672 **ALC_TDA.3.2C**

4673 The TOE implementation representation element names shall be in the same form as used or
4674 referenced by the development tool to generate the TOE.

4675 **ALC_TDA.3.3C**

4676 The timestamp of the list of unique TOE implementation representation identifiers as recorded during
4677 the TOE generation time shall be consistent with the creation time of the TOE.

4678 **ALC_TDA.3.4C**

4679 The (author) origination information of the list of unique TOE implementation representation
4680 identifiers as recorded during the TOE generation time shall be consistent with the (author)
4681 origination information of the TOE. The author origination information may be the name of an affiliate
4682 of an organization.

4683 **ALC_TDA.3.5C**

4684 The list of identifiers of the elements of implementation representation under the configuration scope
4685 of ALC_CMS.3 shall match with the list of unique TOE implementation representation identifiers as
4686 recorded during the TOE generation time.

4687 **ALC_TDA.3.6C**

4688 **The developer's explanation of the functional differences, if any, between the regenerated TOE**
 4689 **copy and the original TOE shall take into account all visible differences, if any, between the**
 4690 **regenerated TOE copy and the original TOE.**

4691 **Evaluator action elements**4692 **ALC_TDA.3.1E**

4693 The evaluator shall confirm that the information provided meets all requirements for content and
 4694 presentation of evidence.

4695 **ALC_TDA.3.2E**

4696 The evaluator shall confirm that the development tool for generating the TOE is capable to use or
 4697 reference the implementation representation element names.

4698 **ALC_TDA.3.3E**

4699 The evaluator shall confirm that the list of unique TOE implementation representation identifiers as
 4700 recorded during the TOE generation time is consistent with the creation time of the TOE.

4701 **ALC_TDA.3.4E**

4702 The evaluator shall confirm that the (author) origination information of the list of unique TOE
 4703 implementation representation identifiers as recorded during the TOE generation time is consistent
 4704 with the (author) origination information of the TOE.

4705 **ALC_TDA.3.5E**

4706 The evaluator shall check the integrity of the list of unique TOE implementation representation
 4707 identifiers as recorded during the TOE generation time and its associated timestamp and (author)
 4708 origination information.

4709 **ALC_TDA.3.6E**

4710 The evaluator shall confirm the developer's ability to trace from the TOE to the list of unique TOE
 4711 implementation representation identifiers as recorded during the TOE generation time.

4712 **ALC_TDA.3.7E**

4713 The evaluator shall confirm that the list of identifiers of the elements of implementation
 4714 representation under the configuration scope of ALC_CMS.3 matches with the list of unique TOE
 4715 implementation representation identifiers as recorded during the TOE generation time.

4716 **ALC_TDA.3.8E**

4717 **The evaluator shall check that the developer's explanation of the functional differences, if any,**
 4718 **between the regenerated TOE copy and the original TOE takes into account all visible**
 4719 **differences, if any, between the regenerated TOE copy and the original TOE.**

4720 **12.9 Tools and techniques (ALC_TAT)**

4721 **12.9.1 Objectives**

4722 Tools and techniques is an aspect of selecting tools that are used to develop, analyse and implement
 4723 the TOE. It includes requirements to prevent ill-defined, inconsistent or incorrect development tools
 4724 from being used to develop the TOE. This includes, but is not limited to, programming languages,
 4725 documentation, implementation standards, and other parts of the TOE such as supporting runtime
 4726 libraries.

4727 **12.9.2 Component levelling**

4728 The components in this family are levelled on the basis of increasing requirements on the description
 4729 and scope of the implementation standards and the documentation of implementation-dependent
 4730 options.

4731 **12.9.3 Application notes**

4732 There is a requirement for well-defined development tools. These are tools that are clearly and
 4733 completely described. For example, programming languages and computer aided design (CAD)
 4734 systems that are based on a standard published by standards bodies are considered to be well-defined.
 4735 Self-made tools would need further investigation to clarify whether they are well-defined.

4736 The requirement in ALC_TAT.1.2C is especially applicable to programming languages so as to ensure
 4737 that all statements in the source code have an unambiguous meaning.

4738 In ALC_TAT.2 and ALC_TAT.3, implementation guidelines may be accepted as an implementation
 4739 standard if they have been approved by some group of experts (e.g. academic experts, standards
 4740 bodies). Implementation standards are normally public, well accepted and common practise in a
 4741 specific industry, but developer-specific implementation guidelines may also be accepted as a
 4742 standard; the emphasis is on the expertise.

4743 Tools and techniques distinguishes between the implementation standards applied by the developer
 4744 (ALC_TAT.2.3D) and the implementation standards for “all parts of the TOE” (ALC_TAT.3.3D) which
 4745 include third party software, hardware, or firmware. The configuration list introduced in CM scope
 4746 (ALC_CMS) requires that for each TSF relevant configuration item to indicate if it has been generated
 4747 by the TOE developer or by third party developers

4748 **12.9.4 ALC_TAT.1 Well-defined development tools**

4749 Dependencies: ADV_IMP.1 Implementation representation of the TSF

4750 **Developer action elements**

4751 **ALC_TAT.1.1D**

4752 **The developer shall provide the documentation identifying each development tool being used**
 4753 **for the TOE.**

4754 **ALC_TAT.1.2D**

4755 **The developer shall document and provide the selected implementation-dependent options of**
 4756 **each development tool.**

4757 **Content and presentation elements**

4758 **ALC_TAT.1.1C**

4759 **Each development tool used for implementation shall be well-defined.**

4760 **ALC_TAT.1.2C**

4761 **The documentation of each development tool shall unambiguously define the meaning of all**
 4762 **statements as well as all conventions and directives used in the implementation.**

4763 **ALC_TAT.1.3C**

4764 **The documentation of each development tool shall unambiguously define the meaning of all**
 4765 **implementation-dependent options.**

4766 **Evaluator action elements**

4767 **ALC_TAT.1.1E**

4768 **The evaluator shall confirm that the information provided meets all requirements for content**
 4769 **and presentation of evidence.**

4770 **12.9.5 ALC_TAT.2 Compliance with implementation standards**

4771 Dependencies: ADV_IMP.1 Implementation representation of the TSF

4772 **Developer action elements**

4773 **ALC_TAT.2.1D**

4774 **The developer shall provide the documentation identifying each development tool being used for the**
 4775 **TOE.**

4776 **ALC_TAT.2.2D**

4777 **The developer shall document and provide the selected implementation-dependent options of each**
 4778 **development tool.**

4779 **ALC_TAT.2.3D**

4780 **The developer shall describe and provide the implementation standards that are being applied**
 4781 **by the developer.**

4782 **Content and presentation elements**

4783 **ALC_TAT.2.1C**

4784 **Each development tool used for implementation shall be well-defined.**

4785 **ALC_TAT.2.2C**

4786 **The documentation of each development tool shall unambiguously define the meaning of all**
 4787 **statements as well as all conventions and directives used in the implementation.**

4788 **ALC_TAT.2.3C**

4789 The documentation of each development tool shall unambiguously define the meaning of all
 4790 implementation-dependent options.

4791 **Evaluator action elements**4792 **ALC_TAT.2.1E**

4793 The evaluator shall confirm that the information provided meets all requirements for content and
 4794 presentation of evidence.

4795 **ALC_TAT.2.2E**

4796 **The evaluator shall confirm that the implementation standards have been applied.**

4797 **12.9.6 ALC_TAT.3 Compliance with implementation standards - all parts**

4798 Dependencies: ADV_IMP.1 Implementation representation of the TSF

4799 **Developer action elements**4800 **ALC_TAT.3.1D**

4801 The developer shall provide the documentation identifying each development tool being used for the
 4802 TOE.

4803 **ALC_TAT.3.2D**

4804 The developer shall document and provide the selected implementation-dependent options of each
 4805 development tool.

4806 **ALC_TAT.3.3D**

4807 The developer shall describe and provide the implementation standards that are being applied by the
 4808 developer **and by any third-party providers for all parts of the TOE.**

4809 **Content and presentation elements**4810 **ALC_TAT.3.1C**

4811 Each development tool used for implementation shall be well-defined.

4812 **ALC_TAT.3.2C**

4813 The documentation of each development tool shall unambiguously define the meaning of all
 4814 statements as well as all conventions and directives used in the implementation.

4815 **ALC_TAT.3.3C**

4816 The documentation of each development tool shall unambiguously define the meaning of all
 4817 implementation-dependent options.

4818 **Evaluator action elements**

4819 **ALC_TAT.3.1E**

4820 The evaluator shall confirm that the information provided meets all requirements for content and
4821 presentation of evidence.

4822 **ALC_TAT.3.2E**

4823 **The evaluator shall confirm that the implementation standards have been applied.**

4824 **12.10 Integration of composition parts and consistency check of delivery procedures**
4825 **(ALC_COMP)**

4826 **12.10.1 Objectives**

4827 The aim of this family is to determine whether

- 4828 - the correct version of the dependent component is installed onto / embedded into the correct
4829 version of the related base component, and
- 4830 - the preparative guidance procedures of the base component developer and the dependent
4831 component developer are compatible with the acceptance procedures of the composite
4832 product integrator.

4833 **12.10.2 Component levelling**

4834 This family contains only one component.

4835 **12.10.3 Application notes**

4836 The composite product evaluator shall verify that the correct version of the dependent component
4837 under evaluation has been installed onto / embedded into the evaluated version of the related base
4838 component of the composite product.

4839 The composite product evaluation sponsor shall ensure that appropriate evidence generated by the
4840 composite product integrator is available for the composite product evaluator. This evidence may
4841 include, amongst other, the configuration list of the base component developer (e.g. provided within
4842 his acknowledgement statement).

4843 The composite product evaluator shall verify that the delivery procedures of the base component
4844 developer and the dependent component developer are compatible with the acceptance procedures
4845 used by the composite product integrator.

4846 The composite product evaluator shall verify that all configuration parameters prescribed by the base
4847 component developer and the dependent component developer (e.g. pre-personalization data, pre-
4848 personalisation scripts) are used by the composite product integrator.

4849 The composite product evaluation sponsor shall ensure that appropriate evidence generated by the
4850 composite product integrator is available for the composite product evaluator. This evidence may
4851 include, amongst other, the element of evidence for the dependent component reception, acceptance
4852 and parameterisation by the base component developer (e.g. in form of his acknowledgement
4853 statement).

4854

4855 **12.10.4 ALC_COMP.1 Integration of the dependent component into the related base component**
 4856 **and Consistency check for delivery and acceptance procedures**

4857 Dependencies: No dependencies

4858 **12.10.4.1 Developer action elements**

4859 **12.10.4.1.1 ALC_COMP.1.1D**

4860 **The developer shall provide components configuration evidence.**

4861 **12.10.4.2 Content and presentation elements**

4862 **12.10.4.2.1 ALC_COMP.1.1C**

4863 **The components configuration evidence shall show that the evaluated version of the dependent**
 4864 **component has been installed onto / embedded into the evaluated version of the related base**
 4865 **component.**

4866 **12.10.4.2.2 ALC_COMP.1.2C**

4867 **The components configuration evidence shall show that:**

4868 **i. The evidence for delivery and acceptance compatibility shall show that the delivery**
 4869 **procedures of the base component developer and the dependent component developer are**
 4870 **compatible with the acceptance procedures of the composite product integrator.**

4871 **ii. The evidence shall show that preparative guidance procedures prescribed by the base**
 4872 **component developer and the dependent component developer are either actually being**
 4873 **used by the composite product integrator or compatible with the composite product**
 4874 **integrator guidance and do not contradict each other.**

4875 **12.10.4.3 Evaluator action elements**

4876 **12.10.4.3.1 ALC_COMP.1.1E**

4877 **The evaluator shall confirm that the information provided meets all requirements for content**
 4878 **and presentation of evidence.**

4879 **12.10.4.3.2 ALC_COMP.1.2E**

4880 **The evaluator shall confirm that the evidence for delivery compatibility is complete, coherent,**
 4881 **and internally consistent.**

4882 **13 Class ATE: Tests**

4883 **13.1 Introduction**

4884 The class "Tests" encompasses five families: Coverage (ATE_COV), Depth (ATE_DPT), Independent
 4885 testing (ATE_IND) (i.e. functional testing performed by evaluators), Functional tests (ATE_FUN) and
 4886 Composite functional testing (ATE_COMP). Testing provides assurance that the TSF behaves as
 4887 described (in the functional specification, TOE design, implementation representation, and allows
 4888 straightforward traceability of SFR in test scenario).

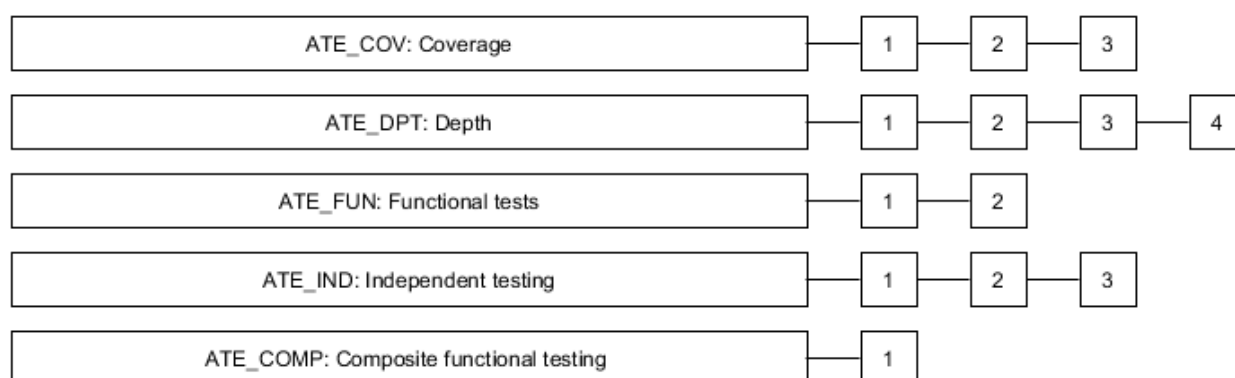
4889 The emphasis in this class is on confirmation that the TSF operates according to its design
 4890 descriptions. This class does not address penetration testing, which is based upon an analysis of the
 4891 TSF that specifically seeks to identify vulnerabilities in the design and implementation of the TSF.
 4892 Penetration testing is addressed separately as an aspect of vulnerability assessment in the AVA:
 4893 Vulnerability assessment class.

4894 The ATE: Tests class separates testing into developer testing and evaluator testing. The Coverage
 4895 (ATE_COV), and Depth (ATE_DPT) families address the completeness of developer testing. Coverage
 4896 (ATE_COV) addresses the rigour with which the functional specification is tested; Depth (ATE_DPT)
 4897 addresses whether testing against other design descriptions (security architecture, TOE design, and
 4898 implementation representation) is required.

4899 Functional tests (ATE_FUN) addresses the performing of the tests by the developer and how this
 4900 testing should be documented. Finally, Independent testing (ATE_IND) then addresses evaluator
 4901 testing: whether the evaluator should repeat part or all of the developer testing and how much
 4902 independent testing the evaluator should do.

4903 Composite functional testing (ATE_COMP) determines whether the composite product as a whole
 4904 exhibits the properties necessary to satisfy the functional requirements of its Security Target.

4905 Figure 11 shows the families within this class, and the hierarchy of components within the families.



4906
 4907 **Figure 11 — ATE: Tests class decomposition**

4908 **13.2 Coverage (ATE_COV)**

4909 **13.2.1 Objectives**

4910 This family establishes that the TSF has been tested against its functional specification. This is
 4911 achieved through an examination of developer evidence of correspondence.

4912 **13.2.2 Component levelling**

4913 The components in this family are levelled on the basis of specification.

4914 **13.2.3 Application notes**

4915 **13.2.4 ATE_COV.1 Evidence of coverage**

4916 Dependencies: ADV_FSP.2 Security-enforcing functional specification

4917 ATE_FUN.1 Functional testing

4918 **Objectives**

4919 The objective of this component is to establish that some of the TSFIs have been tested.

4920 **Application notes**

4921 In this component the developer shows how tests in the test documentation correspond to TSFIs in the
4922 functional specification. This can be achieved by a statement of correspondence, perhaps using a table.

4923 **Developer action elements**

4924 **ATE_COV.1.1D**

4925 **The developer shall provide evidence of the test coverage.**

4926 **Content and presentation elements**

4927 **ATE_COV.1.1C**

4928 **The evidence of the test coverage shall show the correspondence between the tests in the test**
4929 **documentation and the TSFIs in the functional specification.**

4930 **Evaluator action elements**

4931 **ATE_COV.1.1E**

4932 **The evaluator shall confirm that the information provided meets all requirements for content**
4933 **and presentation of evidence.**

4934 **13.2.5 ATE_COV.2 Analysis of coverage**

4935 Dependencies: ADV_FSP.2 Security-enforcing functional specification

4936 ATE_FUN.1 Functional testing

4937 **Objectives**

4938 The objective of this component is to confirm that all of the TSFIs have been tested.

4939 **Application notes**

4940 In this component the developer confirms that tests in the test documentation correspond to all of the
4941 TSFIs in the functional specification. This can be achieved by a statement of correspondence, perhaps
4942 using a table, but the developer also provides an analysis of the test coverage.

4943 **Developer action elements**

4944 **ATE_COV.2.1D**

4945 The developer shall provide **an analysis** of the test coverage.

4946 **Content and presentation elements**

4947 **ATE_COV.2.1C**

4948 The **analysis** of the test coverage shall **demonstrate** the correspondence between the tests in the test
4949 documentation and the TSFIs in the functional specification.

4950 **ATE_COV.2.2C**

4951 **The analysis of the test coverage shall demonstrate that all TSFIs in the functional specification**
4952 **have been tested.**

4953 **Evaluator action elements**

4954 **ATE_COV.2.1E**

4955 The evaluator shall confirm that the information provided meets all requirements for content and
4956 presentation of evidence.

4957 **13.2.6 ATE_COV.3 Rigorous analysis of coverage**

4958 Dependencies: ADV_FSP.2 Security-enforcing functional specification

4959 ATE_FUN.1 Functional testing

4960 **Objectives**

4961 In this component, the objective is to confirm that the developer performed exhaustive tests of all
4962 interfaces in the functional specification.

4963 The objective of this component is to confirm that all parameters of all of the TSFIs have been tested.

4964 **Application notes**

4965 In this component the developer is required to show how tests in the test documentation correspond
4966 to all of the TSFIs in the functional specification. This can be achieved by a statement of
4967 correspondence, perhaps using a table, but in addition the developer is required to demonstrate that
4968 the tests exercise all of the parameters of all TSFIs. This additional requirement includes bounds
4969 testing (i.e. verifying that errors are generated when stated limits are exceeded) and negative testing
4970 (e.g. when access is given to User A, verifying not only that User A now has access, but also that User B
4971 did not suddenly gain access). This kind of testing is not, strictly speaking, *exhaustive* because not
4972 every possible value of the parameters is expected to be checked.

4973 **Developer action elements**

4974 **ATE_COV.3.1D**

4975 The developer shall provide an analysis of the test coverage.

4976 **Content and presentation elements**

4977 **ATE_COV.3.1C**

4978 The analysis of the test coverage shall demonstrate the correspondence between the tests in the test
4979 documentation and the TSFIs in the functional specification.

4980 **ATE_COV.3.2C**

4981 The analysis of the test coverage shall demonstrate that all TSFIs in the functional specification have
 4982 been **completely** tested.

4983 **Evaluator action elements**4984 **ATE_COV.3.1E**

4985 The evaluator shall confirm that the information provided meets all requirements for content and
 4986 presentation of evidence.

4987 **13.3 Depth (ATE_DPT)**4988 **13.3.1 Objectives**

4989 The components in this family deal with the level of detail to which the TSF is tested by the developer.
 4990 Testing of the TSF is based upon increasing depth of information derived from additional design
 4991 representations and descriptions (TOE design, implementation representation, and security
 4992 architecture description).

4993 The objective is to counter the risk of missing an error in the development of the TOE. Testing that
 4994 exercises specific internal interfaces can provide assurance not only that the TSF exhibits the desired
 4995 external security behaviour, but also that this behaviour stems from correctly operating internal
 4996 functionality.

4997 **13.3.2 Component levelling**

4998 The components in this family are levelled on the basis of increasing detail provided in the TSF
 4999 representations, from the TOE design to the implementation representation. This levelling reflects the
 5000 TSF representations presented in the ADV class.

5001 **13.3.3 Application notes**

5002 The TOE design describes the internal components (e.g. subsystems) and, perhaps, modules of the TSF,
 5003 together with a description of the interfaces among these components and modules. Evidence of
 5004 testing of this TOE design must show that the internal interfaces have been exercised and seen to
 5005 behave as described. This may be achieved through testing via the external interfaces of the TSF, or by
 5006 testing of the TOE subsystem or module interfaces in isolation, perhaps employing a test harness. In
 5007 cases where some aspects of an internal interface cannot be tested via the external interfaces, there
 5008 should either be justification that these aspects need not be tested, or the internal interface needs to
 5009 be tested directly. In the latter case the TOE design needs to be sufficiently detailed in order to
 5010 facilitate direct testing.

5011 In cases where the description of the TSF's architectural soundness (in Security Architecture
 5012 (ADV_ARC)) cites specific mechanisms, the tests performed by the developer must show that the
 5013 mechanisms have been exercised and seen to behave as described.

5014 At the highest component of this family, the testing is performed not only against the TOE design, but
 5015 also against the implementation representation.

5016 **13.3.4 ATE_DPT.1 Testing: basic design**

5017 Dependencies: ADV_ARC.1 Security architecture description

5018 ADV_TDS.2 Architectural design

5019 ATE_FUN.1 Functional testing

5020 **Objectives**

5021 The subsystem descriptions of the TSF provide a high-level description of the internal workings of the
5022 TSF. Testing at the level of the TOE subsystems provides assurance that the TSF subsystems behave
5023 and interact as described in the TOE design and the security architecture description.

5024 **Developer action elements**

5025 **ATE_DPT.1.1D**

5026 **The developer shall provide the analysis of the depth of testing.**

5027 **Content and presentation elements**

5028 **ATE_DPT.1.1C**

5029 **The analysis of the depth of testing shall demonstrate the correspondence between the tests in**
5030 **the test documentation and the TSF subsystems in the TOE design.**

5031 **ATE_DPT.1.2C**

5032 **The analysis of the depth of testing shall demonstrate that all TSF subsystems in the TOE design**
5033 **have been tested.**

5034 **Evaluator action elements**

5035 **ATE_DPT.1.1E**

5036 **The evaluator shall confirm that the information provided meets all requirements for content**
5037 **and presentation of evidence.**

5038 **13.3.5 ATE_DPT.2 Testing: security enforcing modules**

5039 Dependencies: ADV_ARC.1 Security architecture description

5040 ADV_TDS.3 Basic modular design

5041 ATE_FUN.1 Functional testing

5042 **Objectives**

5043 The subsystem and module descriptions of the TSF provide a high-level description of the internal
5044 workings, and a description of the interfaces of the SFR-enforcing modules, of the TSF. Testing at this
5045 level of TOE description provides assurance that the TSF subsystems and SFR-enforcing modules
5046 behave and interact as described in the TOE design and the security architecture description.

5047 **Developer action elements**

5048 **ATE_DPT.2.1D**

5049 The developer shall provide the analysis of the depth of testing.

5050 **Content and presentation elements**

5051 **ATE_DPT.2.1C**

5052 The analysis of the depth of testing shall demonstrate the correspondence between the tests in the test
5053 documentation and the TSF subsystems **and SFR-enforcing modules** in the TOE design.

5054 **ATE_DPT.2.2C**

5055 The analysis of the depth of testing shall demonstrate that all TSF subsystems in the TOE design have
5056 been tested.

5057 **ATE_DPT.2.3C**

5058 **The analysis of the depth of testing shall demonstrate that the SFR-enforcing modules in the**
5059 **TOE design have been tested.**

5060 **Evaluator action elements**

5061 **ATE_DPT.2.1E**

5062 The evaluator shall confirm that the information provided meets all requirements for content and
5063 presentation of evidence.

5064 **13.3.6 ATE_DPT.3 Testing: modular design**

5065 Dependencies: ADV_ARC.1 Security architecture description

5066 ADV_TDS.4 Semiformal modular design

5067 ATE_FUN.1 Functional testing

5068 **Objectives**

5069 The subsystem and module descriptions of the TSF provide a high-level description of the internal
5070 workings, and a description of the interfaces of the modules, of the TSF. Testing at this level of TOE
5071 description provides assurance that the TSF subsystems and modules behave and interact as
5072 described in the TOE design and the security architecture description.

5073 **Developer action elements**

5074 **ATE_DPT.3.1D**

5075 The developer shall provide the analysis of the depth of testing.

5076 **Content and presentation elements**

5077 **ATE_DPT.3.1C**

5078 The analysis of the depth of testing shall demonstrate the correspondence between the tests in the test
5079 documentation and the TSF subsystems and modules in the TOE design.

5080 **ATE_DPT.3.2C**

5081 The analysis of the depth of testing shall demonstrate that all TSF subsystems in the TOE design have
5082 been tested.

5083 **ATE_DPT.3.3C**

5084 The analysis of the depth of testing shall demonstrate that **all TSF** modules in the TOE design have
5085 been tested.

5086 **Evaluator action elements**5087 **ATE_DPT.3.1E**

5088 The evaluator shall confirm that the information provided meets all requirements for content and
5089 presentation of evidence.

5090 **13.3.7 ATE_DPT.4 Testing: implementation representation**

5091 Dependencies: ADV_ARC.1 Security architecture description

5092 ADV_TDS.4 Semiformal modular design

5093 ADV_IMP.1 Implementation representation of the TSF

5094 ATE_FUN.1 Functional testing

5095 **Objectives**

5096 The subsystem and module descriptions of the TSF provide a high-level description of the internal
5097 workings, and a description of the interfaces of the modules, of the TSF. Testing at this level of TOE
5098 description provides assurance that the TSF subsystems and modules behave and interact as
5099 described in the TOE design and the security architecture description, and in accordance with the
5100 implementation representation.

5101 **Developer action elements**5102 **ATE_DPT.4.1D**

5103 The developer shall provide the analysis of the depth of testing.

5104 **Content and presentation elements**5105 **ATE_DPT.4.1C**

5106 The analysis of the depth of testing shall demonstrate the correspondence between the tests in the test
5107 documentation and the TSF subsystems and modules in the TOE design.

5108 **ATE_DPT.4.2C**

5109 The analysis of the depth of testing shall demonstrate that all TSF subsystems in the TOE design have
5110 been tested.

5111 **ATE_DPT.4.3C**

5112 The analysis of the depth of testing shall demonstrate that all modules in the TOE design have been
5113 tested.

5114 **ATE_DPT.4.4C**

5115 **The analysis of the depth of testing shall demonstrate that the TSF operates in accordance with**
 5116 **its implementation representation.**

5117 **Evaluator action elements**5118 **ATE_DPT.4.1E**

5119 The evaluator shall confirm that the information provided meets all requirements for content and
 5120 presentation of evidence.

5121 **13.4 Functional tests (ATE_FUN)**5122 **13.4.1 Objectives**

5123 Functional testing performed by the developer provides assurance that the tests in the test
 5124 documentation are performed and documented correctly. The correspondence of these tests to the
 5125 design descriptions of the TSF is achieved through the Coverage (ATE_COV) and Depth (ATE_DPT)
 5126 families.

5127 This family contributes to providing assurance that the likelihood of undiscovered flaws is relatively
 5128 small.

5129 The families Coverage (ATE_COV), Depth (ATE_DPT) and Functional tests (ATE_FUN) are used in
 5130 combination to define the evidence of testing to be supplied by a developer. Independent functional
 5131 testing by the evaluator is specified by Independent testing (ATE_IND).

5132 **13.4.2 Component levelling**

5133 This family contains two components, the higher requiring that ordering dependencies are analysed.

5134 **13.4.3 Application notes**

5135 Procedures for performing tests are expected to provide instructions for using test programs and test
 5136 suites, including the test environment, test conditions, test data parameters and values. The test
 5137 procedures should also show how the test results are derived from the test inputs.

5138 Ordering dependencies are relevant when the successful execution of a particular test depends upon
 5139 the existence of a particular state. For example, this might require that test A be executed immediately
 5140 before test B, since the state resulting from the successful execution of test A is a prerequisite for the
 5141 successful execution of test B. Thus, failure of test B could be related to a problem with the ordering
 5142 dependencies. In the above example, test B could fail because test C (rather than test A) was executed
 5143 immediately before it, or the failure of test B could be related to a failure of test A.

5144 **13.4.4 ATE_FUN.1 Functional testing**

5145 Dependencies: ATE_COV.1 Evidence of coverage

5146 **Objectives**

5147 The objective is for the developer to demonstrate that the tests in the test documentation are
 5148 performed and documented correctly.

5149 **Developer action elements**

5150 **ATE_FUN.1.1D**

5151 **The developer shall test the TSF and document the results.**

5152 **ATE_FUN.1.2D**

5153 **The developer shall provide test documentation.**

5154 **Content and presentation elements**

5155 **ATE_FUN.1.1C**

5156 **The test documentation shall consist of test plans, expected test results and actual test results.**

5157 **ATE_FUN.1.2C**

5158 **The test plans shall identify the tests to be performed and describe the scenarios for**
 5159 **performing each test. These scenarios shall include any ordering dependencies on the results**
 5160 **of other tests.**

5161 **ATE_FUN.1.3C**

5162 **The expected test results shall show the anticipated outputs from a successful execution of the**
 5163 **tests.**

5164 **ATE_FUN.1.4C**

5165 **The actual test results shall be consistent with the expected test results.**

5166 **Evaluator action elements**

5167 **ATE_FUN.1.1E**

5168 **The evaluator shall confirm that the information provided meets all requirements for content**
 5169 **and presentation of evidence.**

5170 **13.4.5 ATE_FUN.2 Ordered functional testing**

5171 Dependencies: ATE_COV.1 Evidence of coverage

5172 **Objectives**

5173 The objectives are for the developer to demonstrate that the tests in the test documentation are
 5174 performed and documented correctly, and to ensure that testing is structured such as to avoid circular
 5175 arguments about the correctness of the interfaces being tested.

5176 **Application notes**

5177 Although the test procedures may state pre-requisite initial test conditions in terms of ordering of
 5178 tests, they may not provide a rationale for the ordering. An analysis of test ordering is an important
 5179 factor in determining the adequacy of testing, as there is a possibility of faults being concealed by the
 5180 ordering of tests.

5181 **Developer action elements**

5182 **ATE_FUN.2.1D**

5183 The developer shall test the TSF and document the results.

5184 **ATE_FUN.2.2D**

5185 The developer shall provide test documentation.

5186 **Content and presentation elements**

5187 **ATE_FUN.2.1C**

5188 The test documentation shall consist of test plans, expected test results and actual test results.

5189 **ATE_FUN.2.2C**

5190 The test plans shall identify the tests to be performed and describe the scenarios for performing each
5191 test. These scenarios shall include any ordering dependencies on the results of other tests.

5192 **ATE_FUN.2.3C**

5193 The expected test results shall show the anticipated outputs from a successful execution of the tests.

5194 **ATE_FUN.2.4C**

5195 The actual test results shall be consistent with the expected test results.

5196 **ATE_FUN.2.5C**

5197 **The test documentation shall include an analysis of the test procedure ordering dependencies.**

5198 **Evaluator action elements**

5199 **ATE_FUN.2.1E**

5200 The evaluator shall confirm that the information provided meets all requirements for content and
5201 presentation of evidence.

5202 **13.5 Independent testing (ATE_IND)**

5203 **13.5.1 Objectives**

5204 The objectives of this family are built upon the assurances achieved in the ATE_FUN, ATE_COV, and
5205 ATE_DPT families by verifying the developer testing and performing additional tests by the evaluator.

5206 **13.5.2 Component levelling**

5207 Levelling is based upon the amount of developer test documentation and test support and the amount
5208 of evaluator testing.

5209 **13.5.3 Application notes**

5210 This family deals with the degree to which there is independent functional testing of the TSF.
5211 Independent functional testing may take the form of repeating the developer's functional tests (in

5212 whole or in part) or of extending the scope or the depth of the developer's tests. These activities are
 5213 complementary, and an appropriate mix must be planned for each TOE, which takes into account the
 5214 availability and coverage of test results, and the functional complexity of the TSF.

5215 Sampling of developer tests is intended to provide confirmation that the developer has carried out his
 5216 planned test programme on the TSF, and has correctly recorded the results. The size of sample
 5217 selected will be influenced by the detail and quality of the developer's functional test results. The
 5218 evaluator will also need to consider the scope for devising additional tests, and the relative benefit that
 5219 may be gained from effort in these two areas. It is recognised that repetition of all developer tests may
 5220 be feasible and desirable in some cases, but may be very arduous and less productive in others. The
 5221 highest component in this family should therefore be used with caution. Sampling will address the
 5222 whole range of test results available, including those supplied to meet the requirements of both
 5223 Coverage (ATE_COV) and Depth (ATE_DPT).

5224 There is also a need to consider the different configurations of the TOE that are included within the
 5225 evaluation. The evaluator will need to assess the applicability of the results provided, and to plan his
 5226 own testing accordingly.

5227 The suitability of the TOE for testing is based on the access to the TOE, and the supporting
 5228 documentation and information required (including any test software or tools) to run tests. The need
 5229 for such support is addressed by the dependencies to other assurance families.

5230 Additionally, suitability of the TOE for testing may be based on other considerations. For example, the
 5231 version of the TOE submitted by the developer may not be the final version.

5232 The term *interfaces* refers to interfaces described in the functional specification and TOE design, and
 5233 parameters passed through invocations identified in the implementation representation. The exact set
 5234 of interfaces to be used is selected through Coverage (ATE_COV) and the Depth (ATE_DPT)
 5235 components.

5236 References to a subset of the interfaces are intended to allow the evaluator to design an appropriate
 5237 set of tests which is consistent with the objectives of the evaluation being conducted.

5238 **13.5.4 ATE_IND.1 Independent testing - conformance**

5239 Dependencies: ADV_FSP.1 Basic functional specification

5240 AGD_OPE.1 Operational user guidance

5241 AGD_PRE.1 Preparative procedures

5242 **Objectives**

5243 In this component, the objective is to demonstrate that the TOE operates in accordance with its design
 5244 representations and guidance documents.

5245 **Application notes**

5246 This component does not address the use of developer test results. It is applicable where such results
 5247 are not available, and also in cases where the developer's testing is accepted without validation. The
 5248 evaluator is required to devise and conduct tests with the objective of confirming that the TOE
 5249 operates in accordance with its design representations, including but not limited to the functional
 5250 specification. The approach is to gain confidence in correct operation through representative testing,
 5251 rather than to conduct every possible test. The extent of testing to be planned for this purpose is a

5252 methodology issue, and needs to be considered in the context of a particular TOE and the balance of
5253 other evaluation activities.

5254 **Developer action elements**

5255 **ATE_IND.1.1D**

5256 **The developer shall provide the TOE for testing.**

5257 **Content and presentation elements**

5258 **ATE_IND.1.1C**

5259 **The TOE shall be suitable for testing.**

5260 **Evaluator action elements**

5261 **ATE_IND.1.1E**

5262 **The evaluator shall confirm that the information provided meets all requirements for content**
5263 **and presentation of evidence.**

5264 **ATE_IND.1.2E**

5265 **The evaluator shall test a subset of the TSF to confirm that the TSF operates as specified.**

5266 **13.5.5 ATE_IND.2 Independent testing - sample**

5267 Dependencies: ADV_FSP.2 Security-enforcing functional specification

5268 AGD_OPE.1 Operational user guidance

5269 AGD_PRE.1 Preparative procedures

5270 ATE_COV.1 Evidence of coverage

5271 ATE_FUN.1 Functional testing

5272 **Objectives**

5273 In this component, the objective is to demonstrate that the TOE operates in accordance with its design
5274 representations and guidance documents. Evaluator testing confirms that the developer performed
5275 some tests of some interfaces in the functional specification.

5276 **Application notes**

5277 The intent is that the developer should provide the evaluator with materials necessary for the efficient
5278 reproduction of developer tests. This may include such things as machine-readable test
5279 documentation, test programs, etc.

5280 This component contains a requirement that the evaluator has available test results from the
5281 developer to supplement the programme of testing. The evaluator will repeat a sample of the
5282 developer's tests to gain confidence in the results obtained. Having established such confidence the
5283 evaluator will build upon the developer's testing by conducting additional tests that exercise the TOE
5284 in a different manner. By using a platform of validated developer test results the evaluator is able to

5285 gain confidence that the TOE operates correctly in a wider range of conditions than would be possible
 5286 purely using the developer's own efforts, given a fixed level of resource. Having gained confidence that
 5287 the developer has tested the TOE, the evaluator will also have more freedom, where appropriate, to
 5288 concentrate testing in areas where examination of documentation or specialist knowledge has raised
 5289 particular concerns.

5290 **Developer action elements**

5291 **ATE_IND.2.1D**

5292 The developer shall provide the TOE for testing.

5293 **Content and presentation elements**

5294 **ATE_IND.2.1C**

5295 The TOE shall be suitable for testing.

5296 **ATE_IND.2.2C**

5297 **The developer shall provide an equivalent set of resources to those that were used in the**
 5298 **developer's functional testing of the TSF.**

5299 **Evaluator action elements**

5300 **ATE_IND.2.1E**

5301 The evaluator shall confirm that the information provided meets all requirements for content and
 5302 presentation of evidence.

5303 **ATE_IND.2.2E**

5304 **The evaluator shall execute a sample of tests in the test documentation to verify the developer**
 5305 **test results.**

5306 **ATE_IND.2.3E**

5307 The evaluator shall test a subset of the TSF to confirm that the TSF operates as specified.

5308 **13.5.6 ATE_IND.3 Independent testing - complete**

5309 Dependencies: ADV_FSP.4 Complete functional specification

5310 AGD_OPE.1 Operational user guidance

5311 AGD_PRE.1 Preparative procedures

5312 ATE_COV.1 Evidence of coverage

5313 ATE_FUN.1 Functional testing

5314 Objectives

5315 In this component, the objective is to demonstrate that the TOE operates in accordance with its design
 5316 representations and guidance documents. Evaluator testing includes repeating all of the developer
 5317 tests.

5318 Application notes

5319 The intent is that the developer should provide the evaluator with materials necessary for the efficient
 5320 reproduction of developer tests. This may include such things as machine-readable test
 5321 documentation, test programs, etc.

5322 In this component the evaluator must repeat all of the developer's tests as part of the programme of
 5323 testing. As in the previous component the evaluator will also conduct tests that aim to exercise the TSF
 5324 in a different manner from that achieved by the developer. In cases where developer testing has been
 5325 exhaustive, there may remain little scope for this.

5326 Developer action elements

5327 ATE_IND.3.1D

5328 The developer shall provide the TOE for testing.

5329 Content and presentation elements

5330 ATE_IND.3.1C

5331 The TOE shall be suitable for testing.

5332 ATE_IND.3.2C

5333 The developer shall provide an equivalent set of resources to those that were used in the developer's
 5334 functional testing of the TSF.

5335 Evaluator action elements

5336 ATE_IND.3.1E

5337 The evaluator shall confirm that the information provided meets all requirements for content and
 5338 presentation of evidence.

5339 ATE_IND.3.2E

5340 The evaluator shall execute **all** tests in the test documentation to verify the developer test results.

5341 ATE_IND.3.3E

5342 The evaluator shall test the TSF to confirm that the **entire** TSF operates as specified.

5343 13.6 Composite functional testing (ATE_COMP)

5344 13.6.1 Objectives

5345 The aim of this family is to determine whether the composite product as a whole exhibits the
 5346 properties necessary to satisfy the functional requirements of its composite product Security Target.

5347 **13.6.2 Component levelling**

5348 This family contains only one component.

5349 **13.6.3 Application notes**

5350 A composite product can be tested by testing its components separately and by testing the integrated
 5351 product. Separate testing means that its base component and its dependent component are being
 5352 tested independently of each other. A lot of tests of the base component may have been performed
 5353 within the scope of its accomplished evaluation. The dependent component may be tested on a
 5354 simulator or an emulator, which represent a virtual machine.

5355 Integration testing means that the composite product is being tested as it is: the dependent component
 5356 is running together with the related base component.

5357 Some dependent component functionality testing can only be performed on emulators, before its
 5358 embedding/integration onto the base component, as effectiveness of this testing may not be visible
 5359 using the interfaces of the composite product. Nevertheless, functional testing of the composite
 5360 product shall be performed also on composite product samples according to the description of the
 5361 security functions of the composite product and using the standard approach as required by the
 5362 relevant ATE assurance class. No additional developer's action is required here.

5363 Since the amount, the coverage and the depth of the functional tests of the base component have
 5364 already been validated by the base component evaluation, it is not necessary to re-perform these tasks
 5365 in the composite evaluation. Please note that the *ETR for composite evaluation* does not provide any
 5366 information on functional testing for the base component.

5367 The behaviour of implementation of some SFRs can depend on properties of the base component as
 5368 well as on the dependent component (e.g. correctness of the measures of the composite product to
 5369 withstand a side channel attack or correctness of the implementation of tamper resistance against
 5370 physical attacks). In such case the SFR implementation shall be tested on the final composite product,
 5371 but not on a simulator or an emulator.

5372 This family focuses exclusively on testing of the composite product as a whole and represents merely
 5373 partial efforts within the general test approach being covered by the assurance class ATE. These
 5374 integration tests shall be specified and performed, whereby the approach of the standard assurance
 5375 families of the class ATE shall be applied.

5376 The composite product evaluation sponsor shall ensure that the following is available for the
 5377 composite product evaluator:

5378 composite product samples suitable for testing.

5379 **13.6.4 ATE_COMP.1 Composite product functional testing**

5380 Dependencies: No dependencies

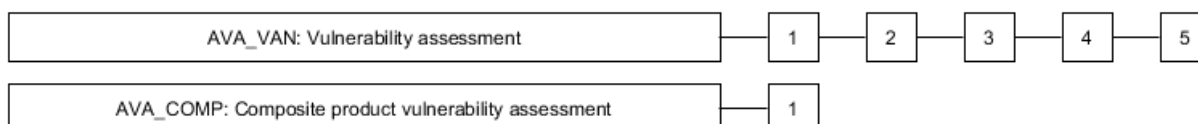
5381 **13.6.4.1 Developer action elements**

5382 **13.6.4.1.1 ATE_COMP.1.1D**

5383 **The developer shall provide a set of tests as required by the assurance package chosen.**

5384 **13.6.4.1.2 ATE_COMP.1.2D**5385 **The developer shall provide the composite product for testing.**5386 **13.6.4.2 Content and presentation elements**5387 **13.6.4.2.1 ATE_COMP.1.1C**5388 **Content and presentation of the specification and documentation of the *integration* tests shall**
5389 **correspond to the standard¹⁰ requirements of the assurance families ATE_FUN and ATE_COV.**5390 **13.6.4.2.2 ATE_COMP.1.2C**5391 **The composite product provided shall be suitable for testing.**5392 **13.6.4.3 Evaluator action elements**5393 **13.6.4.3.1 ATE_COMP.1.1E**5394 **The evaluator shall confirm that the information provided meets all requirements for content**
5395 **and presentation of evidence.**5396 **14 Class AVA: Vulnerability assessment**5397 **14.1 Introduction**5398 The AVA: Vulnerability assessment class addresses the possibility of exploitable vulnerabilities
5399 introduced in the development or the operation of the TOE.

5400 Figure 12 shows the families within this class, and the hierarchy of components within the families.



5401

5402 **Figure 12 — AVA: Vulnerability assessment class decomposition**5403 **14.2 Application notes**5404 Generally, the vulnerability assessment activity covers various vulnerabilities in the development and
5405 operation of the TOE. Development vulnerabilities take advantage of some property of the TOE ,or the
5406 product where the TOE resides, which was introduced during its development, e.g. defeating the TSF
5407 self-protection through tampering, direct attack or monitoring of the TSF, defeating the TSF domain
5408 separation through monitoring or direct attack the TSF, or defeating non-bypassability through
5409 circumventing (bypassing) the TSF. Explicit dependencies of the TOE on IT systems in the
5410 environment must also be considered. Operational vulnerabilities take advantage of weaknesses in
5411 non-technical countermeasures to violate the TOE SFRs, e.g. misuse or incorrect configuration. Misuse

¹⁰ i.e. as defined by ISO/IEC 18045

5412 investigates whether the TOE can be configured or used in a manner that is insecure, but that an
 5413 administrator or user of the TOE would reasonably believe to be secure.

5414 Assessment of development vulnerabilities is covered by the assurance family AVA_VAN. Basically, all
 5415 development vulnerabilities can be considered in the context of AVA_VAN due to the fact, that this
 5416 family allows application of a wide range of assessment methodologies being unspecific to the kind of
 5417 an attack scenario. These unspecific assessment methodologies comprise, among other, also the
 5418 specific methodologies for those TSF where covert channels are to be considered (a channel capacity
 5419 estimation can be done using informal engineering measurements, as well as actual test
 5420 measurements) or can be overcome by the use of sufficient resources in the form of a direct attack
 5421 (underlying technical concept of those TSF is based on probabilistic or permutational mechanisms; a
 5422 qualification of their security behaviour and the effort required to overcome them can be made using a
 5423 quantitative or statistical analysis).

5424 If there are security objectives specified in the ST to either to prevent one user of the TOE from
 5425 observing activity associated with another user of the TOE, or to ensure that information flows cannot
 5426 be used to achieve enforced illicit data signals, covert channel analysis should be considered during
 5427 the conduct of the vulnerability analysis. This is often reflected by the inclusion of Unobservability
 5428 (FPR_UNO) and multilevel access control policies specified through Access control policy (FDP_ACC)
 5429 and/or Information flow control policy (FDP_IFC) requirements in the ST.

5430 **14.3 Vulnerability analysis (AVA_VAN)**

5431 **14.3.1 Objectives**

5432 Vulnerability analysis is an assessment to determine whether potential vulnerabilities identified,
 5433 during the evaluation of the development and anticipated operation of the TOE or by other methods
 5434 (e.g. by flaw hypotheses or quantitative or statistical analysis of the security behaviour of the
 5435 underlying security mechanisms), could allow attackers to violate the SFRs.

5436 Vulnerability analysis deals with the threats that an attacker will be able to discover flaws that will
 5437 allow unauthorised access to data and functionality, allow the ability to interfere with or alter the TSF,
 5438 or interfere with the authorised capabilities of other users.

5439 In case of a **multi-assurance evaluation** the vulnerability analysis shall assess the defined **sub-TSF** as
 5440 well as the TOE as a whole.

5441 **14.3.2 Component levelling**

5442 Levelling is based on an increasing rigour of vulnerability analysis by the evaluator and increased
 5443 levels of attack potential required by an attacker to identify and exploit the potential vulnerabilities.

5444 **14.3.3 AVA_VAN.1 Vulnerability survey**

5445 Dependencies: ADV_FSP.1 Basic functional specification

5446 AGD_OPE.1 Operational user guidance

5447 AGD_PRE.1 Preparative procedures

5448 **Objectives**

5449 A vulnerability survey of information available in the public domain is performed by the evaluator to
 5450 ascertain potential vulnerabilities that may be easily found by an attacker.

5451 The evaluator performs penetration testing, to confirm that the potential vulnerabilities cannot be
 5452 exploited in the operational environment for the TOE. Penetration testing is performed by the
 5453 evaluator assuming an attack potential of Basic.

5454 **Developer action elements**

5455 **AVA_VAN.1.1D**

5456 **The developer shall provide the TOE for testing.**

5457 **Content and presentation elements**

5458 **AVA_VAN.1.1C**

5459 **The TOE shall be suitable for testing.**

5460 **Evaluator action elements**

5461 **AVA_VAN.1.1E**

5462 **The evaluator shall confirm that the information provided meets all requirements for content**
 5463 **and presentation of evidence.**

5464 **AVA_VAN.1.2E**

5465 **The evaluator shall perform a search of public domain sources to identify potential**
 5466 **vulnerabilities in the TOE.**

5467 **AVA_VAN.1.3E**

5468 **The evaluator shall conduct penetration testing, based on the identified potential**
 5469 **vulnerabilities, to determine that the TOE is resistant to attacks performed by an attacker**
 5470 **possessing Basic attack potential.**

5471 **14.3.4 AVA_VAN.2 Vulnerability analysis**

5472 Dependencies: ADV_ARC.1 Security architecture description

5473 ADV_FSP.2 Security-enforcing functional specification

5474 ADV_TDS.1 Basic design

5475 AGD_OPE.1 Operational user guidance

5476 AGD_PRE.1 Preparative procedures

5477 **Objectives**

5478 A vulnerability analysis is performed by the evaluator to ascertain the presence of potential
 5479 vulnerabilities.

5480 The evaluator performs penetration testing, to confirm that the potential vulnerabilities cannot be
 5481 exploited in the operational environment for the TOE. Penetration testing is performed by the
 5482 evaluator assuming an attack potential of Basic.

5483 **Developer action elements**

5484 **AVA_VAN.2.1D**

5485 The developer shall provide the TOE for testing.

5486 **AVA_VAN.2.2D**

5487 **The developer shall provide a list of third party components included in the TOE and the TOE**
 5488 **delivery.**

5489 **Content and presentation elements**

5490 **AVA_VAN.2.1C**

5491 The TOE shall be suitable for testing.

5492 **AVA_VAN.2.2C**

5493 **The list of third party components shall include components provided by third parties, and that**
 5494 **are part of the TOE or otherwise part of the TOE delivery.**

5495 **Evaluator action elements**

5496 **AVA_VAN.2.1E**

5497 The evaluator shall confirm that the information provided meets all requirements for content and
 5498 presentation of evidence.

5499 **AVA_VAN.2.2E**

5500 The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in
 5501 the TOE **the components in the list of third party components, and specific IT products in the**
 5502 **environment that the TOE depends on.**

5503 **AVA_VAN.2.3E**

5504 **The evaluator shall perform an independent vulnerability analysis of the TOE using the**
 5505 **guidance documentation, functional specification, TOE design and security architecture**
 5506 **description to identify potential vulnerabilities in the TOE.**

5507 **AVA_VAN.2.4E**

5508 The evaluator shall conduct penetration testing, based on the identified potential vulnerabilities, to
 5509 determine that the TOE is resistant to attacks performed by an attacker possessing Basic attack
 5510 potential.

5511 **14.3.5 AVA_VAN.3 Focused vulnerability analysis**

5512 Dependencies: ADV_ARC.1 Security architecture description

5513 ADV_FSP.4 Complete functional specification

5514 ADV_TDS.3 Basic modular design

5515 ADV_IMP.1 Implementation representation of the TSF

5516 AGD_OPE.1 Operational user guidance

5517 AGD_PRE.1 Preparative procedures

5518 ATE_DPT.1 Testing: basic design

5519 **Objectives**

5520 A vulnerability analysis is performed by the evaluator to ascertain the presence of potential
5521 vulnerabilities.

5522 The evaluator performs penetration testing, to confirm that the potential vulnerabilities cannot be
5523 exploited in the operational environment for the TOE. Penetration testing is performed by the
5524 evaluator assuming an attack potential of Enhanced-Basic.

5525 **Developer action elements**

5526 **AVA_VAN.3.1D**

5527 The developer shall provide the TOE for testing.

5528 **AVA_VAN.3.2D**

5529 The developer shall provide a list of third party components included in the TOE and the TOE delivery.

5530 **Content and presentation elements**

5531 **AVA_VAN.3.1C**

5532 The TOE shall be suitable for testing.

5533 **AVA_VAN.3.2C**

5534 The list of third party components shall include components provided by third parties, and that are
5535 part of the TOE or otherwise part of the TOE delivery.

5536 **Evaluator action elements**

5537 **AVA_VAN.3.1E**

5538 The evaluator shall confirm that the information provided meets all requirements for content and
5539 presentation of evidence.

5540 **AVA_VAN.3.2E**

5541 The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in
5542 the TOE the components in the list of third party components, and specific IT products in the
5543 environment that the TOE depends on.

5544 **AVA_VAN.3.3E**

5545 The evaluator shall perform an independent, **focused** vulnerability analysis of the TOE using the
 5546 guidance documentation, functional specification, TOE design, security architecture description **and**
 5547 **implementation representation** to identify potential vulnerabilities in the TOE.

5548 **AVA_VAN.3.4E**

5549 The evaluator shall conduct penetration testing, based on the identified potential vulnerabilities, to
 5550 determine that the TOE is resistant to attacks performed by an attacker possessing **Enhanced-Basic**
 5551 attack potential.

5552 **14.3.6 AVA_VAN.4 Methodical vulnerability analysis**

5553 Dependencies: ADV_ARC.1 Security architecture description

5554 ADV_FSP.4 Complete functional specification

5555 ADV_TDS.3 Basic modular design

5556 ADV_IMP.1 Implementation representation of the TSF

5557 AGD_OPE.1 Operational user guidance

5558 AGD_PRE.1 Preparative procedures

5559 ATE_DPT.1 Testing: basic design

5560 **Objectives**

5561 A methodical vulnerability analysis is performed by the evaluator to ascertain the presence of
 5562 potential vulnerabilities.

5563 The evaluator performs penetration testing, to confirm that the potential vulnerabilities cannot be
 5564 exploited in the operational environment for the TOE. Penetration testing is performed by the
 5565 evaluator assuming an attack potential of Moderate.

5566 **Developer action elements**5567 **AVA_VAN.4.1D**

5568 The developer shall provide the TOE for testing.

5569 **AVA_VAN.4.2D**

5570 The developer shall provide a list of third party components included in the TOE and the TOE delivery.

5571 **Content and presentation elements**5572 **AVA_VAN.4.1C**

5573 The TOE shall be suitable for testing.

5574 **AVA_VAN.4.2C**

5575 The list of third party components shall include components provided by third parties, and that are
 5576 part of the TOE or otherwise part of the TOE delivery.

5577 **Evaluator action elements**5578 **AVA_VAN.4.1E**

5579 The evaluator shall confirm that the information provided meets all requirements for content and
 5580 presentation of evidence.

5581 **AVA_VAN.4.2E**

5582 The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in
 5583 the TOE the components in the list of third party components, and specific IT products in the
 5584 environment that the TOE depends on.

5585 **AVA_VAN.4.3E**

5586 The evaluator shall perform an independent, **methodical** vulnerability analysis of the TOE using the
 5587 guidance documentation, functional specification, TOE design, security architecture description and
 5588 implementation representation to identify potential vulnerabilities in the TOE.

5589 **AVA_VAN.4.4E**

5590 The evaluator shall conduct penetration testing based on the identified potential vulnerabilities to
 5591 determine that the TOE is resistant to attacks performed by an attacker possessing **Moderate** attack
 5592 potential.

5593 **14.3.7 AVA_VAN.5 Advanced methodical vulnerability analysis**

5594 Dependencies: ADV_ARC.1 Security architecture description

5595 ADV_FSP.4 Complete functional specification

5596 ADV_TDS.3 Basic modular design

5597 ADV_IMP.1 Implementation representation of the TSF

5598 AGD_OPE.1 Operational user guidance

5599 AGD_PRE.1 Preparative procedures

5600 ATE_DPT.1 Testing: basic design

5601 **Objectives**

5602 A methodical vulnerability analysis is performed by the evaluator to ascertain the presence of
 5603 potential vulnerabilities.

5604 The evaluator performs penetration testing, to confirm that the potential vulnerabilities cannot be
 5605 exploited in the operational environment for the TOE. Penetration testing is performed by the
 5606 evaluator assuming an attack potential of High.

5607 **Developer action elements**

5608 **AVA_VAN.5.1D**

5609 The developer shall provide the TOE for testing.

5610 **AVA_VAN.5.2D**

5611 The developer shall provide a list of third party components included in the TOE and the TOE delivery.

5612 **Content and presentation elements**

5613 **AVA_VAN.5.1C**

5614 The TOE shall be suitable for testing.

5615 **AVA_VAN.5.2C**

5616 The list of third party components shall include components provided by third parties, and that are
5617 part of the TOE or otherwise part of the TOE delivery.

5618 **Evaluator action elements**

5619 **AVA_VAN.5.1E**

5620 The evaluator shall confirm that the information provided meets all requirements for content and
5621 presentation of evidence.

5622 **AVA_VAN.5.2E**

5623 The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in
5624 the TOE the components in the list of third party components, and specific IT products in the
5625 environment that the TOE depends on.

5626 **AVA_VAN.5.3E**

5627 The evaluator shall perform an independent, methodical vulnerability analysis of the TOE using the
5628 guidance documentation, functional specification, TOE design, security architecture description and
5629 implementation representation to identify potential vulnerabilities in the TOE.

5630 **AVA_VAN.5.4E**

5631 The evaluator shall conduct penetration testing based on the identified potential vulnerabilities to
5632 determine that the TOE is resistant to attacks performed by an attacker possessing **High** attack
5633 potential.

5634 **14.4 Composite vulnerability assessment (AVA_COMP)**

5635 **14.4.1 Objectives**

5636 The aim of this family is to determine the exploitability of flaws or weaknesses in the composite
5637 product as a whole in the intended environment.

5638 **14.4.2 Component levelling**

5639 This family contains only one component.

5640 14.4.3 AVA_COMP.1 Composite product vulnerability assessment

5641 Dependencies: No dependencies

5642 Application notes

5643 This family focuses exclusively on the vulnerability assessment of the composite product *as a whole*
 5644 and represents merely *partial efforts* within the general approach being covered by the standard¹¹
 5645 assurance family of the class AVA: AVA_VAN.

5646 The composite product evaluator shall perform a vulnerability analysis for the composite product
 5647 using, amongst other, the results of the base component evaluation. This vulnerability analysis shall be
 5648 confirmed by penetration testing.

5649 The composite product evaluator shall check that the confidentiality protection of the dependent
 5650 component embedded into/installed onto the base component is consistent with the confidentiality
 5651 level claimed by the dependent component developer for ALC_DVS.

5652 In special cases, the vulnerability analysis and the definition of attacks might be difficult, need
 5653 considerable time and require extensive pre-testing, if only documentation is available. The base
 5654 component may also be used in a way that was not foreseen by the base component developer and the
 5655 base component evaluator, or the dependent component developer may not have followed the
 5656 stipulations provided with the base component. Different possibilities exist to shorten composite
 5657 product vulnerability analysis in such cases: E.g. the composite product evaluator may consult the base
 5658 component evaluator and draw on his experience gained during the base component evaluation.
 5659 Alternatively, an approach aiming on the separation of vulnerabilities of the dependent component
 5660 and the base component by using specific test samples of the base component on which the composite
 5661 product evaluator may load test dependent components on his own discretion. The intention hereby is
 5662 to use test dependent components without countermeasures and without deactivating any base
 5663 component inherent countermeasure.

5664 The results of the vulnerability assessment for the base component of the composite product
 5665 represented in the *ETR for composite evaluation* can be re-used under the following conditions: they
 5666 are up-to-date and all composite activities for correctness – ASE_COMP.1, ALC_COMP.1, ADV_COMP.1
 5667 and ATE_COMP.1 – are finalised with the verdict PASS.

5668 Due to composing of the base component and the dependent component a new quality arises, which
 5669 may cause additional vulnerabilities of the base component which might be not mentioned in the *ETR*
 5670 *for composite evaluation*. In these circumstances the composite product evaluation authority may
 5671 require a re-assessment or re-evaluation of the base component focusing on the new vulnerabilities'
 5672 issues.

5673 The composite product evaluation sponsor shall ensure that the following is made available for the
 5674 composite product evaluator:

- 5675 • the base component-related user guidance,
- 5676 • the base component-related *ETR for composite evaluation* prepared by the base component
 5677 evaluator,

¹¹ i.e. as defined by ISO/IEC 18045

- the report of the base component evaluation authority.

14.4.4 AVA_COMP.1 Composite product vulnerability assessment

Dependencies: No dependencies

14.4.4.1 Developer action elements

14.4.4.1.1 AVA_COMP.1.1D

The developer shall provide the composite product for penetrating testing.

14.4.4.2 Content and presentation elements

14.4.4.2.1 AVA_COMP.1.1C

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

14.4.4.3 Evaluator action elements

14.4.4.3.1 AVA_COMP.1.1E

The evaluator shall conduct penetration testing of the composite product *as a whole* building on the evaluator's own vulnerability analysis to ensure that the vulnerabilities being relevant for the composite product Security Target are not exploitable.

15 Class ACO: Composition

15.1 Introduction

The class ACO: Composition encompasses five families. These families specify assurance requirements that are designed to provide confidence that a composed TOE will operate securely when relying upon security functionality provided by previously evaluated software, firmware or hardware components.

Composition involves taking two or more IT entities successfully evaluated against the ISO/IEC 15408 series security assurance requirements packages (base components and dependent components, see Annex B) and combining them for use, with no further development of either IT entity. The development of additional IT entities is not included (entities that have not previously been the subject of a component evaluation). The composed TOE forms a new product that can be installed and integrated into any specific environment instance that meets the objectives for the environment.

This approach does not provide an alternative approach for the evaluation of components. Composition under ACO provides a composed TOE integrator a method, which can be used as an alternative to other assurance levels specified in ISO/IEC 15408, to gain confidence in a TOE that is the combination of two or more successfully evaluated components without having to re-evaluate the composite TSF. (The composed TOE integrator is referred to as "developer" throughout the ACO class, with any references to the developer of the base or dependent components clarified as such.)

Composed Assurance Packages, as defined in part 5 provide an assurance scale for composed TOEs. This assurance scale is required in addition to other assurance packages, for example the EALs, because to combine components evaluated against another assurance package and gain equivalent assurance in the resulting composed TOE, all SARs have to be applied to the composed TOE. Although reuse can be made of the component TOE evaluation results, there are often additional aspects of the components that have to be considered in the composed TOE, as described in Annex B.3.

Due to the different parties involved in a composed TOE evaluation activity it is generally not possible to gain all necessary evidence about these additional aspects of the components to apply the appropriate EAL. Hence, CAPs have been defined to address the issue of combining evaluated components and gaining a meaningful result. This is discussed further in Annex B.

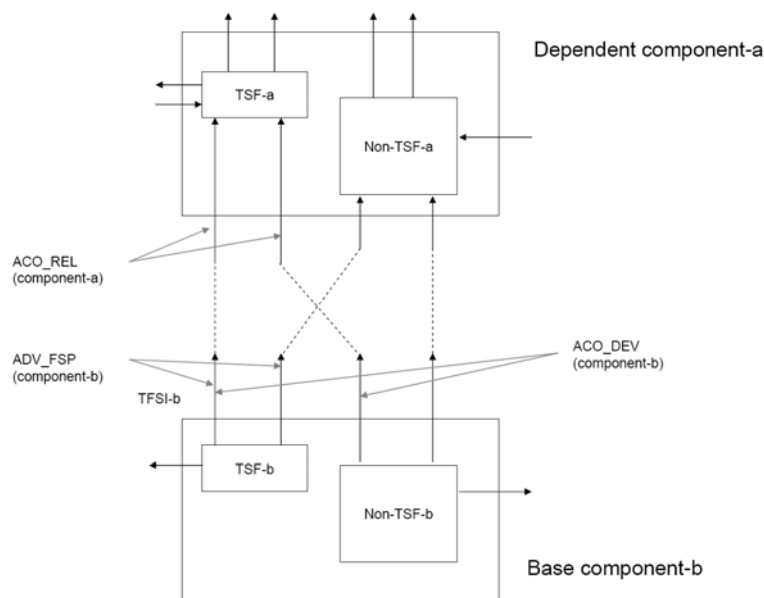


Figure 13 — Relationship between ACO families and interactions between components

In a composed TOE it is generally the case that one component relies on the services provided by another component. The component requiring services is termed the dependent component and the component providing the services is termed the base component. This interaction and distinct is discussed further in Annex B. It is assumed to be the case that the developer of the dependent component is supporting the composed TOE evaluation in some manner (as developer, sponsor, or just cooperating and providing the necessary evaluation evidence from the dependent component evaluation) The ACO components included in the CAP assurance packages should not be used as augmentations for component TOE evaluations, as this would provide no meaningful assurance for the component.

The families within the ACO class interact in a similar manner to the ADV, ATE and AVA classes in a component TOE evaluation and hence leverage from the specification of requirements from those classes where applicable. There are however a few items specific to composed TOE evaluations. To determine how the components interact and identify any deviations from the evaluations of the components, the dependencies that the dependent component has upon the underlying base component are identified (ACO_REL). This reliance on the base component is specified in terms of the interfaces through which the dependent component makes calls for services in support of the dependent component SFRs. The interfaces, and at higher levels the supporting behaviour, provided by the base component in response to those service requests are analysed in ACO_DEV. The ACO_DEV family is based on the ADV_TDS family, as at the simplest level the TSF of each component can be viewed as a subsystem of the composed TOE, with additional portions of each component seen as additional subsystems. Therefore, the interfaces between the components are seen as interactions between subsystems in a component TOE evaluation (see Figure 13).

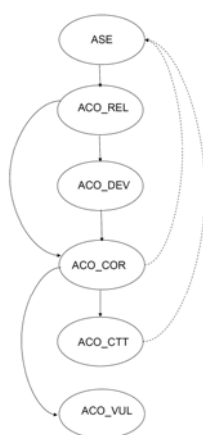
It is possible that the interfaces and supporting behaviour descriptions provided for ACO_DEV are incomplete. This is determined during the conduct of ACO_COR. The ACO_COR family takes the outputs

5745 of ACO_REL and ACO_DEV and determines whether the components are being used in their evaluated
 5746 configuration and identifies where any specifications are incomplete, which are then identified as
 5747 inputs into testing (ACO_CTT) and vulnerability analysis (ACO_VUL) activities of the composed TOE.

5748 Testing of the composed TOE is performed to determine that the composed TOE exhibits the expected
 5749 behaviour as determined by the composed TOE SFRs, and at higher levels demonstrates the
 5750 compatibility of the interfaces between the components of the composed TOE.

5751 The vulnerability analysis of the composed TOE leverages from the outputs of the vulnerability
 5752 analysis of the component evaluations. The composed TOE vulnerability analysis considers any
 5753 residual vulnerabilities from the component evaluations to determine that the residual vulnerabilities
 5754 are not applicable to the composed TOE. A search of publicly available information relating to the
 5755 components is also performed to identify any issues reported in the components since the completion
 5756 of the respective evaluations.

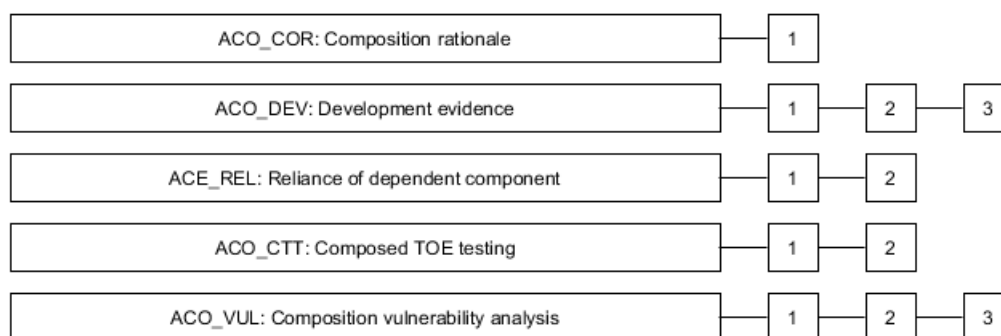
5757 The interaction between the ACO families is depicted in Figure 14 below. This shows by solid arrowed
 5758 lines where the evidence and understanding gained in one family feeds into the next activity and the
 5759 dashed arrows identify where an activity explicitly traces back to the composed TOE SFRs, as
 5760 described above.



5761
 5762 **Figure 14 — Relationship between ACO families**

5763 Further discussion of the definition and interactions within composed TOEs is provided in Annex B.

5764 Figure 15 shows the families within this class, and the hierarchy of components within the families.



5765
 5766 **Figure 15 — ACO: Composition class decomposition**

5767 **15.2 Composition rationale (ACO_COR)**

5768 **15.2.1 Objectives**

5769 This family addresses the requirement to demonstrate that the base component can provide an
5770 appropriate level of assurance for use in composition.

5771 **15.2.2 Component levelling**

5772 There is only a single component in this family.

5773 **15.2.3 ACO_COR.1 Composition rationale**

5774 Dependencies: ACO_DEV.1 Functional Description

5775 ALC_CMC.1 Labelling of the TOE

5776 ACO_REL.1 Basic reliance information

5777 **Developer action elements**

5778 **ACO_COR.1.1D**

5779 **The developer shall provide composition rationale for the base component.**

5780 **Content and presentation elements**

5781 **ACO_COR.1.1C**

5782 **The composition rationale shall demonstrate that a level of assurance at least as high as that of**
5783 **the dependent component has been obtained for the support functionality of the base**
5784 **component, when the base component is configured as required to support the TSF of the**
5785 **dependent component.**

5786 **Evaluator action elements**

5787 **ACO_COR.1.1E**

5788 **The evaluator shall confirm that the information meets all requirements for content and**
5789 **presentation of evidence.**

5790 **15.3 Development evidence (ACO_DEV)**

5791 **15.3.1 Objectives**

5792 This family sets out requirements for a specification of the base component in increasing levels of
5793 detail. Such information is required to gain confidence that the appropriate security functionality is
5794 provided to support the requirements of the dependent component (as identified in the reliance
5795 information).

5796 **15.3.2 Component levelling**

5797 The components are levelled on the basis of increasing amounts of detail about the interfaces
5798 provided, and how they are implemented.

5799 **15.3.3 Application notes**

5800 The TSF of the base component is often defined without knowledge of the dependencies of the
 5801 possible applications with which it may be composed. The TSF of this base component is defined to
 5802 include all parts of the base component that have to be relied upon for enforcement of the base
 5803 component SFRs. This will include all parts of the base component required to implement the base
 5804 component SFRs.

5805 The functional specification of the base component will describe the TSFI in terms of the interfaces the
 5806 base component provides to allow an external entity to invoke operations of the TSF. This includes
 5807 interfaces to the human user to permit interaction with the operation of the TSF invoking SFRs and
 5808 also interfaces allowing an external IT entity to make calls into the TSF.

5809 The functional specification only provides a description of what the TSF provides at its interface and
 5810 the means by which that TSF functionality are invoked. Therefore, the functional specification does not
 5811 necessarily provide a complete interface specification of all possible interfaces available between an
 5812 external entity and the base component. It does not include what the TSF expects/requires from the
 5813 operational environment. The description of what a dependent component TSF relies upon of a base
 5814 component is considered in Reliance of dependent component (ACO_REL) and the development
 5815 information evidence provides a response to the interfaces specified.

5816 The development information evidence includes a specification of the base component. This may be
 5817 the evidence used during evaluation of the base component to satisfy the ADV requirements, or may
 5818 be another form of evidence produced by either the base component developer or the composed TOE
 5819 developer. This specification of the base component is used during Development evidence (ACO_DEV)
 5820 to gain confidence that the appropriate security functionality is provided to support the requirements
 5821 of the dependent component. The level of detail required of this evidence increases to reflect the level
 5822 of required assurance in the composed TOE. This is expected to broadly reflect the increasing
 5823 confidence gained from the application of the assurance packages to the components. The evaluator
 5824 determines that this description of the base component is consistent with the reliance information
 5825 provided for the dependent component.

5826 **15.3.4 ACO_DEV.1 Functional Description**

5827 Dependencies: ACO_REL.1 Basic reliance information

5828 **Objectives**

5829 A description of the interfaces in the base component, on which the dependent component relies, is
 5830 required. This is examined to determine whether or not it is consistent with the description of
 5831 interfaces on which the dependent component relies, as provided in the reliance information.

5832 **Developer action elements**

5833 **ACO_DEV.1.1D**

5834 **The developer shall provide development information for the base component.**

5835 **Content and presentation elements**

5836 **ACO_DEV.1.1C**

5837 **The development information shall describe the purpose of each interface of the base**
 5838 **component used in the composed TOE.**

5839 **ACO_DEV.1.2C**

5840 **The development information shall show correspondence between the interfaces, used in the**
 5841 **composed TOE, of the base component and the dependent component to support the TSF of the**
 5842 **dependent component.**

5843 **Evaluator action elements**5844 **ACO_DEV.1.1E**

5845 **The evaluator shall confirm that the information meets all requirements for content and**
 5846 **presentation of evidence.**

5847 **ACO_DEV.1.2E**

5848 **The evaluator shall determine that the interface description provided is consistent with the**
 5849 **reliance information provided for the dependent component.**

5850 **15.3.5 ACO_DEV.2 Basic evidence of design**

5851 Dependencies: ACO_REL.1 Basic reliance information

5852 **Objectives**

5853 A description of the interfaces in the base component, on which the dependent component relies, is
 5854 required. This is examined to determine whether or not it is consistent with the description of
 5855 interfaces on which the dependent component relies, as provided in the reliance information.

5856 In addition, the security behaviour of the base component that supports the dependent component
 5857 TSF is described.

5858 **Developer action elements**5859 **ACO_DEV.2.1D**

5860 The developer shall provide development information for the base component.

5861 **Content and presentation elements**5862 **ACO_DEV.2.1C**

5863 The development information shall describe the purpose **and method of use of** each interface of the
 5864 base component used in the composed TOE.

5865 **ACO_DEV.2.2C**

5866 **The development information shall provide a high-level description of the behaviour of the**
 5867 **base component, which supports the enforcement of the dependent component SFRs.**

5868 **ACO_DEV.2.3C**

5869 The development information shall show correspondence between the interfaces, used in the
 5870 composed TOE, of the base component and the dependent component to support the TSF of the
 5871 dependent component.

5872 **Evaluator action elements**

5873 **ACO_DEV.2.1E**

5874 The evaluator shall confirm that the information meets all requirements for content and presentation
5875 of evidence.

5876 **ACO_DEV.2.2E**

5877 The evaluator shall determine that the interface description provided is consistent with the reliance
5878 information provided for the dependent component.

5879 **15.3.6 ACO_DEV.3 Detailed evidence of design**

5880 Dependencies: ACO_REL.2 Reliance information

5881 **Objectives**

5882 A description of the interfaces in the base component, on which the dependent component relies, is
5883 required. This is examined to determine whether or not it is consistent with the description of
5884 interfaces on which the dependent component relies, as provided in the reliance information.

5885 The interface description of the architecture of the base component is provided to enable the evaluator
5886 to determine whether or not that interface formed part of the TSF of the base component.

5887 **Developer action elements**

5888 **ACO_DEV.3.1D**

5889 The developer shall provide development information for the base component.

5890 **Content and presentation elements**

5891 **ACO_DEV.3.1C**

5892 The development information shall describe the purpose and method of use of each interface of the
5893 base component used in the composed TOE.

5894 **ACO_DEV.3.2C**

5895 **The development information shall identify the subsystems of the base component that**
5896 **provide interfaces of the base component used in the composed TOE.**

5897 **ACO_DEV.3.3C**

5898 The development information shall provide a high-level description of the behaviour of the base
5899 component **subsystems**, which **support** the enforcement of the dependent component SFRs.

5900 **ACO_DEV.3.4C**

5901 **The development information shall provide a mapping from the interfaces to the subsystems of**
5902 **the base component.**

5903 **ACO_DEV.3.5C**

5904 The development information shall show correspondence between the interfaces, used in the
 5905 composed TOE, of the base component and the dependent component to support the TSF of the
 5906 dependent component.

5907 **Evaluator action elements**5908 **ACO_DEV.3.1E**

5909 The evaluator shall confirm that the information meets all requirements for content and presentation
 5910 of evidence.

5911 **ACO_DEV.3.2E**

5912 The evaluator shall determine that the interface description provided is consistent with the reliance
 5913 information provided for the dependent component.

5914 **15.4 Reliance of dependent component (ACO_REL)**5915 **15.4.1 Objectives**

5916 The purpose of this family is to provide evidence that describes the reliance that a dependent
 5917 component has upon the base component. This information is useful to persons responsible for
 5918 integrating the component with other evaluated IT components to form the composed TOE, and for
 5919 providing insight into the security properties of the resulting composition.

5920 This provides a description of the interface between the dependent and base components of the
 5921 composed TOE that may not have been analysed during evaluation of the individual components, as
 5922 the interfaces were not TSFIs of the individual component TOEs.

5923 **15.4.2 Component levelling**

5924 The components in this family are levelled according to the amount of detail provided in the
 5925 description of the reliance by the dependent component upon the base component.

5926 **15.4.3 Application notes**

5927 The Reliance of dependent component (ACO_REL) family considers the interactions between the
 5928 components where the dependent component relies upon a service from the base component to
 5929 support the operation of security functionality of the dependent component. The interfaces into these
 5930 services of the base component may not have been considered during evaluation of the base
 5931 component because the service in the base component was not considered security-relevant in the
 5932 component evaluation, either because of the inherent purpose of the service (e.g., adjust type font) or
 5933 because associated ISO/IEC 15408-2 SFRs are not being claimed in the base component's ST (e.g. the
 5934 login interface when no FIA: Identification and authentication SFRs are claimed). These interfaces into
 5935 the base component are often viewed as functional interfaces in the evaluation of the base component,
 5936 and are in addition to the security interfaces (TSFI) considered in the functional specification.

5937 In summary, the TSFIs described in the functional specification only include the calls made into a TSF
 5938 by external entities and responses to those calls. Calls made by a TSF, which were not explicitly
 5939 considered during evaluation of the components, are described by the reliance information provided
 5940 to satisfy Reliance of dependent component (ACO_REL).

5941 **15.4.4 ACO_REL.1 Basic reliance information**

5942 Dependencies: No dependencies.

5943 **Developer action elements**

5944 **ACO_REL.1.1D**

5945 **The developer shall provide reliance information of the dependent component.**

5946 **Content and presentation elements**

5947 **ACO_REL.1.1C**

5948 **The reliance information shall describe the functionality of the base component hardware,**
5949 **firmware and/or software that is relied upon by the dependent component TSF.**

5950 **ACO_REL.1.2C**

5951 **The reliance information shall describe all interactions through which the dependent**
5952 **component TSF requests services from the base component.**

5953 **ACO_REL.1.3C**

5954 **The reliance information shall describe how the dependent TSF protects itself from**
5955 **interference and tampering by the base component.**

5956 **Evaluator action elements**

5957 **ACO_REL.1.1E**

5958 **The evaluator shall confirm that the information provided meets all requirements for content**
5959 **and presentation of evidence.**

5960 **15.4.5 ACO_REL.2 Reliance information**

5961 Dependencies: No dependencies.

5962 **Developer action elements**

5963 **ACO_REL.2.1D**

5964 The developer shall provide reliance information of the dependent component.

5965 **Content and presentation elements**

5966 **ACO_REL.2.1C**

5967 The reliance information shall describe the functionality of the base component hardware, firmware
5968 and/or software that is relied upon by the dependent component TSF.

5969 **ACO_REL.2.2C**

5970 The reliance information shall describe all interactions through which the dependent component TSF
5971 requests services from the base component.

5972 **ACO_REL.2.3C**

5973 **The reliance information shall describe each interaction in terms of the interface used and the**
 5974 **return values from those interfaces.**

5975 **ACO_REL.2.4C**

5976 The reliance information shall describe how the dependent TSF protects itself from interference and
 5977 tampering by the base component.

5978 **Evaluator action elements**5979 **ACO_REL.2.1E**

5980 The evaluator shall confirm that the information provided meets all requirements for content and
 5981 presentation of evidence.

5982 **15.5 Composed TOE testing (ACO_CTT)**5983 **15.5.1 Objectives**

5984 This family requires that testing of composed TOE and testing of the base component, as used in the
 5985 composed TOE, is performed.

5986 **15.5.2 Component levelling**

5987 The components in this family are levelled on the basis of increasing rigour of interface testing and
 5988 increasing rigour of the analysis of the sufficiency of the tests to demonstrate that the composed TSF
 5989 operates in accordance with the reliance information and the composed TOE SFRs.

5990 **15.5.3 Application notes**

5991 There are two distinct aspects of testing associated with this family:

5992 a) testing of the interfaces between the base component and the dependent component, which the
 5993 dependent component rely upon for enforcement of security functionality, to demonstrate their
 5994 compatibility;

5995 b) testing of the composed TOE to demonstrate that the TOE behaves in accordance with the SFRs for
 5996 the composed TOE.

5997 If the test configurations used during evaluation of the dependent component included use of the base
 5998 component as a “platform” and the test analysis sufficiently demonstrates that the TSF behaves in
 5999 accordance with the SFRs, the developer need perform no further testing of the composed TOE
 6000 functionality. However, if the base component was not used in the testing of the dependent
 6001 component, or the configuration of either component varied, then the developer is to perform testing
 6002 of the composed TOE. This may take the form of repeating the dependent component developer
 6003 testing of the dependent component, provided this adequately demonstrates the composed TOE TSF
 6004 behaves in accordance with the SFRs.

6005 The developer is to provide evidence of testing the base component interfaces used in the
 6006 composition. The operation of base component TSFIs would have been tested as part of the ATE: Tests
 6007 activities during evaluation of the base component. Therefore, provided the appropriate interfaces
 6008 were included within the test sample of the base component evaluation and it was determined in
 6009 Composition rationale (ACO_COR) that the base component is operating in accordance with the base

6010 component evaluated configuration, with all security functionality required by the dependent
 6011 component included in the TSF, the evaluator action ACO_CTT.1.1E may be met through reuse of the
 6012 base component ATE: Tests verdicts.

6013 If this is not the case, the base component interfaces used relevant to the composition that are affected
 6014 by any variations to the evaluated configuration and any additional security functionality will be tested
 6015 to ensure they demonstrate the expected behaviour. The expected behaviour to be tested is that
 6016 described in the reliance information (Reliance of dependent component (ACO_REL) evidence).

6017 **15.5.4 ACO_CTT.1 Interface testing**

6018 Dependencies: ACO_REL.1 Basic reliance information

6019 ACO_DEV.1 Functional Description

6020 **Objectives**

6021 The objective of this component is to ensure that each interface of the base component, on which the
 6022 dependent component relies, is tested.

6023 **Developer action elements**

6024 **ACO_CTT.1.1D**

6025 **The developer shall provide composed TOE test documentation.**

6026 **ACO_CTT.1.2D**

6027 **The developer shall provide base component interface test documentation.**

6028 **ACO_CTT.1.3D**

6029 **The developer shall provide the composed TOE for testing.**

6030 **ACO_CTT.1.4D**

6031 **The developer shall provide an equivalent set of resources to those that were used in the base**
 6032 **component developer's functional testing of the base component.**

6033 **Content and presentation elements**

6034 **ACO_CTT.1.1C**

6035 **The composed TOE and base component interface test documentation shall consist of test**
 6036 **plans, expected test results and actual test results.**

6037 **ACO_CTT.1.2C**

6038 **The test documentation from the developer execution of the composed TOE tests shall**
 6039 **demonstrate that the TSF behaves as specified.**

6040 **ACO_CTT.1.3C**

6041 **The test documentation from the developer execution of the base component interface tests**
 6042 **shall demonstrate that the base component interface relied upon by the dependent component**
 6043 **behaves as specified.**

6044 **ACO_CTT.1.4C**

6045 **The base component shall be suitable for testing.**

6046 **Evaluator action elements**6047 **ACO_CTT.1.1E**

6048 **The evaluator shall confirm that the information provided meets all requirements for content**
 6049 **and presentation of evidence.**

6050 **ACO_CTT.1.2E**

6051 **The evaluator shall execute a sample of test in the test documentation to verify the developer**
 6052 **test results.**

6053 **ACO_CTT.1.3E**

6054 **The evaluator shall test a subset of the TSF interfaces of the composed TOE to confirm that the**
 6055 **composed TSF operates as specified.**

6056 **15.5.5 ACO_CTT.2 Rigorous interface testing**

6057 Dependencies: ACO_REL.2 Reliance information

6058 ACO_DEV.2 Basic evidence of design

6059 **Objectives**

6060 The objective of this component is to ensure that each interface of the base component, on which the
 6061 dependent component relies, is tested.

6062 **Developer action elements**6063 **ACO_CTT.2.1D**

6064 The developer shall provide composed TOE test documentation.

6065 **ACO_CTT.2.2D**

6066 The developer shall provide base component interface test documentation.

6067 **ACO_CTT.2.3D**

6068 The developer shall provide the composed TOE for testing.

6069 **ACO_CTT.2.4D**

6070 The developer shall provide an equivalent set of resources to those that were used in the base
 6071 component developer's functional testing of the base component.

6072 **Content and presentation elements**

6073 **ACO_CTT.2.1C**

6074 The composed TOE and base component interface test documentation shall consist of test plans,
6075 expected test results and actual test results.

6076 **ACO_CTT.2.2C**

6077 The test documentation from the developer execution of the composed TOE tests shall demonstrate
6078 that the TSF behaves as specified **and is complete**.

6079 **ACO_CTT.2.3C**

6080 The test documentation from the developer execution of the base component interface tests shall
6081 demonstrate that the base component interface relied upon by the dependent component behaves as
6082 specified **and is complete**.

6083 **ACO_CTT.2.4C**

6084 The base component shall be suitable for testing.

6085 **Evaluator action elements**

6086 **ACO_CTT.2.1E**

6087 The evaluator shall confirm that the information provided meets all requirements for content and
6088 presentation of evidence.

6089 **ACO_CTT.2.2E**

6090 The evaluator shall execute a sample of test in the test documentation to verify the developer test
6091 results.

6092 **ACO_CTT.2.3E**

6093 The evaluator shall test a subset of the TSF interfaces of the composed TOE to confirm that the
6094 composed TSF operates as specified.

6095 **15.6 Composition vulnerability analysis (ACO_VUL)**

6096 **15.6.1 Objectives**

6097 This family calls for an analysis of vulnerability information available in the public domain and of
6098 vulnerabilities that may be introduced as a result of the composition.

6099 **15.6.2 Component levelling**

6100 The components in this family are levelled on the basis of increasing scrutiny of vulnerability
6101 information from the public domain and independent vulnerability analysis.

6102 **15.6.3 Application notes**

6103 The developer will provide details of any residual vulnerabilities reported during evaluation of the
6104 components. These may be gained from the component developers or evaluation reports for the

6105 components. These will be used as inputs into the evaluator's vulnerability analysis of the composed
6106 TOE in the operational environment.

6107 The operational environment of the composed TOE is examined to ensure that the assumptions and
6108 objectives for the component operational environment (specified in each component ST) are satisfied
6109 in the composed TOE. An initial analysis of the consistency of assumptions and objectives between the
6110 components and the composed TOE STs will have been performed during the conduct of the ASE
6111 activities for the composed TOE. However, this analysis is revisited with the knowledge acquired
6112 during the ACO_REL, ACO_DEV and the ACO_COR activities to ensure that, for example, assumptions of
6113 the dependent component that were addressed by the environment in the dependent component ST
6114 are not reintroduced as a result of composition (i.e. that the base component adequately addresses the
6115 assumptions of the dependent component ST in the composed TOE).

6116 A search by the evaluator for issues in each component will identify potential vulnerabilities reported
6117 in the public domain since completion of the evaluation of the components. Any potential
6118 vulnerabilities will then be subject to testing.

6119 If the base component used in the composed TOE has been the subject of assurance continuity
6120 activities since certification, the evaluator will consider during the composed TOE vulnerability
6121 analysis activities the changes made in base component.

6122 **15.6.4 ACO_VUL.1 Composition vulnerability review**

6123 Dependencies: ACO_DEV.1 Functional Description

6124 **Developer action elements**

6125 **ACO_VUL.1.1D**

6126 **The developer shall provide the composed TOE for testing.**

6127 **Content and presentation elements**

6128 **ACO_VUL.1.1C**

6129 **The composed TOE shall be suitable for testing.**

6130 **Evaluator action elements**

6131 **ACO_VUL.1.1E**

6132 **The evaluator shall confirm that the information provided meets all requirements for content**
6133 **and presentation of evidence.**

6134 **ACO_VUL.1.2E**

6135 **The evaluator shall perform an analysis to determine that any residual vulnerabilities**
6136 **identified for the base and dependent components are not exploitable in the composed TOE in**
6137 **its operational environment.**

6138 **ACO_VUL.1.3E**

6139 **The evaluator shall perform a search of public domain sources to identify possible**
6140 **vulnerabilities arising from use of the base and dependent components in the composed TOE**
6141 **operational environment.**

6142 **ACO_VUL.1.4E**

6143 **The evaluator shall conduct penetration testing, based on the identified vulnerabilities, to**
 6144 **demonstrate that the composed TOE is resistant to attacks by an attacker with basic attack**
 6145 **potential.**

6146 **15.6.5 ACO_VUL.2 Composition vulnerability analysis**

6147 Dependencies: ACO_DEV.2 Basic evidence of design

6148 **Developer action elements**6149 **ACO_VUL.2.1D**

6150 The developer shall provide the composed TOE for testing.

6151 **Content and presentation elements**6152 **ACO_VUL.2.1C**

6153 The composed TOE shall be suitable for testing.

6154 **Evaluator action elements**6155 **ACO_VUL.2.1E**

6156 The evaluator shall confirm that the information provided meets all requirements for content and
 6157 presentation of evidence.

6158 **ACO_VUL.2.2E**

6159 The evaluator shall perform an analysis to determine that any residual vulnerabilities identified for
 6160 the base and dependent components are not exploitable in the composed TOE in its operational
 6161 environment.

6162 **ACO_VUL.2.3E**

6163 The evaluator shall perform a search of public domain sources to identify possible vulnerabilities
 6164 arising from use of the base and dependent components in the composed TOE operational
 6165 environment.

6166 **ACO_VUL.2.4E**

6167 **The evaluator shall perform an independent vulnerability analysis of the composed TOE, using**
 6168 **the guidance documentation, reliance information and composition rationale to identify**
 6169 **potential vulnerabilities in the composed TOE.**

6170 **ACO_VUL.2.5E**

6171 The evaluator shall conduct penetration testing, based on the identified vulnerabilities, to demonstrate
 6172 that the composed TOE is resistant to attacks by an attacker with basic attack potential.

6173 **15.6.6 ACO_VUL.3 Enhanced-Basic Composition vulnerability analysis**

6174 Dependencies: ACO_DEV.3 Detailed evidence of design

6175 **Developer action elements**

6176 **ACO_VUL.3.1D**

6177 The developer shall provide the composed TOE for testing.

6178 **Content and presentation elements**

6179 **ACO_VUL.3.1C**

6180 The composed TOE shall be suitable for testing.

6181 **Evaluator action elements**

6182 **ACO_VUL.3.1E**

6183 The evaluator shall confirm that the information provided meets all requirements for content and
6184 presentation of evidence.

6185 **ACO_VUL.3.2E**

6186 The evaluator shall perform an analysis to determine that any residual vulnerabilities identified for
6187 the base and dependent components are not exploitable in the composed TOE in its operational
6188 environment.

6189 **ACO_VUL.3.3E**

6190 The evaluator shall perform a search of public domain sources to identify possible vulnerabilities
6191 arising from use of the base and dependent components in the composed TOE operational
6192 environment.

6193 **ACO_VUL.3.4E**

6194 The evaluator shall perform an independent vulnerability analysis of the composed TOE, using the
6195 guidance documentation, reliance information and composition rationale to identify potential
6196 vulnerabilities in the composed TOE.

6197 **ACO_VUL.3.5E**

6198 The evaluator shall conduct penetration testing, based on the identified vulnerabilities, to demonstrate
6199 that the composed TOE is resistant to attacks by an attacker with **Enhanced-Basic** attack potential.

Annex A (informative)

Development (ADV)

6204 This annex contains ancillary material to further explain and provide additional examples for the
6205 topics brought up in families of the ADV: Development class.

6206 A.1 ADV_ARC: Supplementary material on security architectures

6207 A security architecture is a set of properties that the TSF exhibits; these properties include self-
6208 protection, domain separation, and non-bypassability. Having these properties provides a basis of
6209 confidence that the TSF is providing its security services. This annex provides additional material on
6210 these properties, as well as discussion on contents of a security architecture description.

6211 The remainder of this clause first explains these properties, then discusses the kinds of information
6212 that are needed to describe how the TSF exhibits those properties.

6213 A.1.1 Security architecture properties

6214 *Self-protection* refers to the ability of the TSF to protect itself from manipulation from external entities
6215 that may result in changes to the TSF. Without these properties, the TSF might be disabled from
6216 performing its security services.

6217 It is oftentimes the case that a TOE uses services or resources supplied by other IT entities in order to
6218 perform its functions (e.g. an application that relies upon its underlying operating system). In these
6219 cases, the TSF does not protect itself entirely on its own, because it depends on the other IT entities to
6220 protect the services it uses.

6221 *Domain separation* is a property whereby the TSF creates separate *security domains* for each untrusted
6222 active entity to operate on its resources, and then keeps those domains separated from one another so
6223 that no entity can run in the domain of any other. For example, an operating system TOE supplies a
6224 domain (address space, per-process environment variables) for each process associated with
6225 untrusted entities.

6226 For some TOEs such domains do not exist because all of the actions of the untrusted entities are
6227 brokered by the TSF. A packet-filter firewall is an example of such a TOE, where there are no untrusted
6228 entity domains; there are only data structures maintained by the TSF. The existence of domains, then,
6229 is dependant upon 1) the type of TOE and 2) the SFRs levied on the TOE. In the cases where the TOE
6230 does provide domains for untrusted entities, this family requires that those domains are isolated from
6231 one another such that untrusted entities in one domain are prevented from tampering (affecting
6232 without brokering by the TSF) from another untrusted entity's domain.

6233 *Non-bypassability* is a property that the security functionality of the TSF (as specified by the SFRs) is
6234 always invoked and cannot be circumvented when appropriate for that specific mechanism. For
6235 example, if access control to files is specified as a capability of the TSF via an SFR, there must be no
6236 interfaces through which files can be accessed without invoking the TSF's access control mechanism
6237 (an interface through which a raw disk access takes place might be an example of such an interface).

6238 As is the case with self-protection, the very nature of some TOEs might depend upon their
 6239 environments to play a role in non-bypassability of the TSF. For example, a security application TOE
 6240 requires that it be invoked by the underlying operating system. Similarly, a firewall depends upon the
 6241 fact that there are no direct connections between the internal and external networks and that all traffic
 6242 between them must go through the firewall.

6243 **A.1.2 Security architecture descriptions**

6244 The security architecture description explains how the properties described above are exhibited by
 6245 the TSF. It describes how domains are defined and how the TSF keeps them separate. It describes what
 6246 prevents untrusted processes from getting to the TSF and modifying it. It describes what ensures that
 6247 all resources under the TSF's control are adequately protected and that all actions related to the SFRs
 6248 are mediated by the TSF. It explains any role the environment plays in any of these (e.g. presuming it
 6249 gets correctly invoked by its underlying environment, how are its security functions invoked?).

6250 The security architecture description presents the TSF's properties of self-protection, domain
 6251 separation, and non-bypassability in terms of the decomposition descriptions. The level of this
 6252 description is commensurate with the TSF description required by the ADV_FSP, ADV_TDS and
 6253 ADV_IMP requirements that are being claimed. For example, if ADV_FSP is the only TSF description
 6254 available, it would be difficult to provide any meaningful security architecture description because
 6255 none of the details of any internal workings of the TSF would be available.

6256 However, if the TOE design were also available, even at the most basic level (ADV_TDS.1), there would
 6257 be some information available concerning the subsystems that make up the TSF, and there would be a
 6258 description of how they work to implement self-protection, domain separation, and non-bypassability.
 6259 For example, perhaps all user interaction with the TOE is constrained through a process that acts on
 6260 that user's behalf, adopting all of the user's security attributes; the security architecture description
 6261 would describe how such a process comes into being, how the process's behaviour is constrained by
 6262 the TSF (so it cannot corrupt the TSF), how all actions of that process are mediated by the TSF
 6263 (thereby explaining why the TSF cannot be bypassed), etc.

6264 If the available TOE design is more detailed (e.g. at the modular level), or the implementation
 6265 representation is also available, then the security architecture description would be correspondingly
 6266 more detailed, explaining how the user's process communicate with the TSF processes, how different
 6267 requests are processed by the TSF, what parameters are passed, what programmatic protections
 6268 (buffer overflow prevention, parameter bounds checking, time of check/time of use checking, etc.) are
 6269 in place. Similarly, a TOE whose ST claimed the ADV_IMP component would go into implementation-
 6270 specific detail.

6271 The explanations provided in the security architecture description are expected to be of sufficient
 6272 detail that one would be able to test their accuracy. That is, simple assertions (e.g. "The TSF keeps
 6273 domains separate") provide no useful information to convince the reader that the TSF does indeed
 6274 create and separate domains.

6275 **A.1.2.1 Domain Separation**

6276 In cases where the TOE exhibits domain separation entirely on its own, there would be a
 6277 straightforward description of how this is attained. The security architecture description would
 6278 explain the different kinds of domains that are defined by the TSF, how they are defined (i.e. what
 6279 resources are allocated to each domain), how no resources are left unprotected, and how the domains
 6280 are kept separated so that active entities in one domain cannot tamper with resources in another
 6281 domain.

6282 For cases where the TOE depends upon other IT entities to play a role in domain separation, that
 6283 sharing of roles must be made clear. For example, a TOE that is solely application software relies upon
 6284 the underlying operating system to correctly instantiate the domains that the TOE defines; if the TOE
 6285 defines separate processing space, memory space, etc, for each domain, it depends upon the
 6286 underlying operating system to operate correctly and benignly (e.g. allow the process to execute only
 6287 in the execution space that is requested by the TOE software).

6288 For example, mechanisms that implement domain separation (e.g., memory management, protected
 6289 processing modes provided by the hardware, etc.) would be identified and described. Or, the TSF
 6290 might implement software protection constructs or coding conventions that contribute to
 6291 implementing separation of software domains, perhaps by delineating user address space from system
 6292 address space.

6293 The vulnerability analysis and testing (see AVA_VAN) activities will likely include attempts to defeat
 6294 the described TSF domain separation through the use of monitoring or direct attack the TSF.

6295 **A.1.2.2 TSF Self-protection**

6296 In cases where the TOE exhibits self-protection entirely on its own, there would be a straightforward
 6297 description of how this self-protection is attained. Mechanisms that provide domain separation to
 6298 define a TSF domain that is protected from other (user) domains would be identified and described.

6299 For cases where the TOE depends upon other IT entities to play a role in protecting itself, that sharing
 6300 of roles must be made clear. For example, a TOE that is solely application software relies upon the
 6301 underlying operating system to operate correctly and benignly; the application cannot protect itself
 6302 against a malicious operating system that subverts it (for example, by overwriting its executable code
 6303 or TSF data).

6304 The security architecture description also covers how user input is handled by the TSF in such a way
 6305 that the TSF does not subject itself to being corrupted by that user input. For example, the TSF might
 6306 implement the notion of privilege and protect itself by using privileged-mode routines to handle user
 6307 data. The TSF might make use of processor-based separation mechanisms (e.g. privilege levels or
 6308 rings) to separate TSF code and data from user code and data. The TSF might implement software
 6309 protection constructs or coding conventions that contribute to implementing separation of software,
 6310 perhaps by delineating user address space from system address space.

6311 For TOEs that start up in a low-function mode (for example, a single-user mode accessible only to
 6312 installers or administrators) and then transition to the evaluated secure configuration (a mode
 6313 whereby untrusted users are able to login and use the services and resources of the TOE), the security
 6314 architecture description also includes an explanation of how the TSF is protected against this
 6315 initialisation code that does not run in the evaluated configuration. For such TOEs, the security
 6316 architecture description would explain what prevents those services that should be available only
 6317 during initialisation (e.g. direct access to resources) from being accessible in the evaluated
 6318 configuration. It would also explain what prevents initialisation code from running while the TOE is in
 6319 the evaluated configuration.

6320 There must also be an explanation of how the trusted initialisation code will maintain the integrity of
 6321 the TSF (and of its initialisation process) such that the initialisation process is able to detect any
 6322 modification that would result in the TSF being spoofed into believe it was in an initial secure state.

6323 The vulnerability analysis and testing (see AVA_VAN) activities will likely include attempts to defeat
 6324 the described TSF self protection through the use of tampering, direct attack, or monitoring of the TSF.

6325 **A.1.2.3 TSF Non-Bypassability**

6326 The property of non-bypassability is concerned with interfaces that permit the bypass of the
 6327 enforcement mechanisms. In most cases this is a consequence of the implementation, where if a
 6328 programmer is writing an interface that accesses or manipulates an object, it is that programmer's
 6329 responsibility to use interfaces that are part of the SFR enforcement mechanism for the object and not
 6330 to try to circumvent those interfaces. For the description pertaining to non-bypassability, then, there
 6331 are two broad areas that have to be covered.

6332 The first consists of those interfaces to the SFR-enforcement. The property for these interfaces is that
 6333 they contain no operations or modes that allow them to be used to bypass the TSF. It is likely that the
 6334 evidence for ADV_FSP and ADV_TDS can be used in large part to make this determination. Because
 6335 non-bypassability is the concern, if only certain operations available through these TSFIs are
 6336 documented (because they are SFR-enforcing) and others are not, the developer should consider
 6337 whether additional information (to that presented in ADV_FSP and ADV_TDS) is necessary to make a
 6338 determination that the SFR-supporting and SFR-non-interfering operations of the TSFI do not afford
 6339 an untrusted entity the ability to bypass the policy being enforced. If such information is necessary, it
 6340 is included in the security architecture description.

6341 The second area of non-bypassability is concerned with those interfaces whose interactions are not
 6342 associated with SFR-enforcement. Depending on the ADV_FSP and ADV_TDS components claimed,
 6343 some information about these interfaces may or may not exist in the functional specification and TOE
 6344 design documentation. The information presented for such interfaces (or groups of interfaces) should
 6345 be sufficient so that a reader can make a determination (at the level of detail commensurate with the
 6346 rest of the evidence supplied in the ADV: Development class) that the enforcement mechanisms cannot
 6347 be bypassed.

6348 The property that the security functionality cannot be bypassed applies to all security functionality
 6349 equally. That is, the design description should cover objects that are protected under the SFRs (e.g.
 6350 FDP_* components) and functionality (e.g., audit) that is provided by the TSF. The description should
 6351 also identify the interfaces that are associated with security functionality; this might make use of the
 6352 information in the functional specification. This description should also describe any design
 6353 constructs, such as object managers, and their method of use. For instance, if routines are to use a
 6354 standard macro to produce an audit record, this convention is a part of the design that contributes to
 6355 the non-bypassability of the audit mechanism. It is important to note that *non-bypassability* in this
 6356 context is not an attempt to answer the question "could a part of the TSF implementation, if malicious,
 6357 bypass the security functionality", but rather to document how the implementation does not bypass
 6358 the security functionality.

6359 The vulnerability analysis and testing (see AVA_VAN) activities will likely include attempts to defeat
 6360 the described non-bypassability by circumventing the TSF.

6361 **A.2 ADV_FSP: Supplementary material on functional specification**

6362 The purpose in specifying the TSFIs is to provide the necessary information to conduct testing;
 6363 without knowing the possible means interact with the TSF, one cannot adequately test the behaviour
 6364 of the TSF.

6365 There are two parts to specifying the TSFIs: identifying them and describing them. Because of the
 6366 diversity of possible TOEs, and of different TSFs therein, there is no standard set of interfaces that
 6367 constitute "TSFIs". This annex provides guidance on the factors that determine which interfaces are
 6368 TSFIs.

6369 **A.2.1 Non-TSF part of the TOE**

6370 The TSF comprises all parts of the TOE the user has to rely on in order to trust the security
6371 functionality.

6372 To say it in other words: Those parts of the TOE that do not belong to the TSF can be modified by an
6373 attacker without any impact on the TOE security functionality. If this isn't the case, these parts of the
6374 TOE have to be included in the TSF.

6375 If the TSF and the TSF implementation are defined then it is clear whether there exist further parts of
6376 the TOE which can be classified as non-TSF parts of the TOE. Such parts do not have to be part of the
6377 TSF but they are still part of the TOE.

6378 The relationship between TSF and non-TSF parts of TOE is given by their definitions and the ARC
6379 properties as follows:

- 6380 • non-TSF parts do not bypass the TSF and
- 6381 • parts of the TSF protects themselves against tampering.

6382 A subsystem of the TOE which is not part of the TSF has to fulfil the following condition (described as a
6383 rule of thumb¹²): The subsystem must not have any security impact of the TOE even if it were
6384 substituted by an attacker.

6385 Therefore between the Non-TSF parts and the TSF parts it seems that some kind of "separation
6386 mechanism" is advisable¹³ because such "separation mechanism" may build the basis for the
6387 assessment that there is no impact on the TSF parts from the Non-TSF parts possible.

6388 Such "separation mechanism" could be implemented by the security architecture or by an explicitly
6389 realised part of the implementation (e.g. a firewall between TSF and Non-TSF parts of the TOE).

6390 The analysis of the "separation mechanism" is then subject of the vulnerability assessment because it
6391 must withstand attacks by an attacker of the respective strength according to the VAN level of the
6392 evaluation.

6393 The developer shall provide evidence for non-bypassability and self-protection in its security
6394 architecture description and the evaluator shall analyse this evidence in subactivity for ADV_ARC.1
6395 and assess the effectiveness in the vulnerability assessment.

6396 The goal of TOE design documentation is to provide sufficient information to determine the TSF
6397 boundary, and to describe how the TSF implements the SFR. Further attention is needed by the fact
6398 that the family ADV_TDS requires only identification of the non-TSF subsystems of the TOE. No
6399 interface description is provided for these subsystems in ADV_FSP or ADV_TDS. SFR non-interference
6400 of these subsystems is assumed but not demonstrated by the developer and not examined in details by
6401 the evaluator. However from the TOE design point of view this is not that important as long as the

¹² This rule is only valid to some extent because the actual requirement "The Non-TSF part must not bypass the TSF." is not that strong as the given rule of thumb.

¹³ The "separation mechanism" is only an proposal here. The developer is free to provide evidence using other kind of security implementation as long as the requirement showing the non-bypassability for the TSF part of the TOE from the non-TSF part of the TOE is fulfilled.

6402 above mentioned separation mechanism is in place and the vulnerability assessment confirms that it is
 6403 strong enough. Therefore this "separation mechanism" implements the TSF or enforces ARC
 6404 properties as security feature. But non-bypassability may be enforced by "pure architecture
 6405 properties" as well.

6406 Parts of the TOE classified as non-TSF must not provide means to bypass the TSF (no matter whether a
 6407 valid user or even an attacker makes uses of those parts) and must not contribute to the TSF. It is
 6408 important that the developer provides clear evidence and demonstrate how this requirement is
 6409 fulfilled.

6410 Therefore the developer shall demonstrate and the evaluator shall examine that the TOE identification
 6411 of subsystems as non-TSF (cf. ADV_TDS.x.1) is correct and consequently no detailed description of
 6412 these subsystems is necessary. The evaluator examination shall include the ARC properties non-
 6413 bypassability and self-protection being described in the ADV_ARC documentation provided by the
 6414 developer (see the paragraphs above).

6415 **A.2.2 Determining the TSFI**

6416 In order to identify the interfaces to the TSF, the parts of the TOE that make up the TSF must first be
 6417 identified. This identification is actually a part of the TOE design (ADV_TDS) analysis, but is also
 6418 performed implicitly (through identification and description of the TSFI) by the developer in cases
 6419 where TOE design (ADV_TDS) is not included in the assurance package. In this analysis, a portion of
 6420 the TOE must be considered to be in the TSF if it contributes to the satisfaction of an SFR in the ST (in
 6421 whole or in part). This includes, for example, everything in the TOE that contributes to TSF run-time
 6422 initialisation, such as software that runs prior to the TSF being able to protect itself because
 6423 enforcement of the SFRs has not yet begun (e.g., while booting up). Also included in the TSF are all
 6424 parts of the TOE that contribute to the architectural principles of TSF self-protection, domain
 6425 separation, and non-bypassability (see Security Architecture (ADV_ARC)).

6426 Once the TSF has been defined, the TSFI are identified. The TSFI consists of all means by which
 6427 external entities (or subjects in the TOE but outside of the TSF) supply data to the TSF, receive data
 6428 from the TSF and invoke services from the TSF. These service invocations and responses are the
 6429 means of crossing the TSF boundary. While many of these are readily apparent, others might not be as
 6430 obvious. The question that should be asked when determining the TSFIs is: "How can a potential
 6431 attacker interact with the TSF in an attempt to subvert the SFRs?"

6432 Therefore from the evaluation point of view it is also important whether the interface can be misused
 6433 by an attacker to get access to the security functionality in order to compromise the assets protected
 6434 by TSF.

6435 Any interface of the TSF which can be potentially used by an attacker belongs to the TSFI (regardless
 6436 of the further classification as SFR-enforcing, SFR-supporting or SFR-non-interfering).

6437 It is not important whether the TSF will be accessed from outside or whether the TSF accesses the
 6438 external resources (e.g. TSF calls platform or user). The only criteria is whether there is a potential
 6439 interference with the TSF from outside.

6440 The following discussions illustrate the application of the TSFI definition in different contexts.

6441 **A.2.2.1 Electrical interfaces**

6442 In TOEs such as smart cards, where the adversary has not only logical access to the TOE, but also
 6443 complete physical access to the TOE, the TSF boundary is the physical boundary. Therefore, the

6444 exposed electrical interfaces are considered TSFI because their manipulation could affect the
 6445 behaviour of the TSF. As such, all these interfaces (electrical contacts) need to be described: various
 6446 voltages that might be applied, etc.

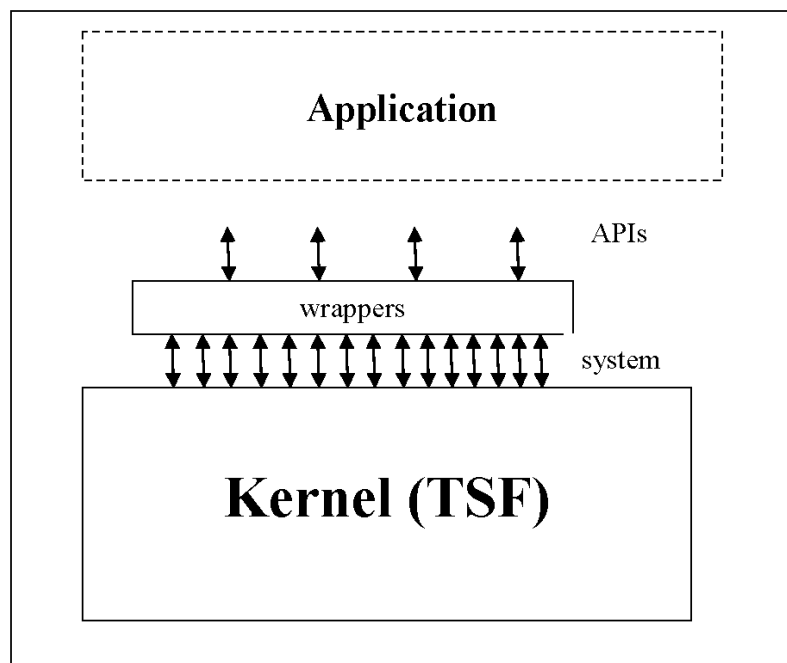
6447 **A.2.2.2 Network protocol stack**

6448 The TSFIs of a TOE that performs protocol processing would be those protocol layers to which a
 6449 potential attacker has direct access. This need not be the entire protocol stack, but it might be.

6450 For example, if the TOE were some sort of a network appliance that allowed potential attackers to
 6451 affect every level of the protocol stack (i.e. to send arbitrary signals, arbitrary voltages, arbitrary
 6452 packets, arbitrary datagrams, etc.), then the TSF boundary exists at each layer of the stack. Therefore,
 6453 the functional specification would have to address every protocol at every layer of the stack.

6454 If, however, the TOE were a firewall that protects an internal network from the Internet, a potential
 6455 attacker would have no means of directly manipulating the voltages that enter the TOE; any extreme
 6456 voltages would simply not be passed though the Internet. That is, the attacker would have access only
 6457 to those protocols at the Internet layer or above. The TSF boundary exists at each layer of the stack.
 6458 Therefore, the functional specification would have to address only those protocols at or above the
 6459 Internet layer: it would describe each of the different communication layers at which the firewall is
 6460 exposed in terms of what constitutes well-formed input for what might appear on the line, and the
 6461 result of both well-formed and malformed inputs. For example, the description of the Internet protocol
 6462 layer would describe what constitutes a well-formed IP packet and what happens when both
 6463 correctly-formed and malformed packets are received. Likewise, the description of the TCP layer
 6464 would describe a successful TCP connection and what happens both when successful connections are
 6465 established and when connections cannot be established or are inadvertently dropped. Presuming the
 6466 firewall's purpose is to filter application-level commands (like FTP or telnet), the description of the
 6467 application layer would describe the application-level commands that are recognised and filtered by
 6468 the firewall, as well as the results of encountering unknown commands.

6469 The descriptions of these layers would likely reference published communication standards (telnet,
 6470 FTP, TCP, etc.) that are used, noting which user-defined options are chosen.

6471 **A.2.2.3 Wrappers**6472
6473 **Figure A.1 — Wrappers**

6474 “Wrappers” translate complex series of interactions into simplified common services, such as when
 6475 Operating Systems create APIs for use by applications (as shown in Figure A.1). Whether the TSFIs
 6476 would be the system calls or the APIs depends upon what is available to the application: if the
 6477 application can use the system calls directly, then the system calls are the TSFIs. If, however, there
 6478 were something that prohibits their direct use and requires all communication through the APIs, then
 6479 the APIs would be the TSFIs.

6480 A Graphical User interface is similar: it translates between machine-understandable commands and
 6481 user-friendly graphics. Similarly, the TSFIs would be the commands if users have access to them, or the
 6482 graphics (pull-down menus, check-boxes, text fields) if the users are constrained to using them.

6483 It is worth noting that, in both of these examples, if the user is prohibited from using the more
 6484 primitive interfaces (i.e. the system calls or the commands), the description of this restriction and of
 6485 its enforcement would be included in the Security Architecture Description (see A.1). Also, the
 6486 wrapper would be part of the TSF.

6487 **A.2.2.4 Inaccessible interfaces**

6488 For a given TOE, not all of the interfaces may be *accessible*. That is, the security objectives for the
 6489 operational environment (in the Security Target) may prevent access to these interfaces or limit
 6490 access in such a way that they are practically inaccessible. Such interfaces would not be considered
 6491 TSFIs. Some examples:

- 6492 a) If the security objectives for the operational environment for the stand-alone firewall state that
 6493 “the firewall will be operational in a server room environment to which only trusted and trained
 6494 personnel will have access, and which will be equipped with an interruptible power supply
 6495 (against power failure)”, physical and power interfaces will not be accessible, since trusted and
 6496 trained personnel will not attempt to dismantle the firewall and/or disable its power supply.

b) If the security objectives for the operational environment for the software firewall (application) state that “the OS and the hardware will provide a security domain for the application free from tampering by other programs”, the interfaces through which the firewall can be accessed by other applications on the OS (e.g. deleting or modifying the firewall executable, direct reading or writing to the memory space of the firewall) will not be accessible, since the OS/hardware part of the operational environment makes this interface inaccessible.

c) If the security objectives for the operational environment for the software firewall additionally state that the OS and hardware will faithfully execute the commands of the TOE, and will not tamper with the TOE in any manner, interfaces through which the firewall obtains primitive functionality from the OS and hardware (executing machine code instructions, OS APIs, such as creating, reading, writing or deleting files, graphical APIs etc.) will not be accessible, since the OS/hardware are the only entities that can access that interface, and they are completely trusted.

For all of these examples, these inaccessible interfaces would not be TSFIs.

A.2.3 Example: A complex DBMS

Figure A.2 illustrates a complex TOE: a database management system that relies on hardware and software that is outside the TOE boundary (referred to as the *IT environment* in the rest of this discussion). To simplify this example, the TOE is identical to the TSF. The shaded boxes represent the TSF, while the unshaded boxes represent IT entities in the environment. The TSF comprises the database engine and management GUIs (represented by the box labelled *DB*) and a kernel module that runs as part of the OS that performs some security function (represented by the box labelled *PLG*). The TSF kernel module has entry points defined by the OS specification that the OS will call to invoke some function (this could be a device driver, or an authentication module, etc.). The key is that this pluggable kernel module is providing security services specified by functional requirements in the ST.

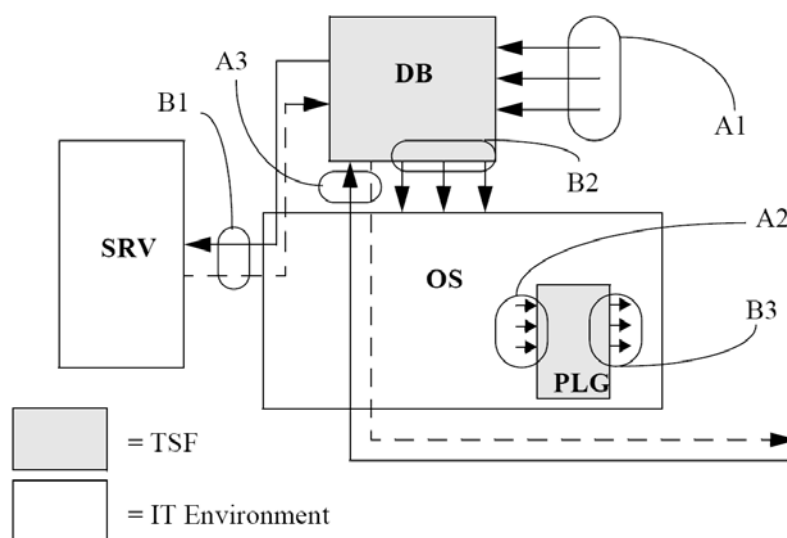


Figure A.2 — Interfaces in a DBMS system

The IT environment consists of the operating system itself (represented by the box labelled *OS*), as well as an external server (labelled *SRV*). This external server, like the OS, provides a service that the TSF depends on, and thus needs to be in the IT environment. Interfaces in the figure are labelled *A_x* for TSFI, and *B_x* for other interfaces that would be documented in ACO: Composition. Each of these groups of interfaces is now discussed.

Interface group A1 represents the most obvious set of TSFI. These are interfaces used by users to directly access the database and its security functionality and resources.

Interface group A2 represent the TSFI that the OS invokes to obtain the functionality provided by the pluggable module. These are contrasted with interface group B3, which represent calls that the pluggable module makes to obtain services from the IT environment.

Interface group A3 represent TSFI that pass through the IT environment. In this case, the DBMS communicates over the network using a proprietary application-level protocol. While the IT environment is responsible for providing various supporting protocols (e.g., Ethernet, IP, TCP), the application layer protocol that is used to obtain services from the DBMS is a TSFI and must be documented as such. The dotted line indicates return values/services from the TSF over the network connection.

The interfaces labelled *Bx* represent interfaces to functionality in the IT Environment. These interfaces are not TSFI and need only be discussed and analysed when the TOE is being used in a composite evaluation as part of the activities associated with the ACO class.

A.2.4 Example Functional Specification

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

The Example firewall presents the following interfaces to the internal network:

a) IP datagrams

b) Administrator Commands

and the following interfaces to the external network:

a) IP datagrams

Interfaces Descriptions: IP Datagrams

The datagrams are in the format specified by RFC 791.

- Purpose - to transmit blocks of data ("datagrams") from source hosts to destination hosts identified by fixed length addresses; also provides for fragmentation and reassembly of long datagrams, if necessary, for transmission through small-packet networks.
- Method of Use - they arrive from the lower-level (e.g. data link) protocol.
- Parameters - the following fields of the IP datagram header: source address, destination address, don't-fragment flag.
- Parameter description - [As defined by RFC 791, subclause 3.1 ("Internet Header Format")]
- Actions - Transmits datagrams that are not masquerading; fragments large datagrams if necessary; reassembles fragments into datagrams.

- 6564 • Error messages - (none). No reliability guaranteed (reliability to be provided by upper-level
- 6565 protocols) Undeliverable datagrams (e.g. must be fragmented for transmission, but don't fragment
- 6566 flag is set) dropped.

6567 Interfaces Descriptions: Administrator Commands

6568 The administrator commands provide a means for the administrator to interact with the firewall.
 6569 These **commands and responses ride atop a telnet (RFC 854) connection established from any**
 6570 **host on the internal** network. Available commands are:

6571 • **Passwd**

- 6572 • Purpose - sets administrator password
- 6573 • Method of Use - **Passwd** <password>
- 6574 • Parameters - password
- 6575 • Parameter description - value of new password
- 6576 • Actions - changes password to new value supplied. There are no restrictions.
- 6577 • Error messages - none.

6578 • **Readaudit**

- 6579 • Purpose - presents the audit log to the administrator
- 6580 • Method of Use - **Readaudit**
- 6581 • Parameters - none
- 6582 • Parameter description - none
- 6583 • Actions - provides the text of the audit log
- 6584 • Error messages - none.

6585 • **Setintaddr**

- 6586 • Purpose - sets the address of the internal address.
- 6587 • Method of Use - **Setintaddr** <address>
- 6588 • Parameters - address
- 6589 • Parameter description - first three fields of an IP address (as defined in RFC 791). For example:
 6590 123.123.123.
- 6591 • Actions - changes the internal value of the variable defining the internal network, the value of
 6592 which is used to judge attempted masquerades.

- 6593 • Error messages - “address in use”: indicates the identified internal network is the same as the
6594 external network.
- 6595 • **Setextaddr**
- 6596 • Purpose - sets the address of the external address
- 6597 • Method of Use - **Setextaddr** <address>
- 6598 • Parameters - address
- 6599 • Parameter description - first three fields of an IP address (as defined in RFC 791). For example:
6600 123.123.123.
- 6601 • Actions - changes the internal value of the variable defining the external network.
- 6602 • Error messages - “address in use”: indicates the identified external network is the same as the
6603 internal network.

6604 **A.3 ADV_INT: Supplementary material on TSF internals**

6605 The wide variety of TOEs makes it impossible to codify anything more specific than “well-structured”
6606 or “minimum complexity”. Judgements on structure and complexity are expected to be derived from
6607 the specific technologies used in the TOE. For example, software is likely to be considered well-
6608 structured if it exhibits the characteristics cited in the software engineering disciplines.

6609 This annex provides supplementary material on assessing the structure and complexity of procedure-
6610 based software portions of the TSF. This material is based on information readily available in software
6611 engineering literature. For other kinds of internals (e.g. hardware, non-procedural software such as
6612 object-oriented code, etc.), corresponding literature on good practises should be consulted.

6613 **A.3.1 Structure of procedural software**

6614 The structure of procedural software is traditionally assessed according to its *modularity*. Software
6615 written with a modular design aids in achieving understandability by clarifying what dependencies a
6616 module has on other modules (*coupling*) and by including in a module only tasks that are strongly
6617 related to each other (*cohesion*). The use of modular design reduces the interdependence between
6618 elements of the TSF and thus reduces the risk that a change or error in one module will have effects
6619 throughout the TOE. Its use enhances clarity of design and provides for increased assurance that
6620 unexpected effects do not occur. Additional desirable properties of modular decomposition are a
6621 reduction in the amount of redundant or unneeded code.

6622 Minimising the amount of functionality in the TSF allows the evaluator as well as the developer to
6623 focus only on that functionality which is necessary for SFR enforcement, contributing further to
6624 understandability and further lowering the likelihood of design or implementation errors.

6625 The incorporation of modular decomposition, layering and minimisation into the design and
6626 implementation process must be accompanied by sound software engineering considerations. A
6627 practical, useful software system will usually entail some undesirable coupling among modules, some
6628 modules that include loosely-related functions, and some subtlety or complexity in a module's design.
6629 These deviations from the ideals of modular decomposition are often deemed necessary to achieve
6630 some goal or constraint, be it related to performance, compatibility, future planned functionality, or
6631 some other factors, and may be acceptable, based on the developer's justification for them. In applying

the requirements of this class, due consideration must be given to sound software engineering principles; however, the overall objective of achieving understandability must be achieved.

A.3.1.1 Cohesion

Cohesion is the manner and degree to which the tasks performed by a single software module are related to one another; types of cohesion include coincidental, communicational, functional, logical, sequential, and temporal. These types of cohesion are characterised below, listed in the order of decreasing desirability.

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

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

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

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

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

f) *coincidental* cohesion - a module with coincidental cohesion performs unrelated, or loosely related, activities.

A.3.1.2 Coupling

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

a) *call*: two modules are call coupled if they communicate strictly through the use of their documented function calls; examples of call coupling are data, stamp, and control, which are defined below.

1) *data*: two modules are data coupled if they communicate strictly through the use of call parameters that represent single data items.

2) *stamp*: two modules are stamp coupled if they communicate through the use of call parameters that comprise multiple fields or that have meaningful internal structures.

3) *control*: two modules are control coupled if one passes information that is intended to influence the internal logic of the other.

b) *common*: two modules are common coupled if they share a common data area or a common system resource. Global variables indicate that modules using those global variables are common coupled. Common coupling through global variables is generally allowed, but only to a limited degree. For example, variables that are placed into a global area, but are used by only a single module, are inappropriately placed, and should be removed. Other factors that need to be considered in assessing the suitability of global variables are:

- 1) The number of modules that modify a global variable: In general, only a single module should be allocated the responsibility for controlling the contents of a global variable, but there may be situations in which a second module may share that responsibility; in such a case, sufficient justification must be provided. It is unacceptable for this responsibility to be shared by more than two modules. (In making this assessment, care should be given to determining the module actually responsible for the contents of the variable; for example, if a single routine is used to modify the variable, but that routine simply performs the modification requested by its caller, it is the calling module that is responsible, and there may be more than one such module). Further, as part of the complexity determination, if two modules are responsible for the contents of a global variable, there should be clear indications of how the modifications are coordinated between them.
- 2) The number of modules that reference a global variable: Although there is generally no limit on the number of modules that reference a global variable, cases in which many modules make such a reference should be examined for validity and necessity.

c) *content*: two modules are content coupled if one can make direct reference to the internals of the other (e.g. modifying code of, or referencing labels internal to, the other module). The result is that some or all of the content of one module are effectively included in the other. Content coupling can be thought of as using unadvertised module interfaces; this is in contrast to call coupling, which uses only advertised module interfaces.

A.3.2 Complexity of procedural software

Complexity is the measure of the decision points and logical paths of execution that code takes. Software engineering literature cites complexity as a negative characteristic of software because it impedes understanding of the logic and flow of the code. Another impediment to the understanding of code is the presence of code that is unnecessary, in that it is unused or redundant.

The use of layering to separate levels of abstraction and minimise circular dependencies further enables a better understanding of the TSF, providing more assurance that the TOE security functional requirements are accurately and completely instantiated in the implementation.

Reducing complexity also includes reducing or eliminating mutual dependencies, which pertains both to modules in a single layer and to those in separate layers. Modules that are mutually dependent may rely on one another to formulate a single result, which could result in a deadlock condition, or worse yet, a race condition (e.g., time of check vs. time of use concern), where the ultimate conclusion could be indeterminate and subject to the computing environment at the given instant in time.

Design complexity minimisation is a key characteristic of a reference validation mechanism, the purpose of which is to arrive at a TSF that is easily understood so that it can be completely analysed. (There are other important characteristics of a reference validation mechanism, such as TSF self-protection and non-bypassability; these other characteristics are covered by requirements in the ADV_ARC family.)

A.4 ADV_TDS: Subsystems and Modules

This clause provides additional guidance on the TDS family, and its use of the terms “subsystem” and “module”. This is followed by a discussion of how, as more-detailed becomes available, the requirement for the less-detailed is reduced.

A.4.1 Subsystems

Figure A.3 shows that, depending on the complexity of the TSF, the design may be described in terms of subsystems *and* modules (where subsystems are at a higher level of abstraction than modules); or it may just be described in terms of one level of abstraction (e.g., *subsystems* at lower assurance levels, *modules* at higher levels). In cases where a lower level of abstraction (modules) is presented, requirements levied on higher-level abstractions (subsystems) are essentially met by default. This concept is further elaborated in the discussion on subsystems and modules below.

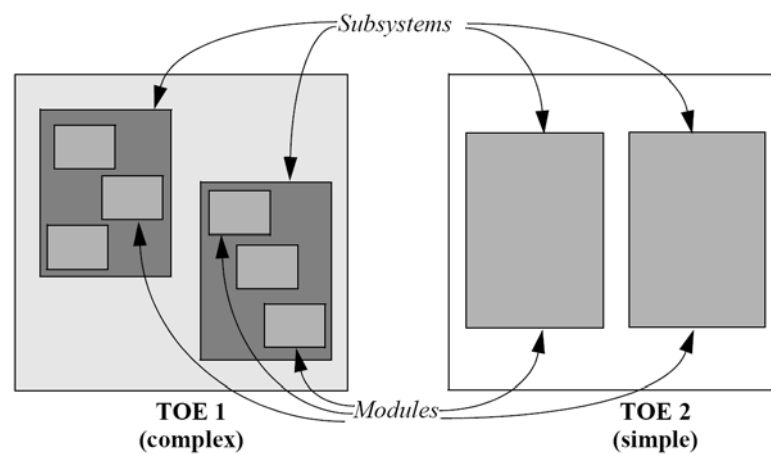


Figure A.3 — Subsystems and Modules

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

The first use of subsystems is to distinguish the TSF boundary; that is, the portions of the TOE that comprise the TSF. In general, a subsystem is part of the TSF if it has the capability (whether by design or implementation) to affect the correct operation of any of the SFRs. For example, for software that depends on different hardware execution modes to provide domain separation (see A.1) where SFR-enforcing code is executed in one domain, then all subsystems that execute in that domain would be considered part of the TSF. Likewise, if a server outside that domain implemented an SFR (e.g. enforced an access control policy over objects it managed), then it too would be considered part of the TSF.

The second use of subsystems is to provide a structure for describing the TSF at a level of description that, while describing how the TSF works, does not necessarily contain low-level implementation detail found in module descriptions (discussed later). subsystems are described at either a high level (lacking an abundance of implementation detail) or a detailed level (providing more insight into the implementation). The level of description provided for a subsystem is determined by the degree to which that subsystem is responsible for implementing an SFR.

6745 An *SFR-enforcing* subsystem is a subsystem that provides mechanisms for enforcing an element of any
 6746 SFR, or directly supports a subsystem that is responsible for enforcing an SFR. If a subsystem provides
 6747 (implements) an SFR-enforcing TSFI, then the subsystem is SFR-enforcing.

6748 Subsystems can also be identified as *SFR-supporting* and *SFR-non-interfering*. An SFR-supporting
 6749 subsystem is one that is depended on by an SFR-enforcing subsystem in order to implement an SFR,
 6750 but does not play as direct a role as an SFR-enforcing subsystem. An SFR-non-interfering subsystem is
 6751 one that is not depended upon, in either a supporting or enforcing role, to implement an SFR.

6752 **A.4.2 Modules**

6753 A module is generally a relatively small architectural unit that can be characterised in terms of the
 6754 properties discussed in TSF internals (ADV_INT). When both ADV_TDS.3 Basic modular design (or
 6755 above) requirements and TSF internals (ADV_INT) requirements are present in a PP or ST, a “module”
 6756 in terms of the TOE design (ADV_TDS) requirements refers to the same entity as a “module” for the
 6757 TSF internals (ADV_INT) requirements. Unlike subsystems, modules describe the implementation in a
 6758 level of detail that can serve as a guide to reviewing the implementation representation.

6759 It is important to note that, depending on the TOE, modules and subsystems may refer to the same
 6760 abstraction. For ADV_TDS.1 Basic design and ADV_TDS.2 Architectural design (which do not require
 6761 description at the module level) the subsystem description provides the lowest level detail available
 6762 about the TSF. For ADV_TDS.3 Basic modular design (which require module descriptions) these
 6763 descriptions provide the lowest level of detail, while the subsystem descriptions (if they exist as
 6764 separate entities) merely serve to put to the module descriptions in context. That is, it is not necessary
 6765 to provide detailed subsystem descriptions if module descriptions exist. In TOEs that are sufficiently
 6766 simple, a separate “subsystem description” is not necessary; the requirements can be met through
 6767 documentation provided by modules. For complex TOEs, the purpose of the subsystem description
 6768 (with respect to the TSF) is to provide the reader context so they can focus their analysis
 6769 appropriately. This difference is illustrated in Figure A.3.

6770 An SFR-enforcing module is a module that completely or partially implements a security functional
 6771 requirement (SFR) in the ST. Such modules may implement an SFR-enforcing TSFI, but some
 6772 functionality expressed in an SFR (for example, audit and object re-use functionality) may not be
 6773 directly tied to a single TSFI. As was the case with subsystems, SFR-supporting modules are those
 6774 modules that are depended upon by an SFR-enforcing module, but are not responsible for directly
 6775 implementing an SFR. SFR-non-interfering modules are those modules that do not deal, directly or
 6776 indirectly, with the enforcement of SFRs.

6777 It is important to note that the determination of what “directly implements” means is somewhat
 6778 subjective. In the narrowest sense of the term, it could be interpreted to mean the one or two lines of
 6779 code that actually perform a comparison, zeroing operation, etc. that implements a requirement. A
 6780 broader interpretation might be that it includes the module that is invoked in response to a SFR-
 6781 enforcing TSFI, and all modules that may be invoked in turn by that module (and so on until the
 6782 completion of the call). Neither of these interpretations is particularly satisfying, since the narrowness
 6783 of the first interpretation may lead to important modules being incorrectly categorised as SFR
 6784 supporting, while the second leads to modules that are actually not SFR-enforcing being classified as
 6785 such.

6786 A description of a module should be such that one could create an implementation of the module from
 6787 the description, and the resulting implementation would be 1) identical to the actual TSF
 6788 implementation in terms of the interfaces presented, 2) identical in the use of interfaces that are
 6789 mentioned in the design, and 3) functionally equivalent to the description of the purpose of the TSF
 6790 module. For instance, RFC 793 provides a high-level description of the TCP protocol. It is necessarily

6791 implementation independent. While it provides a wealth of detail, it is *not* a suitable design
 6792 description because it is not specific to an implementation. An actual implementation can add to the
 6793 protocol specified in the RFC, and implementation choices (for example, the use of global data vs. local
 6794 data in various parts of the implementation) may have an impact on the analysis that is performed.
 6795 The design description of the TCP module would list the interfaces presented by the implementation
 6796 (rather than just those defined in RFC 793), as well as an algorithm description of the processing
 6797 associated with the modules implementing TCP (assuming they were part of the TSF).

6798 In the design, modules are described in detail in terms of the function they provide (the purpose); the
 6799 interfaces they present (when required by the criteria); the return values from such interfaces; the
 6800 interfaces (presented by other modules) they use (provided those interfaces are required to be also
 6801 described); and a description of how they provide their functionality using a technique appropriate to
 6802 the method used to implement the module.

6803 The purpose of a module should be described indicating what function the module is providing. It
 6804 should be sufficient so that the reader could get a general idea of what the module's function is in the
 6805 architecture.

6806 The interfaces presented by a module are those interfaces used by other modules to invoke the
 6807 functionality provided. Interfaces include both *explicit* interfaces (e.g., a calling sequence invoked by
 6808 other modules) as well as *implicit* interfaces (e.g., global data manipulated by the module). Interfaces
 6809 are described in terms of how they are invoked, and any values that are returned. This description
 6810 would include a list of parameters, and descriptions of these parameters. If a parameter were expected
 6811 to take on a set of values (e.g., a "flag" parameter), the complete set of values the parameter could take
 6812 on that would have an effect on module processing would be specified. Likewise, parameters
 6813 representing data structures are described such that each field of the data structure is identified and
 6814 described. Global data should be described to the extent required to understand their purpose. The
 6815 level of description required for a global data structure needs to be identical to the one for module
 6816 interfaces, where the input parameter and return values correspond to the individual fields and their
 6817 possible values in the data structure. Global data structures may be described separate from the
 6818 modules that manipulate or read them as long as the design of the modules contain sufficient
 6819 information about the global data structures updated or the information extracted from global data
 6820 structures.

6821 Note that different programming languages may have additional "interfaces" that would be non-
 6822 obvious; an example would be operator/function overloading in C++. This "implicit interface" in the
 6823 class description would also be described as part of the module design. Note that although a module
 6824 could present only one interface, it is more common that a module presents a small set of related
 6825 interfaces.

6826 When it is required to describe the interfaces used by a module, it must be clear from either the design
 6827 description of the module or the purpose of the module called, what service is expected from the
 6828 module called. For example if Module A is being described, and it uses Module B's bubble sort routine,
 6829 the description of the interaction between modules must allow to identify why Module B's bubble sort
 6830 routine is called and what this call contributes to the implementation of the SFRs. The interface and
 6831 purpose of Module B's bubble sort routine must be described as part of the interfaces of Module B
 6832 (provided the level of ADV_TDS and the classification of Module B require a description its interfaces)
 6833 and so Module A just needs to identify what data it needs to have sorted using this routine. An
 6834 adequate description would be: "Module A invokes Module B's interface *double_bubble()* to sort the
 6835 usernames in alphabetical order".

6836 Note that if this sorting of the user names is not important for the enforcement of any SFR (e. g. it is
 6837 just done to speed up things and an algorithmically identical implementation of Module A could also

avoid to have the usernames sorted), the use of Module B's bubble sort routine is not SFR-enforcing and it is sufficient to explain in the description of Module A that the usernames are sorted in alphabetical order to enhance performance. Module B may be classified as "SFR-supporting" only and the level of ADV_TDS chosen indicates if the interfaces of SFR-supporting modules need to be described or if it is sufficient to just describe the purpose of Module B.

As discussed previously, the algorithmic description of the module should describe in an algorithmic fashion the implementation of the module. This can be done in pseudo-code, through flow charts, or (at ADV_TDS.3 Basic modular design) informal text. It discusses how the module inputs and called functions are used to accomplish the module's function. It notes changes to global data, system state, and return values produced by the module. It is at the level of detail that an implementation could be derived that would be very similar to the actual implementation of the TOE.

It should be noted that source code does not meet the module documentation requirements. Although the module design describes the implementation, it is *not* the implementation. The comments surrounding the source code might be sufficient documentation if they provide an explanation of the intent of the source code. In-line comments that merely state what each line of code is doing are useless because they provide no explanation of what the module is meant to accomplish.

In the elements below, the labels (SFR-enforcing, SFR-supporting, and SFR-non-interfering) discussed for subsystems and modules are used to describe the amount and type of information that needs to be made available by the developer. The elements have been structured so that there is no expectation that the developer provide *only* the information specified. That is, if the developer's documentation of the TSF provides the information in the requirements below, there is no expectation that the developer update their documentation and label subsystems and modules as SFR-enforcing, SFR-supporting or SFR-non-interfering. The primary purpose of this labelling is to allow developers with less mature development methodologies (and associated artifacts, such as detailed interface and design documentation) to provide the necessary evidence without undue cost.

6863 **A.4.3 Levelling Approach**

Because there is subjectivity in determining what is SFR-enforcing vs. SFR-supporting (and in some cases, even determining what is SFR-non-interfering the following paradigm has been adopted in this family. In early components of the family, the developer makes a determination about the classification of the subsystems into SFR-enforcing, etc., supplying the appropriate information, and there is little additional evidence for the evaluator to examine to support this claim. As the level of desired assurance increases, while the developer still makes a classification determination, the evaluator obtains more and more evidence that is used to confirm the developer's classification.

In order to focus the evaluator's analysis on the SFR-related portions of the TOE, especially at lower levels of assurance, the components of the family are levelled such that initially detailed information is required only for SFR-enforcing architectural entities. As the level of assurance increases, more information is required for SFR-supporting and (eventually) SFR-non-interfering entities. It should be noted that even when complete information is required, it is not required that all of this information be analysed in the same level of detail. The focus should be in all cases on whether the *necessary* information has been provided and analysed.

Table A.1 summarises the information required at each of the family components for the architectural entities to be described.

	TSF subsystem			TSF Module		
	SFR Enforce	SFR Support	SFR NI	SFR Enforce	SFR Support	SFR NI
ADV_TDS.1 Basic design (informal presentation)	structure, summary of SFR-Enf. behaviour, interactions	designation support ¹⁴	designation support			
ADV_TDS.2 Architectural design (informal presentation)	structure, detailed description of SFR-Enf. behaviour, summary of other behaviour, interactions	structure, summary of other behaviour, interactions	designation support, interactions			
ADV_TDS.3 Basic modular design (informal presentation)	description, interactions	description, interactions	description, interactions	purpose, SFR interfaces ¹⁵	interaction, purpose	interaction, purpose
ADV_TDS.4 Semiformal modular design (semiformal presentation)	description, interactions	description, interactions	description, interactions	purpose, SFR interfaces	purpose, SFR interfaces	interaction, purpose
ADV_TDS.5 Complete semiformal modular design (semiformal presentation)	description, interactions	description, interactions	description, interactions	purpose, all interfaces ¹⁶	purpose, all interfaces	purpose, all interfaces
ADV_TDS.6 Complete semiformal modular design with formal high-level design	description, interactions	description, interactions	description, interactions	purpose, all interfaces	purpose, all interfaces	purpose, all interfaces

¹⁴ *designation support* means that only documentation sufficient to support the classification of the subsystem / module is needed.

¹⁵ *SFR interfaces* means that the module description contains, for each SFR-related interface, the returned values and the called interfaces to other modules.

¹⁶ *All interfaces* means that the module description contains, for each interface, the returned values and the called interfaces to other modules.

	TSF subsystem			TSF Module		
	SFR Enforce	SFR Support	SFR NI	SFR Enforce	SFR Support	SFR NI
presentation (semiformal presentation; additional formal presentation)						

Table A.1 — Description Detail Levelling

A.4.4 Security relevance

The comments to WD2 regarding this chapter are pending as the contributor was not able to answer the comments until the deadline.

The ISO/IEC 15408 series concentrates the description, the evidence and the analysis on the security functionality of the TOE. This requires characterization of security relevance of functional and physical parts of the TOE. Interfaces, subsystems and modules may be categorised (either implicitly or explicitly) as “SFR-enforcing”, “SFR-supporting”, or “SFR-noninterfering”.

The developer evidence and the evaluation analysis relates to the TOE and focus on the TSF and its SFR-enforcing and SFR-supporting implementation. The security architecture description shall demonstrate that the identified non-TSF subsystems of the TOE are not bypassing the TSF and the TSF protects themselves against corruption by non-TSF code or entities. The developer shall describe the SFR-noninterfering interfaces, subsystems and modules in the TOE design and demonstrate that they do not interfere with the TSF because of their purposes, interactions or separation of resources.

An interface, subsystem or module is

- SFR-enforcing, if it directly implements an SFR.
- SFR-supporting if it has to operate functionally correctly in order to support the proper function of the SFRs.
- SFR-non-interfering if it is not related to the implementation of the SFRs.

The focus on security enforcing and security supporting functionality requires evidence of non-interference of the other functionality. Even correct implemented security enforcing functions and security mechanisms may be **bypassed, circumvented, deactivated, corrupted, or directly attacked**. Non-interference implies that the TSF cannot be misused and unauthorized access to the resources of the TSF implementation is prevented or impossible. Therefore the security architecture aspects of non-bypassability and self-protection are critical if security relevance of interfaces, subsystems and modules is categorized and this categorization is used in the vulnerability analysis.

TSF self-protection is the security architecture property whereby the TSF cannot be corrupted by non-TSF code or entities. This includes non-TSF subsystems of TOE and non-TOE parts of the IT product. It is similar to the evidence for SFR-non-interfering subsystems/modules.

The security domains are environments provided by the TSF for the use by untrusted entities in such a way that these environments are isolated and protected from each other.

Therefore the analysis of non-interference during evaluation requires examination of the security architecture of the TOE (ADV_ARC) and may need more information on non-TSF subsystems than only the TOE structure in terms of subsystems as provided for ADV_TDS.x.1. The developers shall provide a rationale that TSF is correctly defined and the analysis of SFR-non-interfering module in terms of its purpose and interaction with other modules

- **purpose:** how a module provides their functionality, no further design decisions are needed.

- **interaction:** reason that subsystems or modules communicate, and characterizes the information that is passed (less details than for interfaces).

During evaluation non-interference shall be analyzed as part of the examination of functional specification and TOE design, and the vulnerability analysis. The categorization of interfaces, subsystems and modules as SFR-enforcing, SFR-supporting and SFR-noninterfering implies specific examination of the functional specification, design and testing. An interpretation of TSFI as all accessible external interfaces of the TSF would help this analysis. The functional tests of all TSF subsystems (beginning with ATE_DPT.1) and all TSF modules (ATE_DPT.3 and higher) should provide evidence for the correctness of their security categorization.

A.5 Supplementary material on formal methods

Formal methods provide a mathematical representation of the TSF and its behaviour and are required by the ADV_FSP.6 Complete semi-formal functional specification with additional formal specification, ADV_SPM.1 Formal TOE security policy model, and ADV_TDS.6 Complete semiformal modular design with formal high-level design presentation components. There are two aspects of formal methods: the *specification language* that is used for formal expression, and the *theorem prover* that mathematically proves the completeness and correctness of the formal specification.

A formal specification is expressed within a formal system based upon well-established mathematical concepts. These mathematical concepts are used to define well-defined semantics, syntax and rules of inference. A formal system is an abstract system of identities and relations that can be described by specifying a formal alphabet, a formal language over that alphabet which is based on a formal syntax, and a set of formal rules of inference for constructing derivations of sentences in the formal language.

The evaluator should examine the identified formal systems to make sure that:

- The semantics, syntax and inference rules of the formal system are defined or a definition is referenced.

- Each formal system is accompanied by explanatory text that provides defined **semantics** so that:

- 1) the explanatory text provides defined meanings of terms, abbreviations and acronyms that are used in a context other than that accepted by normal usage;

- 2) the use of a formal system and semiformal notation use is accompanied by supporting explanatory text in informal style appropriate for unambiguous meaning;

- 3) the formal system is able to express rules and characteristics of applicable SFPs, security functionality and interfaces (providing details of effects, exceptions and error messages) of TSF, their subsystems or modules to be specified for the assurance family for which the notations are used;

- 4) the notation provides rules to determine the meaning of syntactical valid constructs.

- 6951 • Each formal system uses a formal syntax that provides rules to unambiguously recognise
6952 constructs.
- 6953 • Each formal system provides proof rules which
- 6954 5) support logical reasoning of well-established mathematical concepts,
- 6955 6) help to prevent derivation of contradictions.
- 6956 If the developer uses a formal system which is already accepted by the evaluation authority the
6957 evaluator can rely on the level of formality and strength of the system and focus on the instantiation of
6958 the formal system to the TOE specifications and correspondence proofs.
- 6959 The formal style supports mathematical proofs of the security properties based on the security
6960 features, the consistency of refinements and the correspondence of the representations. Formal tool
6961 support seems adequate whenever manual derivations would otherwise become long winded and
6962 incomprehensible. Formal tools are also apt to reduce the error probability inherent in manual
6963 derivations.
- 6964 Examples of formal systems:
- 6965 • The **Z specification language** is highly expressive, and supports many different methods or styles
6966 of formal specification. The use of Z has been predominantly for model-oriented specification,
6967 using *schemas* to formally specify operations. See https://en.wikipedia.org/wiki/Z_notation for
6968 more information.
- 6969 • **ACL2** is an open-source formal system comprising a LISP-based specification language and a
6970 theorem prover. See <http://www.cs.utexas.edu/users/moore/acl2/> for further information.
- 6971 • **Isabelle** is a popular generic theorem proving environment that allows mathematical formulae to
6972 be expressed in a formal language and provides tools for proving those formulae within a logical
6973 calculus (see e.g. <http://www.cl.cam.ac.uk/Research/HVG/Isabelle/> for additional information).
- 6974 • The **B method** is a formal system based on the propositional calculus, the first order predicate
6975 calculus with inference rules and set theory (see e.g. <https://en.wikipedia.org/wiki/B-Method> for
6976 further information).
- 6977 • **NuSMV** (based on its predecessor SMV) is a symbolic model checker designed to be an open
6978 architecture for model checking which can be reliably used for the verification of industrial
6979 designs, as a core for custom verification tools, and as a testbed for formal verification techniques.
6980 See <http://nusmv.fbk.eu/> for more information.
- 6981 • **Coq** is a formal proof management system that provides a formal language to write mathematical
6982 definitions, executable algorithms and theorems together with an environment for semi-
6983 interactive development of machine-checked proofs. See <https://coq.inria.fr/> for more
6984 information.
- 6985 • **SystemVerilog** is a combined hardware description language and hardware verification language
6986 based on Verilog.

Annex B (informative)

Composition (ACO)

6991 The goal of this annex is to explain the concepts behind composition evaluations and the ACO criteria.
6992 This annex does not define the ASE criteria; this definition can be found in clause 9.

6993 **B.1 Necessity for composed TOE evaluations**

6994 The IT market is, on the whole, made up of vendors offering a particular type of product/technology.
6995 Although there is some overlap, where a PC hardware vendor may also offer application software
6996 and/or operating systems or a chip manufacturer may also develop a dedicated operating system for
6997 their own chipset, it is often the case that an IT solution is implemented by a variety of vendors.

6998 There is sometimes a need for assurance in the combination (composition) of components in addition
6999 to the assurance of the individual components. Although there is cooperation between these vendors,
7000 in the dissemination of certain material required for the technical integration of the components, the
7001 agreements rarely stretch to the extent of providing detailed design information and development
7002 process/procedure evidence. This lack of information from the developer of a component on which
7003 another component relies means that the dependent component developer does not have access to the
7004 type of information necessary to perform an evaluation of both the dependent and base components at
7005 EAL2 or above. Therefore, while an evaluation of the dependent component can still be performed at
7006 any assurance level, to compose components with assurance at EAL2 or above it is necessary to reuse
7007 the evaluation evidence and results of evaluations performed for the component developer.

7008 It is intended that the ACO criteria are applicable in the situation where one IT entity is dependent on
7009 another for the provision of security services. The entity providing the services is termed the “base
7010 component”, and that receiving the services is termed the “dependent component”. This relationship
7011 may exist in a number of contexts. For example, an application (dependent component) may use
7012 services provided by an operating system (base component). Alternatively, the relationship may be
7013 peer-to-peer, in the sense of two linked applications, either running in a common operating system
7014 environment, or on separate hardware platforms. If there is a dominant peer providing the services to
7015 the minor peer, the dominant peer is considered to be the base component and the minor peer the
7016 dependent component. If the peers provide services to each other in a mutual manner, each peer will
7017 be considered to be the base component for the services offered and dependent component for the
7018 services required. This will require iterations of the ACO components applying all requirements to
7019 each type of component peer.

7020 The criteria are also intended to be more broadly applicable, stepwise (where a composed TOE
7021 comprised of a dependent component and a base component itself becomes the base component of
7022 another composed TOE), in more complex relationships, but this may require further interpretation.

7023 It is still required for composed TOE evaluations that the individual components are evaluated
7024 independently, as the composition evaluation builds on the results of the individual component
7025 evaluations. The evaluation of the dependent component may still be in progress when the composed
7026 TOE evaluation commences. However, the dependent component evaluation must complete before the
7027 composed TOE evaluation completes.

7028 The composed evaluation activities may take place at the same time as the dependent component
7029 evaluation. This is due to two factors:

7030 a) Economic/business drivers - the dependent component developer will either be sponsoring the
7031 composition evaluation activities or supporting these activities as the evaluation deliverables from
7032 the dependent component evaluation are required for composed evaluation activities.

7033 b) Technical drivers - the components consider whether the requisite assurance is provided by the
7034 base component (e.g. considering the changes to the base component since completion of the
7035 component evaluation) with the understanding that the dependent component has recently
7036 undergone (is undergoing) component evaluation and all evaluation deliverables associated with
7037 the evaluation are available. Therefore, there are no activities during composition requesting the
7038 dependent component evaluation activities to be re-verified. Also, it is verified that the base
7039 component forms (one of) the test configurations for the testing of the dependent component
7040 during the dependent component evaluation, leaving ACO_CTT to consider the base component in
7041 this configuration.

7042 The evaluation evidence from the evaluation of the dependent component is required input into the
7043 composed TOE evaluation activities. The only evaluation material from the evaluation of the base
7044 component that is required as input into the composed TOE evaluation activities:

7045 a) Residual vulnerabilities in the base component, as reported during the base component
7046 evaluation. This is required for the ACO_VUL activities.

7047 No other evaluation evidence from the base component activities should be required for the composed
7048 TOE evaluation, as the evaluation results from the component evaluation of the base component
7049 should be reused. Additional information about the base component may be required if the composed
7050 TOE TSF includes more of the base component than was considered to be TSF during component
7051 evaluation of the base component.

7052 The component evaluation of the base and dependent components are assumed to be complete by the
7053 time final verdicts are assigned for the ACO components.

7054 The ACO_VUL components only consider resistance against an attacker with an attack potential up to
7055 Enhanced-Basic. This is due to the level of design information that can be provided of how the base
7056 component provides the services on which the dependent component relies through application of the
7057 ACO_DEV activities. Therefore, the confidence arising from composed TOE evaluations using CAPs is
7058 limited to a level similar to that obtained from EAL4 component TOE evaluations. Although assurance
7059 in the components that comprise the composed TOE may be higher than EAL4.

7060 **B.2 Performing Security Target evaluation for a composed TOE**

7061 An ST will be submitted by the developer for the evaluation of the composed (base component +
7062 dependent component) TOE. This ST will identify the assurance package to be applied to the
7063 composed TOE, providing assurance in the composed entity by drawing upon the assurance gained in
7064 the component evaluations.

7065 The purpose of considering the composition of components within an ST is to validate the
7066 compatibility of the components from the point of view of both the environment and the requirements,
7067 and also to assess that the composed TOE ST is consistent with the component STs and the security
7068 policies expressed within them. This includes determining that the component STs and the security
7069 policies expressed within them are compatible.

7070 The composed TOE ST may refer out to the content of the component STs, or the ST author may chose
 7071 to reiterate the material of the component STs within the composed TOE ST providing a rationale of
 7072 how the component STs are represented in the composed TOE ST.

7073 During the conduct of the ASE_CCL evaluation activities for a composed TOE ST the evaluator
 7074 determines that the component STs are accurately represented in the composed TOE ST. This is
 7075 achieved through determining that the composed TOE ST demonstrably conforms to the component
 7076 TOE STs. Also, the evaluator will need to determine that the dependencies of the dependent
 7077 component on the operational environment are adequately fulfilled in the composed TOE.

7078 The composed TOE description will describe the composed solution. The logical and physical scope
 7079 and boundary of the composed solution will be described, and the logical boundary(ies) between the
 7080 components will also be identified. The description will identify the security functionality to be
 7081 provided by each component.

7082 The statement of SFRs for the composed TOE will identify which component is to satisfy an SFR. If an
 7083 SFR is met by both components, then the statement will identify which component meets the different
 7084 aspects of the SFR. Similarly the composed TOE Summary Specification will identify which component
 7085 provides the security functionality described.

7086 The package of ASE: Security Target evaluation requirements applied to the composed TOE ST should
 7087 be consistent with the package of ASE: Security Target evaluation requirements used in the
 7088 component evaluations.

7089 Reuse of evaluation results from the evaluation of component STs can be made in the instances that
 7090 the composed TOE ST directly refers to the component STs. e.g. if the composed TOE ST refers to a
 7091 component ST for part of its statement of SFRs, the evaluator can understand that the requirement for
 7092 the completion of all assignment and selection operations (as stated in ASE_REQ.*.3C has been
 7093 satisfied in the component evaluations.

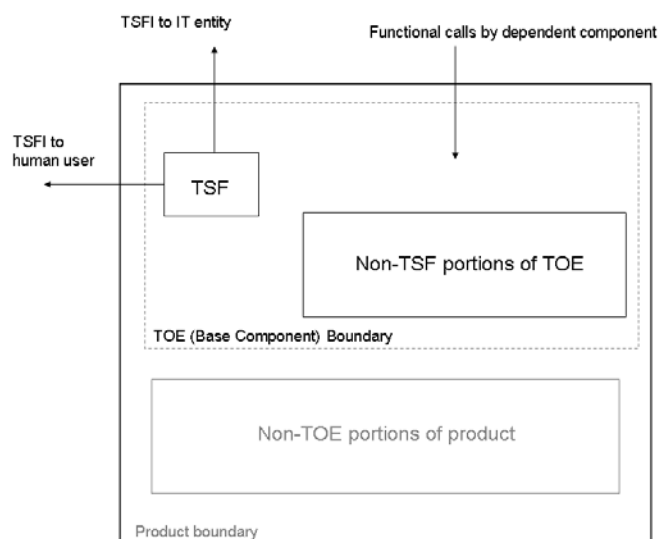
7094 **B.3 Interactions between composed IT entities**

7095 The TSF of the base component is often defined without knowledge of the dependencies of the
 7096 possible applications with which it may be composed. The TSF of this base component is defined to
 7097 include all parts of the base component that have to be relied upon for enforcement of the base
 7098 component SFRs. This will include all parts of the base component required to implement the base
 7099 component SFRs.

7100 The TSFI of this base component represents the interfaces provided by the TSF to the external entities
 7101 defined in the statement of SFRs to invoke a service of the TSF. This includes interfaces to the human
 7102 user and also interfaces to external IT entities. However, the TSFI only includes those interfaces to the
 7103 TSF, and therefore is not necessarily an exhaustive interface specification of all possible interfaces
 7104 available between an external entity and the base component. The base component may present
 7105 interfaces to services that were not considered security-relevant, either because of the inherent
 7106 purpose of the service (e.g., adjust type font) or because associated ISO/IEC 15408-2 SFRs are not
 7107 being claimed in the base component's ST (e.g. the login interface when no FIA: Identification and
 7108 authentication SFRs are claimed).

7109 The functional interfaces provided by the base component are in addition to the security interfaces
 7110 (TSFIs), and are not required to be considered during the base component evaluation. These often
 7111 include interfaces that are used by a dependent component to invoke a service provided by the base
 7112 component.

7113 The base component may include some indirect interfaces through which TSFIs may be called, e.g.
 7114 APIs that can be used to invoke a service of the TSF, which were not considered during the evaluation
 7115 of the base component.



7116

7117 **Figure B.1 — Base component abstraction**

7118 The dependent component, which relies on the base component, is similarly defined: interfaces to
 7119 external entities defined in the SFRs of the component ST are categorised as TSFI and are examined in
 7120 ADV_FSP. This is illustrated in Figure B.1).

7121 Any call out from the dependent TSF to the environment in support of an SFR will indicate that the
 7122 dependent TSF requires some service from the environment in order to satisfy the enforcement of the
 7123 stated dependent component SFRs. Such a service is outside the dependent component boundary and
 7124 the base component is unlikely to be defined in the dependent ST as an external entity. Hence, the calls
 7125 for services made out by the dependent TSF to its underlying platform (the base component) will not
 7126 be analysed as part of the Functional specification (ADV_FSP) activities. These dependencies on the
 7127 base component are expressed in the dependent component ST as security objectives for the
 7128 environment.

7129 This abstraction of the dependent component and the interfaces is shown in Figure B.2 below.

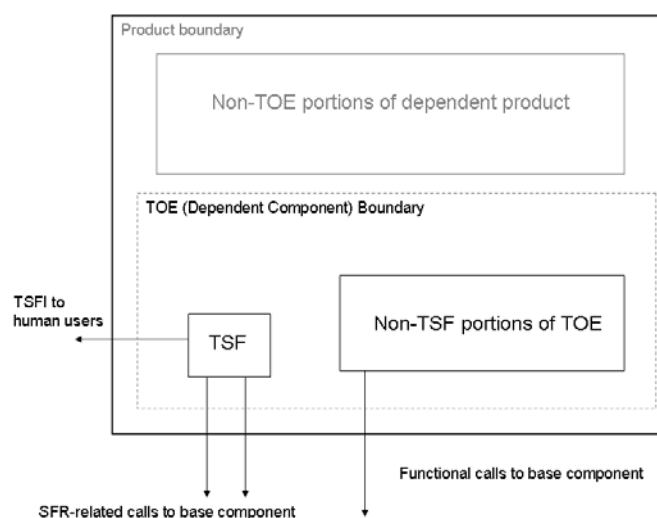


Figure B.2 — Dependent component abstraction

When considering the composition of the base component and the dependent component, if the dependent component's TSF requires services from the base component to support the implementation of the SFR, the interface to the service will need to be defined. If that service is provided by the base component's TSF, then that interface should be a TSFI of the base component and will therefore already be defined within the functional specification of the base component.

If, however, the service called by the dependent component's TSF is not provided by the TSF of the base component (i.e., it is implemented in the non-TSF portion of the base component or possibly even in the non-TOE portion of the base component (not illustrated in Figure B.3), there is unlikely to be a TSFI of the base component relating to the service, unless the service is mediated by the TSF of the base component. The interfaces to these services from the dependent component to the operational environment are considered in the family Reliance of dependent component (ACO_REL).

The non-TSF portion of the base component is drawn into the TSF of the composed TOE due to the dependencies the dependent component has on the base component to support the SFRs of the dependent component. Therefore, in such cases, the TSF of the composed TOE would be larger than simply the sum of the components' TSFs.

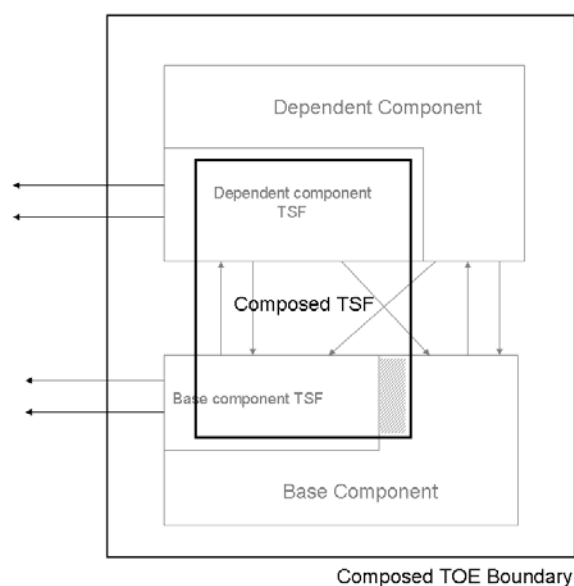


Figure B.3 — Composed TOE abstraction

It may be the case that the base component TSFI is being called in a manner that was unforeseen in the base component evaluation. Hence there would be a requirement for further testing of the base component TSFI.

The possible interfaces are further described in the following diagram (Figure B.4) and supporting text.

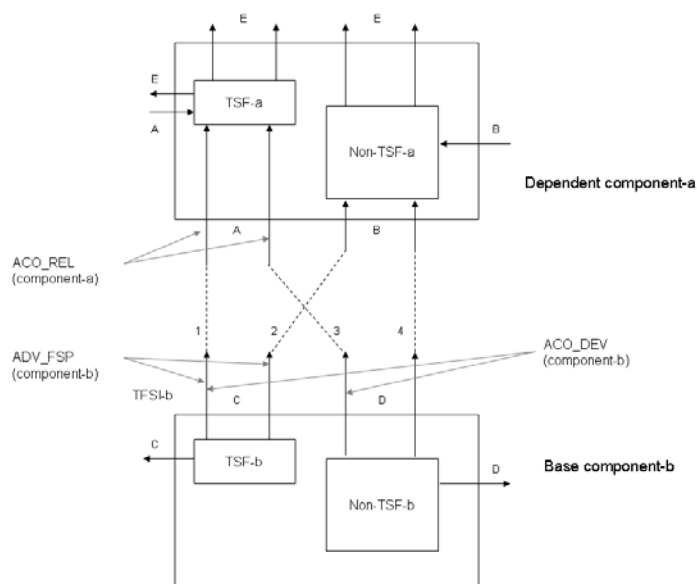


Figure B.4 — Composed component interfaces

- a) Arrows going *into* 'dependent component-a' (A and B) = where the component expects the environment to respond to a service request (responding to calls out from dependent component to the environment);

- 7159 b) Arrows coming *out* of 'base component-b' (C and D) = interfaces of services provided by the base
7160 component to the environment;
- 7161 c) Broken lines between components = types of communication between pairs of interfaces;
- 7162 d) The other (grey) arrows = interfaces that are described by the given criteria.
- 7163 The following is a simplification, but explains the considerations that need to be made.
- 7164 There are components a ('dependent component-a') and b ('base component-b'): the arrows coming
7165 *out* of TSF-a are services provided by TSF-a and are therefore TSFIs(a); likewise, the arrows coming
7166 *out* of TSF-b ("C") are TSFIs(b). These are each detailed in their respective functional specs.
7167 component-a is such that it requires services from its environment: those needed by the TSF(a) are
7168 labelled "A"; the other (not related to TSF-a) services are labelled "B".
- 7169 When component-a and component-b are combined, there are four possible combinations of {services
7170 needed by component-a} and {services provided by component-b}, shown as broken lines (types of
7171 communication between pairs of interfaces). Any set of these might exist for a particular composition:
- 7172 a) TSF-a needs those services that are provided by TSF-b ("A" is connected to "C"): this is
7173 straightforward: the details about "C" are in the FSP for component-b. In this instance the
7174 interfaces should all be defined in the functional specifications for the component-b.
- 7175 b) Non-TSF-a needs those services that are provided by TSF-b ("B" is connected to "C"): this is
7176 straightforward (again, the details about "C" are in the FSP for component-b), but unimportant:
7177 security-wise.
- 7178 c) Non-TSF-a needs those services that are provided by non-TSF-b ("B" is connected to "D"): we have
7179 no details about D, but there are no security implications about the use of these interfaces, so they
7180 do not need to be considered in the evaluation, although they are likely to be an integration issue
7181 for the developer.
- 7182 d) TSF-a needs those services that are provided by non-TSF-b ("A" is connected to "D"): this would
7183 arise when component-a and component-b have different senses of what a "security service" is.
7184 Perhaps component-b is making no claims about I&A (has no FIA SFRs in its ST), but component-a
7185 needs authentication provided by its environment. There are no details about the "D" interfaces
7186 available (they are not TSFI (b), so they are not in component-b's FSP).
- 7187 Note: if the kind of interaction described in case d above exists, then the TSF of the composed TOE
7188 would be TSF-a + TSF-b + Non-TSF-b. Otherwise, the TSF of the composed TOE would be TSF-a + TSF-
7189 b.
- 7190 Interfaces types 2 and 4 of Figure B.4 are not directly relevant to the evaluation of the composed TOE.
7191 Interfaces 1 and 3 will be considered during the application of different families:
- 7192 a) Functional specification (ADV_FSP) (for component-b) will describe the C interfaces.
- 7193 b) Reliance of dependent component (ACO_REL) will describe the A interfaces.
- 7194 c) Development evidence (ACO_DEV) will describe the C interfaces for connection type 1 and the D
7195 interfaces for connection type 3.
- 7196 A typical example where composition may be applied is a database management system (DBMS) that
7197 relies upon its underlying operating system (OS). During the evaluation of the DBMS component, there

7198 will be an assessment made of the security properties of that DBMS (to whatever degree of rigour is
 7199 dictated by the assurance components used in the evaluation): its TSF boundary will be identified, its
 7200 functional specification will be assessed to determine whether it describes the interfaces to the
 7201 security services provided by the TSF, perhaps additional information about the TSF (its design,
 7202 architecture, internal structure) will be provided, the TSF will be tested, aspects of its life-cycle and its
 7203 guidance documentation will be assessed, etc.

7204 However, the DBMS evaluation will not call for any evidence concerning the dependency the DBMS has
 7205 on the OS. The ST of the DBMS will most likely state assumptions about the OS in its Assumptions
 7206 subclause and state security objectives for the OS in its Environment subclause. The DBMS ST may
 7207 even instantiate those objectives for the environment in terms of SFRs for the OS. However, there will
 7208 be no specification for the OS that mirrors the detail in the functional specification, architecture
 7209 description, or other ADV evidence as for the DBMS. Reliance of dependent component (ACO_REL) will
 7210 fulfil that need.

7211 Reliance of dependent component (ACO_REL) describes the interfaces of the dependent TOE that make
 7212 the calls to the base component for the provision of services. These are the interfaces to which the
 7213 base component is to respond. The interface descriptions are provided from the dependent
 7214 component's viewpoint.

7215 Development evidence (ACO_DEV) describes the interfaces provided by the base component, which
 7216 respond to the dependent component service requests. These interfaces are mapped to the relevant
 7217 dependent component interfaces that are identified in the reliance information. (The completeness of
 7218 this mapping, whether the base component interfaces described represent all dependent component
 7219 interfaces, is not verified here, but in Composition rationale (ACO_COR)). At the higher levels of
 7220 ACO_DEV the subsystems providing the interfaces are described.

7221 Any interfaces required by the dependent component that have not been described for the base
 7222 component are reported in the rationale for Composition rationale (ACO_COR). The rationale also
 7223 reports whether the interfaces of the base component on which the dependent component relies were
 7224 considered within the base component evaluation. For any interfaces that were not considered in the
 7225 base component evaluation, a rationale is provided of the impact of using the interface on the base
 7226 component TSF.

Annex C (informative)

Cross reference of assurance component dependencies

The dependencies documented in the components of Clauses 7 and 9-15 are the direct dependencies between the assurance components.

The following dependency tables for assurance components show their direct, indirect and optional dependencies. Each of the components that is a dependency of some assurance component is allocated a column. Each assurance component is allocated a row. The value in the table cell indicate whether the column label component is directly required (indicated by a cross "X"), indirectly required (indicated by a dash "-") or optional (indicated by an "O") , by the row label component. If no character is presented, the component is not dependent upon another component.

ADV	ADV_FSP.1	ADV_FSP.2	ADV_FSP.3	ADV_FSP.4	ADV_FSP.5	ADV_FSP.6	ADV_IMP.1	ADV_TDS.1	ADV_TDS.3	ALC_CMC.5	ALC_CMS.1	ALC_DVS.2	ALC_LCD.1	ALC_TAT.1
ADV_ARC.1	X	-						X						
ADV_COMP.1														
ADV_FSP.1														
ADV_FSP.2		-						X						
ADV_FSP.3		-						X						
ADV_FSP.4		-						X						
ADV_FSP.5		-		-			X	X	-					-
ADV_FSP.6		-		-			X	X	-					-
ADV_IMP.1		-		-			-	-	X					X
ADV_IMP.2		-		-			-	-	X	X	-	-	-	X
ADV_INT.1		-		-			X	-	X					X
ADV_INT.2		-		-			X	-	X					X
ADV_INT.3		-		-			X	-	X					X
ADV_SPM.1		-		-	X	X	-	-	-					-

ADV_TDS.1		X						-						
ADV_TDS.2		-	X					-						
ADV_TDS.3		-		X				-						
ADV_TDS.4		-		-	X		-	-	-					-
ADV_TDS.5		-		-	X		-	-	-					-
ADV_TDS.6		-		-		X	-	-	-					-

Table C.1 — Dependency table for Class ADV: Development

AGD	ADV_FSP.1
AGD_OPE.1	X
AGD_PRE.1	

Table C.2 — Dependency table for Class AGD: Guidance documents

ALC	ADV_FSP.2	ADV_FSP.4	ADV_IMP.1	ADV_TDS.1	ADV_TDS.3	ALC_CMS.1	ALC_CMS.3	ALC_DVS.1	ALC_DVS.2	ALC_LCD.1	ALC_TAT.1
ALC_CMC.1						X					
ALC_CMC.2						X					
ALC_CMC.3						X		X		X	
ALC_CMC.4						X		X		X	
ALC_CMC.5						X			X	X	
ALC_CMS.1											
ALC_CMS.2											
ALC_CMS.3											

ALC_CMS.4											
ALC_CMS.5											
ALC_COMP.1											
ALC_DEL.1											
ALC_DVS.1											
ALC_DVS.2											
ALC_FLR.1											
ALC_FLR.2											
ALC_FLR.3											
ALC_LCD.1											
ALC_LCD.2											
ALC_TAT.1	-	-	X	-	-						-
ALC_TAT.2	-	-	X	-	-						-
ALC_TAT.3	-	-	X	-	-						-
ALC_TDA.1											
ALC_TDA.2							X				
ALC_TDA.3	-	-	X	-	-		X				X

Table C.3 — Dependency table for Class ALC: Life-cycle support

APE	APE_ECD.1	APE_INT.1	APE_OBJ.2	APE_REQ.1	APE_SPD.1
APE_CCL.1	X	X		X	
APE_ECD.1					
APE_INT.1					
APE_OBJ.1					
APE_OBJ.2					X

APE_REQ.1	X				
APE_REQ.2	X		X		-
APE_SPD.1					

Table C.4 — Dependency table for Class APE: Protection Profile evaluation

ACE	ACE_CCL.1	ACE_ECD.1	ACE_INT.1	ACE_MCO.1	ACE_OBJ.1	ACE_OBJ.2	ACE_REQ.1	ACE_REQ.2	ACE_SPD.1	APE_ECD.1
ACE_CCL.1		X	X		-		0	0	-	-
ACE_CCO.1	X	X	X	X	0	0	0	0	X	-
ACE_ECD.1										
ACE_INT.1										
ACE_MCO.1		-	X		0	0	0	0	X	-
ACE_OBJ.1										
ACE_OBJ.2									X	
ACE_REQ.1									X	X
ACE_REQ.2		X				X				
ACE_SPD.1										

Table C.5 — Dependency table for Class ACE: Protection Profile Configuration evaluation

ASE	ADV_ARC.1	ADV_FSP.1	ADV_FSP.2	ADV_TDS.1	ASE_ECD.1	ASE_INT.1	ASE_OBJ.2	ASE_REQ.1	ASE_SPD.1
ASE_CCL.1					X	X		X	
ASE_COMP.1									
ASE_ECD.1									
ASE_INT.1									

ASE_OBJ.1									
ASE_OBJ.2									X
ASE_REQ.1					X				
ASE_REQ.2					X		X		-
ASE_SPD.1									
ASE_TSS.1		X			-	X		X	
ASE_TSS.2	X	-	-	-	-	X		X	

Table C.6 — Dependency table for Class ASE: Security Target evaluation

ATE	ADV_ARC.1	ADV_FSP.1	ADV_FSP.2	ADV_FSP.3	ADV_FSP.4	ADV_FSP.5	ADV_IMP.1	ADV_TDS.1	ADV_TDS.2	ADV_TDS.3	ADV_TDS.4	AGD_OPE.1	AGD_PRE.1	ALC_TAT.1	ATE_COV.1	ATE_FUN.1
ATE_COMP.1																
ATE_COV.1			X					-							-	X
ATE_COV.2			X					-							-	X
ATE_COV.3			X					-							-	X
ATE_DPT.1	X	-	-	-				-	X						-	X
ATE_DPT.2	X	-	-		-			-		X					-	X
ATE_DPT.3	X	-	-		-	-	-	-		-	X			-	-	X
ATE_DPT.4	X	-	-		-	-	X	-		-	X			-	-	X
ATE_FUN.1			-					-							X	-
ATE_FUN.2			-					-							X	-
ATE_IND.1		X										X	X			
ATE_IND.2		-	X					-				X	X		X	X
ATE_IND.3		-	-		X			-				X	X		X	X

Table C.7 Dependency table for Class ATE: Tests

AVA	ADV_ARC.1	ADV_FSP.1	ADV_FSP.2	ADV_FSP.3	ADV_FSP.4	ADV_IMP.1	ADV_TDS.1	ADV_TDS.2	ADV_TDS.3	AGD_OPE.1	AGD_PRE.1	ALC_TAT.1	ATE_COV.1	ATE_DPT.1	ATE_FUN.1
AVA_COMP.1															
AVA_VAN.1		X								X	X				
AVA_VAN.2	X	-	X				X			X	X				
AVA_VAN.3	X	-	-	-	X	X	-	-	X	X	X	-	-	X	-
AVA_VAN.4	X	-	-	-	X	X	-	-	X	X	X	-	-	X	-
AVA_VAN.5	X	-	-	-	X	X	-	-	X	X	X	-	-	X	-

Table C.8 Dependency table for Class AVA: Vulnerability assessment

ACO	ACO_DEV.1	ACO_DEV.2	ACO_DEV.3	ACO_REL.1	ACO_REL.2	ALC_CM.C.1	ALC_CMS.1
ACO_COR.1	X			X		X	-
ACO_CTT.1	X			X			
ACO_CTT.2		X		-	X		
ACO_DEV.1				X			
ACO_DEV.2				X			
ACO_DEV.3					X		
ACO_REL.1							
ACO_REL.2							
ACO_VUL.1	X			-			
ACO_VUL.2		X		-			
ACO_VUL.3			X		-		

Table C.9 Dependency table for class ACO: Composition

7258



Form 8A: Committee decision for DIS

Secretariat:	ISO/IEC JTC 1/SC 27
DIN	N 20035
Project number and title: ISO/IEC CD 15408-3 - Information technology — Security techniques — Evaluation criteria for IT security — Part 3: Security assurance components	

This form should be sent to the ISO Central Secretariat (<http://isotc.iso.org/livelink/si/>), together with the draft of the project, by the secretariat of the technical committee or subcommittee concerned.

The accompanying document is submitted for circulation to member body vote: <input checked="" type="checkbox"/> As a DIS
Consensus has been obtained from the P-members of the committee: on 2019-10-18 <input checked="" type="checkbox"/> At the meeting of ISO/IEC JTC 1/SC 27. See Resolution number 16. In document N 20053. <input type="checkbox"/> By ballot initiated on Please attach a copy of the ballot results (if applicable)

Listing of the P-members (NWIP, CD or Resolution)	
P-members in favour:	14
Belgium (NBN), China (SAC), France (AFNOR), Iran, Islamic Republic of (ISIRI), Ireland (NSAI), Japan (JISC), Korea, Republic of (KATS), Lebanon (LIBNOR), Mexico (DGN), Panama (COPANIT), Peru (INACAL), South Africa (SABS), Sri Lanka (SLSI), United Kingdom (BSI)	
P-members voting against:	2
Germany (DIN), United States (ANSI)	

<p>P-members abstaining: 31</p> <p>Algeria (IANOR), Argentina (IRAM), Australia (SA), Austria (ASI), Brazil (ABNT), Canada (SCC), Costa Rica (INTECO), Cyprus (CYS), Denmark (DS), Finland (SFS), India (BIS), Indonesia (BSN), Israel (SII), Italy (UNI), Luxembourg (ILNAS), Malaysia (DSM), Mauritius (MSB), Netherlands (NEN), New Zealand (NZSO), Norway (SN), Poland (PKN), Romania (ASRO), Russian Federation (GOST R), Singapore (SSC), Slovakia (UNMS SR), Spain (UNE), Sweden (SIS), Switzerland (SNV), Ukraine (DSTU), United Arab Emirates (ESMA), Uruguay (UNIT)</p>
<p>P-members who did not vote: 1</p> <p>Saint Kitts and Nevis (SKNBS)</p>
<p>Remarks:</p> <p>As per Paris CRM Recommendation 16 of the Comment Resolution Meeting (contained in SC 27 N20053) the text for a 1st DIS of ISO/IEC 15408-3 as presented in SC 27 N20035 was submitted to the ISO Central Secretariat ITTF) for the 1st DIS ballot processing on 2020-03-17.</p>

I hereby confirm that this draft meets the requirements of Part 2 of the ISO/IEC Directives:		
<p>Secretariat:</p> <p>DIN</p>	<p>Date:</p> <p>2020-03-17</p>	<p>Name/Signature of TC/SC Secretary:</p> <p>Passia, Krystyna Mrs</p>

Result of voting

Ballot Information

Ballot reference	ISO/IEC CD 15408-3.3 - ISO-IECJTC1-SC27_N19507
Ballot type	CD
Ballot title	Information technology -- Security techniques -- Evaluation criteria for IT security -- Part 3: Security assurance components
Opening date	2019-07-12
Closing date	2019-09-06
Note	3rd CD Consideration In accordance with Recommendation 14 (see SC 27 N19523) of the 58th SC 27/WG 3 meeting / CRM held in Tel Aviv, Israel, 2019-04-01/05 the hereby attached document is circulated for a 8-week 3rd CD letter ballot closing by 2019-09-06.

Member responses:

Votes cast (48)	Algeria (IANOR) Argentina (IRAM) Australia (SA) Austria (ASI) Belgium (NBN) Brazil (ABNT) Canada (SCC) China (SAC) Costa Rica (INTECO) Côte d'Ivoire (CODINORM) Cyprus (CYS) Denmark (DS) Finland (SFS) France (AFNOR) Germany (DIN) India (BIS) Indonesia (BSN) Iran, Islamic Republic of (ISIRI) Ireland (NSAI) Israel (SII) Italy (UNI) Japan (JISC) Korea, Republic of (KATS) Lebanon (LIBNOR) Luxembourg (ILNAS) Malaysia (DSM) Mauritius (MSB) Mexico (DGN)
------------------------	----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------

Netherlands (NEN)
 New Zealand (NZSO)
 Norway (SN)
 Panama (COPANIT)
 Peru (INACAL)
 Poland (PKN)
 Romania (ASRO)
 Russian Federation (GOST R)
 Singapore (SSC)
 Slovakia (UNMS SR)
 South Africa (SABS)
 Spain (UNE)
 Sri Lanka (SLSI)
 Sweden (SIS)
 Switzerland (SNV)
 Ukraine (DSTU)
 United Arab Emirates (ESMA)
 United Kingdom (BSI)
 United States (ANSI)
 Uruguay (UNIT)

Comments submitted (0)

Votes not cast (1)

Saint Kitts and Nevis (SKNBS)

Questions:

Q.1 "Do you approve the circulation of the draft as a DIS?"

Votes by members	Q.1
Algeria (IANOR)	Abstention
Argentina (IRAM)	Abstention
Australia (SA)	Abstention
Austria (ASI)	Abstention
Belgium (NBN)	Approval
Brazil (ABNT)	Abstention
Canada (SCC)	Abstention
China (SAC)	Approval with comments
Costa Rica (INTECO)	Abstention
Côte d'Ivoire (CODINORM)	Abstention
Cyprus (CYS)	Abstention
Denmark (DS)	Abstention
Finland (SFS)	Abstention
France (AFNOR)	Approval with comments

Germany (DIN)	Disapproval
India (BIS)	Abstention
Indonesia (BSN)	Abstention
Iran, Islamic Republic of (ISIRI)	Approval
Ireland (NSAI)	Approval
Israel (SII)	Abstention
Italy (UNI)	Abstention
Japan (JISC)	Approval with comments
Korea, Republic of (KATS)	Approval with comments
Lebanon (LIBNOR)	Approval
Luxembourg (ILNAS)	Abstention
Malaysia (DSM)	Abstention
Mauritius (MSB)	Abstention
Mexico (DGN)	Approval
Netherlands (NEN)	Abstention
New Zealand (NZSO)	Abstention
Norway (SN)	Abstention
Panama (COPANIT)	Approval
Peru (INACAL)	Approval
Poland (PKN)	Abstention
Romania (ASRO)	Abstention
Russian Federation (GOST R)	Abstention
Singapore (SSC)	Abstention
Slovakia (UNMS SR)	Abstention
South Africa (SABS)	Approval
Spain (UNE)	Abstention
Sri Lanka (SLSI)	Approval
Sweden (SIS)	Abstention
Switzerland (SNV)	Abstention
Ukraine (DSTU)	Abstention
United Arab Emirates (ESMA)	Abstention
United Kingdom (BSI)	Approval with comments

United States (ANSI)	Disapproval
Uruguay (UNIT)	Abstention

Answers to Q.1: "Do you approve the circulation of the draft as a DIS?"		
9 x	Approval	Belgium (NBN) Iran, Islamic Republic of (ISIRI) Ireland (NSAI) Lebanon (LIBNOR) Mexico (DGN) Panama (COPANIT) Peru (INACAL) South Africa (SABS) Sri Lanka (SLSI)
5 x	Approval with comments	China (SAC) France (AFNOR) Japan (JISC) Korea, Republic of (KATS) United Kingdom (BSI)
2 x	Disapproval	Germany (DIN) United States (ANSI)
32 x	Abstention	Algeria (IANOR) Argentina (IRAM) Australia (SA) Austria (ASI) Brazil (ABNT) Canada (SCC) Costa Rica (INTECO) Cyprus (CYS) Côte d'Ivoire (CODINORM) Denmark (DS) Finland (SFS) India (BIS) Indonesia (BSN) Israel (SII) Italy (UNI) Luxembourg (ILNAS) Malaysia (DSM) Mauritius (MSB) Netherlands (NEN) New Zealand (NZSO) Norway (SN) Poland (PKN) Romania (ASRO) Russian Federation (GOST R) Singapore (SSC) Slovakia (UNMS SR) Spain (UNE) Sweden (SIS) Switzerland (SNV) Ukraine (DSTU) United Arab Emirates (ESMA) Uruguay (UNIT)

Comments from Voters		
Member:	Comment:	Date:
China (SAC)	<i>Comment File</i>	2019-09-03 10:26:45
France (AFNOR)	<i>Comment File</i>	2019-09-06 17:49:47
Germany (DIN)	<i>Comment File</i>	2019-09-03 15:31:30
Japan (JISC)	<i>Comment File</i>	2019-09-04 11:32:49
Korea, Republic of (KATS)	<i>Comment File</i>	2019-09-05 02:33:16
United Kingdom (BSI)	<i>Comment File</i>	2019-09-06 10:55:30
United States (ANSI)	<i>Comment File</i>	2019-08-30 14:41:28

Comments from Commenters		
Member:	Comment:	Date: