

<b>COMMITTEE DRAFT</b>		Reference document: <b>SC 27 N19507</b>	
<b>ISO/IEC 3<sup>rd</sup> CD 15408-3, revision</b>			
Date: <b>2019-07-12</b>		Supersedes document: N18805	
THIS DOCUMENT IS STILL UNDER STUDY AND SUBJECT TO CHANGE. IT SHOULD NOT BE USED FOR REFERENCE PURPOSES.			
ISO/IEC JTC 1/SC 27 Information security, cybersecurity and privacy protection Secretariat: Germany		Circulated to P- and O-members, and to technical committees and organizations in liaison for comments by: <b>2019-09-06</b>  Please submit your comments via the online balloting application by the due date indicated.	
<b>ISO/IEC 3<sup>rd</sup> CD 15408-3, revision</b>			
<b>Title: IT Security techniques – Evaluation criteria for IT security — Part 3: Security assurance components</b>			
Project: 1.27.16.03 (ISO/IEC 15408-3, revision)			
<b>Explanatory Report</b>			
<b>Status</b>	<b>SC 27 Decision</b>	<b>Reference documents</b>	
		<b>Input</b>	<b>Output</b>
<i>For details regarding previous development stages refer to 2<sup>nd</sup> page of this explanatory report.</i>			
<b>ISO/IEC 15408-3</b> <b>1<sup>st</sup> WD</b>	54 <sup>th</sup> WG 3 meeting, April 2017, Recommendations 5,10, 11, 14 (N17041 = WG 3 N1413).	SoV (N17027).	Liaisons to: CCDB (WG 3 N1391); The Open Group (WG 3 N1394); ISO/TC 22/SC 32 (N17373); Text f. 1 <sup>st</sup> WD (WG 3 N1437).
<b>ISO/IEC 15408-3</b> <b>2<sup>nd</sup> WD</b>	55 <sup>th</sup> WG 3 meeting, Oct. 2017, Recommendations 8, 10 (N17666 = WG 3 N1494).	SoCom (WG 3 N1467); Draft DoC (WG 3 N1501).	Editor's report (WG 3 N1465); Liaisons to: CCDB (WG 3 N1455); ISO/TC 22/SC 32 (N18103); DoC (WG 3 N1462); Text f. 2 <sup>nd</sup> WD (WG 3 N1469).
<b>ISO/IEC 15408-3</b> <b>1<sup>st</sup> CD</b>	56 <sup>th</sup> WG 3 meeting, April 2018, Recommendations 8, 10 (N18471 = WG 3 N1557) / 30 <sup>th</sup> SC 27 Plenary, April 2018, Resolution 6 (N18710).	SoCom (WG 3 N1530); Late Com (WG 3 N1564).	Liaison to: CCDB (WG 3 N1521); DoC (WG 3 N1527); Text f. 1 <sup>st</sup> CD (N18752).
<b>ISO/IEC 15408-3</b> <b>2<sup>nd</sup> CD</b>	57 <sup>th</sup> WG 3 meeting / CRM, Sep / Oct 2018, Recommendations 11, 14 (N18820 = WG 3 N1610) .	SoV (N18853).	Liaison to: CCDB (WG 3 N1619); DoC (N18802); Text f. 2 <sup>nd</sup> CD (N18805).
<b>ISO/IEC 15408-3</b> <b>3<sup>rd</sup> CD</b>	58 <sup>th</sup> WG 3 meeting / CRM April 2019, Recommendations 12, 14, 17, 21 (N19523 = WG 3 N1676).	SoV (N19489).	Liaison to: CCDB (WG 3 N1680); DoC (N19504); Text f. 3 <sup>rd</sup> CD (N19507).
<b>3<sup>rd</sup> CD Consideration</b>			
<b>In accordance with Recommendation 14 (see SC 27 N19523) of the 58<sup>th</sup> 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 3<sup>rd</sup> CD letter ballot closing by</b>			
<b>2019-09-06</b>			
Medium: <a href="http://isotc.iso.org/livelink/livelink/open/jtc1sc27">http://isotc.iso.org/livelink/livelink/open/jtc1sc27</a>			
No. of pages: 2 + 222			

Explanatory Report			
Status	SC 27 Decision	Reference documents	
		Input	Output
<b>Study Period IT security testing, evaluation and assurance standards and techniques</b>	51 <sup>st</sup> WG 3 meeting, Oct. 2015, Recommendations 5, 6 (N15594 = WG 3 N1251).		Terms of Reference (WG 5 N1258); 1 <sup>st</sup> /2 <sup>nd</sup> call f. contr. (WG 3 N1259 /1317)..
	52 <sup>nd</sup> WG 3 meeting, April 2016, Recommendation 5, 7 (N16026 = WG 3 N1296).	Expert contr. (WG 3 N1299, 1301).	3 <sup>rd</sup> call f. contr. (WG 3 N1377); Rapporteur's report (WG 3 N1320); Liaison to: CCDB (WG 3 N1266).
<b>ISO/IEC NP 15408-3 (revision) Evaluation criteria for IT security -- Part 3 NWIP</b>	53 <sup>rd</sup> WG 3 meeting, Oct. 2016, Recommendations 6, 15 (N16800 = WG 5 N600).	Expert contr. (WG 3 N1368, N1371, N1373).	SP report (WG 3 N1363); Call f. editor (WG 3 N1387 = N16886); Liaisons to: CCDB (WG 3 N1330 ); The Open Group (WG 3 N1332); Text f. NWIP (N16965 [replaces N16883]).

ISO/IEC 15408-3:2017(E)

ISO/IEC 15408-3:XXXX(E)

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

Secretariat: DIN

**IT Security Techniques — 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

3, rue de Varembé

Case postale 401

Case postale 131

CH-1214 Vernier, Geneva

CH-1211 Geneva 20

Tel. + 41 22 749 01 11

Tel. + 41 22 919 02 11

Fax + 41 22 733 34 30

Fax + 41 22 919 03 00

E-mail [central@iso.org](mailto:central@iso.org) E-mail [inmail@iec.org](mailto:inmail@iec.org)

Web [www.iso.org](http://www.iso.org) Web [www.iec.org](http://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	4.1 Organisation of this document.....	1
7	5 Assurance paradigm.....	2
8	5.1 Introduction.....	2
9	5.2 ISO/IEC 15408 philosophy.....	2
10	5.3 Assurance approach.....	2
11	5.3.1 Introduction.....	2
12	5.3.2 Significance of vulnerabilities .....	2
13	5.3.3 Cause of vulnerabilities.....	3
14	5.3.4 ISO/IEC 15408 series assurance .....	3
15	5.3.5 Assurance through evaluation .....	3
16	5.4 ISO/IEC 15408 series evaluation assurance scale .....	4
17	6 Security assurance components .....	4
18	6.1 Security assurance classes, families and components structure.....	4
19	6.2 Assurance class structure .....	4
20	6.2.1 Class name.....	4
21	6.2.2 Class introduction .....	5
22	6.2.3 Assurance families.....	5
23	6.3 Assurance family structure .....	65
24	6.3.1 Family name .....	65
25	6.3.2 Objectives.....	6
26	6.3.3 Component levelling.....	6
27	6.3.4 Application notes .....	6
28	6.3.5 Assurance components.....	6
29	6.4 Assurance component structure.....	6
30	6.4.1 Component identification .....	7
31	6.4.2 Objectives.....	7
32	6.4.3 Application notes .....	7
33	6.4.4 Dependencies.....	7
34	6.4.5 Assurance elements.....	87
35	6.5 Assurance elements.....	98
36	6.6 Component taxonomy.....	98
37	7 Class APE: Protection Profile evaluation .....	9
38	7.1 Introduction.....	9
39	7.2 PP introduction (APE_INT) .....	109
40	7.2.1 Objectives.....	109
41	7.2.2 APE_INT.1 PP introduction .....	109
42	7.3 Conformance claims (APE_CCL).....	10
43	7.3.1 Objectives.....	10
44	7.3.2 APE_CCL.1 Conformance claims.....	1110
45	7.4 Security problem definition (APE_SPD).....	1312
46	7.4.1 Objectives.....	1312

47	7.4.2	APE_SPD.1 Security problem definition .....	1342
48	7.5	Security objectives (APE_OBJ) .....	1443
49	7.5.1	Objectives .....	1443
50	7.5.2	Component levelling .....	1443
51	7.5.3	APE_OBJ.1 Security objectives for the operational environment .....	1443
52	7.5.4	APE_OBJ.2 Security objectives .....	14
53	7.6	Extended components definition (APE_ECD) .....	15
54	7.6.1	Objectives .....	15
55	7.6.2	APE_ECD.1 Extended components definition .....	15
56	7.7	Security requirements (APE_REQ) .....	16
57	7.7.1	Objectives .....	16
58	7.7.2	Component levelling .....	1746
59	7.7.3	APE_REQ.1 Stated security requirements .....	1746
60	7.7.4	APE_REQ.2 Derived security requirements .....	18
61	8	Class ACE: Protection Profile Configuration evaluation .....	2019
62	8.1	Introduction .....	2019
63	8.2	PP-Module introduction (ACE_INT) .....	20
64	8.2.1	Objectives .....	20
65	8.2.2	ACE_INT.1 PP-Module introduction .....	20
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) .....	23
70	8.4.1	Objectives .....	23
71	8.4.2	ACE_SPD.1 PP-Module Security problem definition .....	2423
72	8.5	PP-Module Security objectives (ACE_OBJ) .....	24
73	8.5.1	Objectives .....	24
74	8.5.2	Component levelling .....	24
75	8.5.3	ACE_OBJ.1 Direct Rationale PP-Module Security objectives .....	2524
76	8.5.4	Application notes .....	Fehler! Textmarke nicht definiert.24
77	8.5.5	ACE_OBJ.2 PP-Module Security objectives .....	25
78	8.5.6	Application notes .....	Fehler! Textmarke nicht definiert.25
79	8.6	PP-Module extended components definition (ACE_ECD) .....	26
80	8.6.1	Objectives .....	26
81	8.6.2	ACE_ECD.1 PP-Module extended components definition .....	26
82	8.7	PP-Module security requirements (ACE_REQ) .....	27
83	8.7.1	Objectives .....	27
84	8.7.2	Component levelling .....	27
85	8.7.3	ACE_REQ.1 PP-Module stated security requirements .....	27
86	8.7.4	ACE_REQ.2 PP-Module derived security requirements .....	29
87	8.8	PP-Module consistency (ACE_MCO) .....	3130
88	8.8.1	Objectives .....	3130
89	8.8.2	ACE_MCO.1 PP-Module consistency .....	31
90	8.9	PP-Configuration consistency (ACE_CCO) .....	32
91	8.9.1	Objectives .....	32
92	8.9.2	ACE_CCO.1 PP-Configuration consistency .....	32
93	9	Class ASE: Security Target evaluation .....	3635
94	9.1	Introduction .....	3635
95	9.2	ST introduction (ASE_INT) .....	3735
96	9.2.1	Objectives .....	3735
97	9.2.2	ASE_INT.1 ST introduction .....	3736
98	9.3	Conformance claims (ASE_CCL) .....	3837

99	9.3.1	Objectives.....	3837
100	9.3.2	ASE_CCL.1 Conformance claims.....	3837
101	9.4	Security problem definition (ASE_SPD) .....	4038
102	9.4.1	Objectives.....	4038
103	9.4.2	ASE_SPD.1 Security problem definition .....	4138
104	9.5	Security objectives (ASE_OBJ).....	4139
105	9.5.1	Objectives.....	4139
106	9.5.2	Component levelling.....	4139
107	9.5.3	ASE_OBJ.1 Security objectives for the operational environment.....	4239
108	9.5.4	ASE_OBJ.2 Security objectives.....	4240
109	9.6	Extended components definition (ASE_ECD).....	4341
110	9.6.1	Objectives.....	4341
111	9.6.2	ASE_ECD.1 Extended components definition.....	4341
112	9.7	Security requirements (ASE_REQ).....	4442
113	9.7.1	Objectives.....	4442
114	9.7.2	Component levelling.....	4442
115	9.7.3	ASE_REQ.1 Stated security requirements.....	4442
116	9.7.4	ASE_REQ.2 Derived security requirements.....	4644
117	9.8	TOE summary specification (ASE_TSS).....	4845
118	9.8.1	Objectives.....	4845
119	9.8.2	Component levelling.....	4845
120	9.8.3	ASE_TSS.1 TOE summary specification .....	4846
121	9.8.4	ASE_TSS.2 TOE summary specification with architectural design summary .....	4946
122	9.9	Consistency of composite product Security Target (ASE_COMP).....	5047
123	9.9.1	Objectives.....	5047
124	9.9.2	ASE_COMP.1 ..... Consistency of Security Target	5047
125	10	Class ADV: Development .....	5249
126	10.1	Introduction.....	5249
127	10.2	Security Architecture (ADV_ARC) .....	5754
128	10.2.1	Objectives.....	5754
129	10.2.2	Component levelling.....	5755
130	10.2.3	Application notes .....	5755
131	10.2.4	ADV_ARC.1 Security architecture description.....	5855
132	10.3	Functional specification (ADV_FSP).....	5957
133	10.3.1	Objectives.....	5957
134	10.3.2	Component levelling.....	5957
135	10.3.3	Application notes .....	5957
136	10.3.4	ADV_FSP.1 Basic functional specification .....	6260
137	10.3.5	ADV_FSP.2 Security-enforcing functional specification .....	6360
138	10.3.6	ADV_FSP.3 Functional specification with complete summary.....	6461
139	10.3.7	ADV_FSP.4 Complete functional specification.....	6563
140	10.3.8	ADV_FSP.5 Complete semi-formal functional specification with additional error information.....	6664
141	10.3.9	ADV_FSP.6 Complete semi-formal functional specification with additional formal specification.....	6765
142	10.4	Implementation representation (ADV_IMP) .....	6866
143	10.4.1	Objectives.....	6866
144	10.4.2	Component levelling.....	6967
145	10.4.3	Application notes .....	6967
146	10.4.4	ADV_IMP.1 Implementation representation of the TSF.....	7068
147	10.4.5	ADV_IMP.2 Complete mapping of the implementation representation of the TSF .....	7168
148	10.5	TSF internals (ADV_INT) .....	7169

151	10.5.1 Objectives.....	7169
152	10.5.2 Component levelling.....	7269
153	10.5.3 Application notes .....	7270
154	10.5.4 ADV_INT.1 Well-structured subset of TSF internals.....	7270
155	10.5.5 ADV_INT.2 Well-structured internals.....	7374
156	10.5.6 ADV_INT.3 Minimally complex internals.....	7472
157	10.6 Security policy modelling (ADV_SPM) .....	7573
158	10.6.1 Objectives.....	7573
159	10.6.2 Component levelling.....	7573
160	10.6.3 Application notes .....	7673
161	10.6.4 ADV_SPM.1 Formal TOE security policy model.....	7774
162	10.7 TOE design (ADV_TDS).....	7876
163	10.7.1 Objectives.....	7876
164	10.7.2 Component levelling.....	7976
165	10.7.3 Application notes .....	7977
166	10.7.4 ADV_TDS.1 Basic design .....	8078
167	10.7.5 ADV_TDS.2 Architectural design .....	8179
168	10.7.6 ADV_TDS.3 Basic modular design.....	8280
169	10.7.7 ADV_TDS.4 Semiformal modular design .....	8482
170	10.7.8 ADV_TDS.5 Complete semiformal modular design .....	8583
171	10.7.9 ADV_TDS.6 Complete semiformal modular design with formal high-level design presentation .....	8684
172	10.8 Composite design compliance (ADV_COMP).....	8886
173	10.8.1 Objectives.....	8886
174	10.8.2 Component levelling.....	8886
175	10.8.3 Application notes .....	8886
176	10.8.4 ADV_COMP.1 Design compliance with the platform certification report, guidance and ETR_COMP .....	8987
177		
178	11 Class AGD: Guidance documents.....	9087
179	11.1 Introduction.....	9087
180	11.2 Operational user guidance (AGD_OPE) .....	9088
181	11.2.1 Objectives.....	9088
182	11.2.2 Component levelling.....	9188
183	11.2.3 Application notes .....	9188
184	11.2.4 AGD_OPE.1 Operational user guidance .....	9189
185	11.3 Preparative procedures (AGD_PRE) .....	9290
186	11.3.1 Objectives.....	9290
187	11.3.2 Component levelling.....	9290
188	11.3.3 Application notes .....	9390
189	11.3.4 AGD_PRE.1 Preparative procedures .....	9390
190	12 Class ALC: Life-cycle support.....	9491
191	12.1 Introduction.....	9491
192	12.2 CM capabilities (ALC_CMC) .....	9592
193	12.2.1 Objectives.....	9592
194	12.2.2 Component levelling.....	9593
195	12.2.3 Application notes .....	9593
196	12.2.4 ALC_CMC.1 Labelling of the TOE.....	9694
197	12.2.5 ALC_CMC.2 Use of the CM system .....	9694
198	12.2.6 ALC_CMC.3 Authorisation controls .....	9795
199	12.2.7 ALC_CMC.4 Production support, acceptance procedures and automation .....	9997
200	12.2.8 ALC_CMC.5 Advanced support.....	10199
201	12.3 CM scope (ALC_CMS).....	104102
202	12.3.1 Objectives.....	104102



203	12.3.2 Component levelling.....	104102
204	12.3.3 Application notes .....	104102
205	12.3.4 ALC_CMS.1 TOE CM coverage.....	104102
206	12.3.5 ALC_CMS.2 Parts of the TOE CM coverage.....	105103
207	12.3.6 ALC_CMS.3 Implementation representation CM coverage.....	106104
208	12.3.7 ALC_CMS.4 Problem tracking CM coverage .....	107105
209	12.3.8 ALC_CMS.5 Development tools CM coverage.....	108106
210	12.4 Delivery (ALC_DEL).....	109107
211	12.4.1 Objectives.....	109107
212	12.4.2 Component levelling.....	109107
213	12.4.3 Application notes .....	109107
214	12.4.4 ALC_DEL.1 Delivery procedures.....	110108
215	12.5 Developer environment security (ALC_DVS) .....	110108
216	12.5.1 Objectives.....	110108
217	12.5.2 Component levelling.....	110108
218	12.5.3 Application notes .....	110108
219	12.5.4 ALC_DVS.1 Identification of security controls.....	111109
220	12.5.5 ALC_DVS.2 Sufficiency of security controls .....	111109
221	12.6 Flaw remediation (ALC_FLR).....	112110
222	12.6.1 Objectives.....	112110
223	12.6.2 Component levelling.....	112110
224	12.6.3 Application notes .....	112110
225	12.6.4 ALC_FLR.1 Basic flaw remediation .....	113111
226	12.6.5 ALC_FLR.2 Flaw reporting procedures.....	113111
227	12.6.6 ALC_FLR.3 Systematic flaw remediation.....	115113
228	12.7 Development Life-cycle definition (ALC_LCD).....	117115
229	12.7.1 Objectives.....	117115
230	12.7.2 Component levelling.....	117115
231	12.7.3 Application notes .....	117115
232	12.7.4 ALC_LCD.1 Developer defined life-cycle processes.....	118116
233	12.7.5 ALC_LCD.2 Measurable life-cycle model.....	118116
234	12.8 TOE Development Artifacts (ALC_TDA).....	119117
235	12.8.1 Objectives.....	119117
236	12.8.2 Component levelling.....	120118
237	12.8.3 Application notes .....	120118
238	12.8.4 ALC_TDA.1 Uniquely identifying implementation representation.....	120118
239	12.8.5 ALC_TDA.2 Matching CMS scope of implementation representation.....	123121
240	12.8.6 ALC_TDA.3 Regenerate TOE with well-defined development tools .....	125123
241	12.9 Tools and techniques (ALC_TAT) .....	128126
242	12.9.1 Objectives.....	128126
243	12.9.2 Component levelling.....	128126
244	12.9.3 Application notes .....	128126
245	12.9.4 ALC_TAT.1 Well-defined development tools .....	128126
246	12.9.5 ALC_TAT.2 Compliance with implementation standards.....	129127
247	12.9.6 ALC_TAT.3 Compliance with implementation standards - all parts.....	130128
248	12.10 Integration of composition parts and consistency check of delivery procedures	
249	(ALC_COMP) .....	131129
250	12.10.1 .....	Objectives 131129
251	12.10.2 .....	Component levelling 131129
252	12.10.3 .....	ALC_COMP.1 Integration of the application into the underlying
253	13 Class ATE: Tests .....	132130
254	13.1 Introduction.....	132130

255	13.2 Coverage (ATE_COV).....	133131
256	13.2.1 Objectives.....	133131
257	13.2.2 Component levelling.....	133131
258	13.2.3 Application notes .....	133131
259	13.2.4 ATE_COV.1 Evidence of coverage .....	133131
260	13.2.5 ATE_COV.2 Analysis of coverage .....	134132
261	13.2.6 ATE_COV.3 Rigorous analysis of coverage.....	135133
262	13.3 Depth (ATE_DPT).....	135133
263	13.3.1 Objectives.....	135133
264	13.3.2 Component levelling.....	136134
265	13.3.3 Application notes .....	136134
266	13.3.4 ATE_DPT.1 Testing: basic design .....	136134
267	13.3.5 ATE_DPT.2 Testing: security enforcing modules.....	137135
268	13.3.6 ATE_DPT.3 Testing: modular design .....	138136
269	13.3.7 ATE_DPT.4 Testing: implementation representation .....	138136
270	13.4 Functional tests (ATE_FUN) .....	139137
271	13.4.1 Objectives.....	139137
272	13.4.2 Component levelling.....	140138
273	13.4.3 Application notes .....	140138
274	13.4.4 ATE_FUN.1 Functional testing.....	140138
275	13.4.5 ATE_FUN.2 Ordered functional testing.....	141139
276	13.5 Independent testing (ATE_IND) .....	142140
277	13.5.1 Objectives.....	142140
278	13.5.2 Component levelling.....	142140
279	13.5.3 Application notes .....	142140
280	13.5.4 ATE_IND.1 Independent testing - conformance.....	143141
281	13.5.5 ATE_IND.2 Independent testing - sample .....	144142
282	13.5.6 ATE_IND.3 Independent testing - complete .....	145143
283	13.6 Composite functional testing (ATE_COMP) .....	146144
284	13.6.1 Objectives.....	146144
285	13.6.2 Application notes .....	146144
286	13.6.3 ATE_COMP.1 Composite product functional testing.....	147145
287	14 Class AVA: Vulnerability assessment.....	147145
288	14.1 Introduction.....	147145
289	14.2 Application notes .....	148146
290	14.3 Vulnerability analysis (AVA_VAN).....	148146
291	14.3.1 Objectives.....	148146
292	14.3.2 Component levelling.....	149147
293	14.3.3 AVA_VAN.1 Vulnerability survey.....	149147
294	14.3.4 AVA_VAN.2 Vulnerability analysis.....	150148
295	14.3.5 AVA_VAN.3 Focused vulnerability analysis.....	151149
296	14.3.6 AVA_VAN.4 Methodical vulnerability analysis .....	152150
297	14.3.7 AVA_VAN.5 Advanced methodical vulnerability analysis .....	153151
298	14.4 Composite vulnerability assessment (AVA_COMP).....	155153
299	14.4.1 Objectives.....	155153
300	14.4.2 AVA_COMP.1 Composite product vulnerability assessment .....	155153
301	15 Class ACO: Composition.....	156154
302	15.1 Introduction.....	156154
303	15.2 Composition rationale (ACO_COR) .....	159157
304	15.2.1 Objectives.....	159157
305	15.2.2 Component levelling.....	159157
306	15.2.3 ACO_COR.1 Composition rationale .....	159157

307	<b>15.3 Development evidence (ACO_DEV)</b> .....	<b>159157</b>
308	15.3.1 Objectives.....	159157
309	15.3.2 Component levelling.....	159157
310	15.3.3 Application notes .....	160158
311	15.3.4 ACO_DEV.1 Functional Description .....	160158
312	15.3.5 ACO_DEV.2 Basic evidence of design.....	161159
313	15.3.6 ACO_DEV.3 Detailed evidence of design.....	162160
314	<b>15.4 Reliance of dependent component (ACO_REL)</b> .....	<b>163161</b>
315	15.4.1 Objectives.....	163161
316	15.4.2 Component levelling.....	163161
317	15.4.3 Application notes .....	163161
318	15.4.4 ACO_REL.1 Basic reliance information .....	164162
319	15.4.5 ACO_REL.2 Reliance information .....	164162
320	<b>15.5 Composed TOE testing (ACO_CTT)</b> .....	<b>165163</b>
321	15.5.1 Objectives.....	165163
322	15.5.2 Component levelling.....	165163
323	15.5.3 Application notes .....	165163
324	15.5.4 ACO_CTT.1 Interface testing.....	166164
325	15.5.5 ACO_CTT.2 Rigorous interface testing.....	167165
326	<b>15.6 Composition vulnerability analysis (ACO_VUL)</b> .....	<b>168166</b>
327	15.6.1 Objectives.....	168166
328	15.6.2 Component levelling.....	168166
329	15.6.3 Application notes .....	168166
330	15.6.4 ACO_VUL.1 Composition vulnerability review .....	169167
331	15.6.5 ACO_VUL.2 Composition vulnerability analysis.....	170168
332	15.6.6 ACO_VUL.3 Enhanced-Basic Composition vulnerability analysis .....	170168
333	<b>Annex A (informative) Development (ADV)</b> .....	<b>172170</b>
334	A.1 ADV_ARC: Supplementary material on security architectures.....	172170
335	A.1.1 Security architecture properties .....	172170
336	A.1.2 Security architecture descriptions.....	173171
337	A.2 ADV_FSP: Supplementary material on functional specification.....	175173
338	A.2.1 Non-TSF part of the TOE .....	176174
339	A.2.2 Determining the TSFI .....	177175
340	A.2.3 Example: A complex DBMS .....	180178
341	A.2.4 Example Functional Specification.....	181179
342	A.3 ADV_INT: Supplementary material on TSF internals .....	183181
343	A.3.1 Structure of procedural software .....	183181
344	A.3.2 Complexity of procedural software.....	185183
345	A.4 ADV_TDS: Subsystems and Modules.....	186184
346	A.4.1 Subsystems.....	186184
347	A.4.2 Modules.....	187185
348	A.4.3 Levelling Approach.....	189187
349	A.4.4 Security relevance.....	191189
350	A.5 Supplementary material on formal methods .....	192190
351	<b>Annex B (informative) Composition (ACO)</b> .....	<b>194192</b>
352	B.1 Necessity for composed TOE evaluations.....	194192
353	B.2 Performing Security Target evaluation for a composed TOE .....	195193
354	B.3 Interactions between composed IT entities .....	196194
355	<b>Annex C (informative) Cross reference of assurance component dependencies</b> .....	<b>202200</b>
356		

## 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, *IT Security techniques*.

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.

385 **Legal Notice**

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

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

## Introduction

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

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

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

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

# Information technology Security techniques — 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) and Security Targets (STs).

## 2 Normative references

The following referenced documents are indispensable for the application of this document and shall be used together with this part of the ISO/IEC 15408 series. 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, IT security techniques -- Evaluation criteria for IT security — *Part 1: Introduction and general model*

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

ISO/IEC 15408-5, IT security techniques -- 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

### 4.1 Organisation of this document

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:



- 496 a) eliminated - that is, active steps should be taken to expose, and remove or neutralise, all  
497 exercisable vulnerabilities;
- 498 b) minimised - that is, active steps should be taken to reduce, to an acceptable residual level, the  
499 potential impact of any exercise of a vulnerability;
- 500 c) monitored - that is, active steps should be taken to ensure that any attempt to exercise a residual  
501 vulnerability will be detected so that steps can be taken to limit the damage.

### 502 5.3.3 Cause of vulnerabilities

503 Vulnerabilities can arise through failures in:

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

### 521 5.3.4 ISO/IEC 15408 series assurance

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

### 526 5.3.5 Assurance through evaluation

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

- 529 a) analysis and checking of process(es) and procedure(s);
- 530 b) checking that process(es) and procedure(s) are being applied;
- 531 c) analysis of the correspondence between TOE design representations;

- 532 d) analysis of the TOE design representation against the requirements;
- 533 e) verification of proofs;
- 534 f) analysis of guidance documents;
- 535 g) analysis of functional tests developed and the results provided;
- 536 h) independent functional testing;
- 537 i) analysis for vulnerabilities (including flaw hypotheses);
- 538 j) penetration testing;
- 539 k) analysis of the delivery process;
- 540 l) analysis of the maintenance process.

#### 541 **5.4 ISO/IEC 15408 series evaluation assurance scale**

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

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

### 549 **6 Security assurance components**

#### 550 **6.1 Security assurance classes, families and components structure**

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

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

#### 558 **6.2 Assurance class structure**

559 Figure 1 illustrates the assurance class structure.

##### 560 **6.2.1 Class name**

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

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

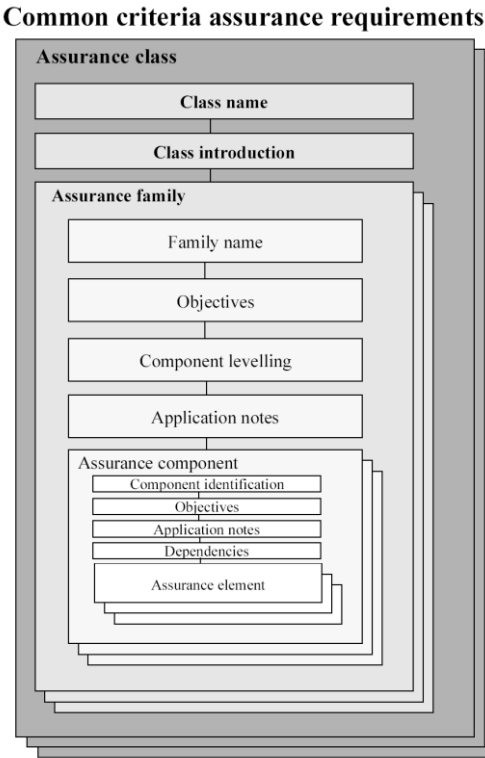
566       **6.2.2 Class introduction**

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

569       **6.2.3 Assurance families**

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

572 Figure 1 illustrates the assurance family structure.



573

574       **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

Figure 2 illustrates the assurance component structure.

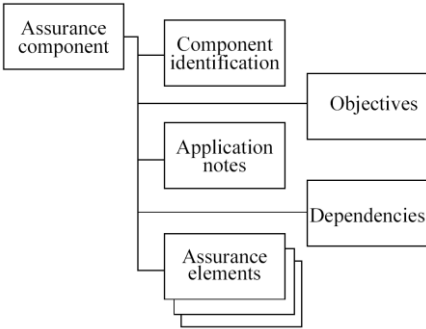


Figure 2 — Assurance component structure

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.

**6.4.1 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.

**6.4.2 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.

**6.4.3 Application notes**

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

**6.4.4 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.

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

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

#### 640 **6.4.5 Assurance elements**

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

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

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

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

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

669

670 The developer action elements, content and presentation of evidence elements, and explicit evaluator  
671 action elements, identify the evaluator effort that shall be expended in verifying the security claims  
672 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.

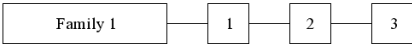


Figure 3 — Sample class decomposition diagram

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.

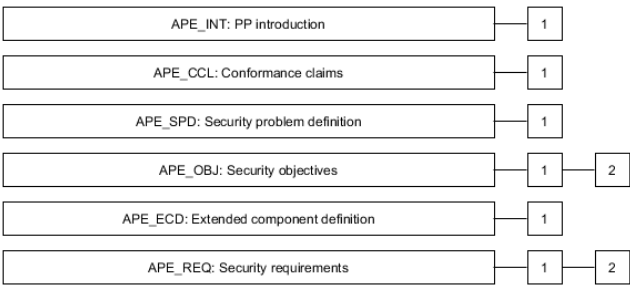


Figure 4 — APE: Protection Profile evaluation class decomposition

699 **7.2 PP introduction (APE\_INT)**

700 **7.2.1 Objectives**

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

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

704 **7.2.2 APE\_INT.1 PP introduction**

705 Dependencies: No dependencies.

706 **7.2.2.1 Developer action elements**

707 **7.2.2.1.1 APE\_INT.1.1D**

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

709 **7.2.2.2 Content and presentation elements**

710 **7.2.2.2.1 APE\_INT.1.1C**

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

712 **7.2.2.2.2 APE\_INT.1.2C**

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

714 **7.2.2.2.3 APE\_INT.1.3C**

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

716 **7.2.2.2.4 APE\_INT.1.4C**

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

718 **7.2.2.2.5 APE\_INT.1.5C**

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

721 **7.2.2.3 Evaluator action elements**

722 **7.2.2.3.1 APE\_INT.1.1E**

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

725 **7.3 Conformance claims (APE\_CCL)**

726 **7.3.1 Objectives**

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



**7.3.2 APE\_CCL.1 Conformance claims**

Dependencies: APE\_INT.1 PP introduction

APE\_ECD.1 Extended components definition

[APE\\_REQ.1 Direct rationale PP-Module security requirements](#) [APE\\_REQ.1 Stated security requirements](#)

**7.3.2.1 Developer action elements**

**7.3.2.1.1 APE\_CCL.1.1D**

The developer shall provide a conformance claim.

**7.3.2.1.2 APE\_CCL.1.2D**

The developer shall provide a conformance claim rationale.

**7.3.2.1.3 APE\_CCL.1.3D**

The developer shall provide a conformance statement.

**7.3.2.2 Content and presentation elements**

**7.3.2.2.1 APE\_CCL.1.1C**

The conformance claim shall contain an ISO/IEC 15408 series conformance claim that identifies the ISO/IEC 15408-1 edition to which the PP claims conformance.

**7.3.2.2.2 APE\_CCL.1.2C**

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

**7.3.2.2.3 APE\_CCL.1.3C**

The ISO/IEC 15408 series conformance claim shall describe the conformance of the PP to this document as either "ISO/IEC 15408-3 conformant" or ISO/IEC 15408-3 extended."

**7.3.2.2.4 APE\_CCL.1.4C**

The ISO/IEC 15408 series conformance claim shall be consistent with the extended components definition.

**7.3.2.2.5 APE\_CCL.1.5C**

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

**7.3.2.2.6 APE\_CCL.1.6C**

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

759 7.3.2.2.7 APE\_CCL.1.7C

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

762 7.3.2.2.8 APE\_CCL.1.8C

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 PPs for  
765 which conformance is being claimed.

766 7.3.2.2.9 APE\_CCL.1.9C

767 The conformance claim rationale shall demonstrate that the statement of security objectives is  
768 consistent with the statement of security objectives in the PPs for which conformance is being  
769 claimed.

770 7.3.2.2.10 APE\_CCL.1.10C

771 The conformance claim rationale shall demonstrate that the statement of security  
772 requirements is consistent with the statement of security requirements in the PPs for which  
773 conformance is being claimed.

774 7.3.2.2.11 APE\_CCL.1.11C

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 7.3.2.2.12 APE\_CCL.1.12C

778 The conformance statement shall identify the set of PPs (if any) to which, in combination with  
779 the PP under evaluation, exact conformance is allowed to be claimed.

780 7.3.2.2.13 APE\_CCL.1.13C

781 The conformance statement shall identify the set of PP-modules (if any) that are allowed to be  
782 used with the PP under evaluation in a PP-Configuration.

783 7.3.2.2.14 APE\_CCL.1.14C

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

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

792 **7.3.2.3 Evaluator action elements**

793 **7.3.2.3.1 APE\_CCL.1.1E**

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

796 **7.4 Security problem definition (APE\_SPD)**

797 **7.4.1 Objectives**

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

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

802 **7.4.2 APE\_SPD.1 Security problem definition**

803 Dependencies: No dependencies.

804 **7.4.2.1 Developer action elements**

805 **7.4.2.1.1 APE\_SPD.1.1D**

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

807 **7.4.2.2 Content and presentation elements**

808 **7.4.2.2.1 APE\_SPD.1.1C**

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

810 **7.4.2.2.2 APE\_SPD.1.2C**

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

812 **7.4.2.2.3 APE\_SPD.1.3C**

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

814 **7.4.2.2.4 APE\_SPD.1.4C**

815 **The security problem definition shall describe the assumptions about the operational**  
816 **environment of the TOE.**

817 **7.4.2.3 Evaluator action elements**

818 **7.4.2.3.1 APE\_SPD.1.1E**

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

821 **7.5 Security objectives (APE\_OBJ)**

822 **7.5.1 Objectives**

823 The security objectives are a concise statement of the intended response to the security problem  
824 defined through the Security problem definition (APE\_SPD) family.

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

828 **7.5.2 Component levelling**

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

831 **7.5.3 APE\_OBJ.1 Security objectives for the operational environment**

832 Dependencies: No dependencies.

833 **7.5.3.1 Developer action elements**

834 **7.5.3.1.1 APE\_OBJ.1.1D**

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

836 **7.5.3.2 Content and presentation elements**

837 **7.5.3.2.1 APE\_OBJ.1.1C**

838 **The statement of security objectives shall describe the security objectives for the operational**  
839 **environment.**

840 **7.5.3.3 Evaluator action elements**

841 **7.5.3.3.1 APE\_OBJ.1.1E**

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

844 **7.5.4 APE\_OBJ.2 Security objectives**

845 Dependencies: APE\_SPD.1 Security problem definition

846 **7.5.4.1 Developer action elements**

847 **7.5.4.1.1 APE\_OBJ.2.1D**

848 The developer shall provide a statement of security objectives.

849 **7.5.4.1.2 APE\_OBJ.2.2D**

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

851 **7.5.4.2 Content and presentation elements**

852 **7.5.4.2.1 APE\_OBJ.2.1C**

853 The statement of security objectives shall describe the security objectives for the **TOE and the**  
854 **security objectives for the** operational environment.

855 **7.5.4.2.2 APE\_OBJ.2.2C**

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

858 **7.5.4.2.3 APE\_OBJ.2.3C**

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

862 **7.5.4.2.4 APE\_OBJ.2.4C**

863 **The security objectives rationale shall demonstrate that the security objectives counter all**  
864 **threats.**

865 **7.5.4.2.5 APE\_OBJ.2.5C**

866 **The security objectives rationale shall demonstrate that the security objectives enforce all**  
867 **OSPs.**

868 **7.5.4.2.6 APE\_OBJ.2.6C**

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

871 **7.5.4.3 Evaluator action elements**

872 **7.5.4.3.1 APE\_OBJ.2.1E**

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

875 **7.6 Extended components definition (APE\_ECD)**

876 **7.6.1 Objectives**

877 Extended security requirements are requirements that are not based on components from ISO/IEC  
878 15408-2 or this document, but are based on extended components: components defined by the PP  
879 author.

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

883 **7.6.2 APE\_ECD.1 Extended components definition**

884 Dependencies: No dependencies.

885 **7.6.2.1 Developer action elements**

886 **7.6.2.1.1 APE\_ECD.1.1D**

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

888 **7.6.2.1.2 APE\_ECD.1.2D**

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

890 **7.6.2.2 Content and presentation elements**

891 **7.6.2.2.1 APE\_ECD.1.1C**

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

893 **7.6.2.2.2 APE\_ECD.1.2C**

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

896 **7.6.2.2.3 APE\_ECD.1.3C**

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

899 **7.6.2.2.4 APE\_ECD.1.4C**

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

902 **7.6.2.2.5 APE\_ECD.1.5C**

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

905 **7.6.2.3 Evaluator action elements**

906 **7.6.2.3.1 APE\_ECD.1.1E**

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

909 **7.6.2.3.2 APE\_ECD.1.2E**

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

912 **7.7 Security requirements (APE\_REQ)**

913 **7.7.1 Objectives**

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

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

919 **7.7.2 Component levelling**

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

922 **7.7.3 APE\_REQ.1 Direct rationale PP-Module security requirements**

923 Dependencies: APE\_ECD.1 Extended components definition

924 **7.7.3.1 Developer action elements**

925 **7.7.3.1.1 APE\_REQ.1.1D**

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

927 **7.7.3.1.2 APE\_REQ.1.2D**

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

929 **7.7.3.2 Content and presentation elements**

930 **7.7.3.2.1 APE\_REQ.1.1C**

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

932 **7.7.3.2.2 APE\_REQ.1.2C**

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

935 **7.7.3.2.3 APE\_REQ.1.3C**

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

940 **7.7.3.2.4 APE\_REQ.1.4C**

941 **The statement of security requirements shall identify all operations on the security**  
942 **requirements.**

943 **7.7.3.2.5 APE\_REQ.1.5C**

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

945 **7.7.3.2.6 APE\_REQ.1.6C**

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

948 **7.7.3.2.7 APE\_REQ.1.7C**

949 The security requirements rationale shall trace each SFR back to the threats countered by that  
950 SFR and the OSPs enforced by that SFR.

951 **7.7.3.2.8 APE\_REQ.1.8C**

952 The security requirements rationale shall trace each security objective for the operational  
953 environment back to threats countered by that security objective, OSPs enforced by that  
954 security objective, and assumptions upheld by that security objective.

955 **7.7.3.2.9 APE\_REQ.1.9C**

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

958 **7.7.3.2.10 APE\_REQ.1.10C**

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

961 **7.7.3.2.11 APE\_REQ.1.11C**

962 The security requirements rationale shall demonstrate that the security objectives for the  
963 operational environment uphold all assumptions.

964 **7.7.3.2.12 APE\_REQ.1.12C**

965 The statement of security requirements shall be internally consistent.

966 **7.7.3.3 Evaluator action elements**

967 **7.7.3.3.1 APE\_REQ.1.1E**

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

970 **7.7.4 APE\_REQ.2 Derived security requirements**

971 Dependencies: APE\_OBJ.2 Security objectives

972 APE\_ECD.1 Extended components definition

973 **7.7.4.1 Developer action elements**

974 **7.7.4.1.1 APE\_REQ.2.1D**

975 The developer shall provide a statement of security requirements.

976 **7.7.4.1.2 APE\_REQ.2.2D**

977 The developer shall provide a security requirements rationale.



978 **7.7.4.2 Content and presentation elements**

979 **7.7.4.2.1 APE\_REQ.2.1C**

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

981 **7.7.4.2.2 APE\_REQ.2.2C**

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

984 **7.7.4.2.3 APE\_REQ.2.3C**

985 **Editor's Note:**

986 To be discussed whether the "natural language" should remain or not.

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

991 **7.7.4.2.4 APE\_REQ.2.4C**

992 **The statement of security requirements shall identify all operations on the security**  
993 **requirements.**

994 **7.7.4.2.5 APE\_REQ.2.5C**

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

996 **7.7.4.2.6 APE\_REQ.2.6C**

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

999 **7.7.4.2.7 APE\_REQ.2.7C**

1000 **The security requirements rationale shall trace each SFR back to the security objectives for the**  
1001 **TOE and OSPs enforced by that SFR.**

1002 **7.7.4.2.8 APE\_REQ.2.8C**

1003 **The security requirements rationale shall demonstrate that the SFRs meet all security**  
1004 **objectives for the TOE.**

1005 **7.7.4.2.9 APE\_REQ.2.9C**

1006 **The security requirements rationale shall demonstrate that the SFRs enforce all OSPs.**

1007 **7.7.4.2.10 APE\_REQ.2.10C**

1008 **The security requirements rationale shall explain why the SARs were chosen.**

1009 **7.7.4.2.11 APE\_REQ.2.11C**

1010 **The statement of security requirements shall be internally consistent.**

1011 **7.7.4.3 Evaluator action elements**

1012 **7.7.4.3.1 APE\_REQ.2.1E**

1013 The evaluator shall confirm that the information provided meets all requirements for content and  
1014 presentation of evidence.

1015 **8 Class ACE: Protection Profile Configuration evaluation**

1016 **8.1 Introduction**

1017 Evaluating a PP-Configuration is required to demonstrate that the PP-Configuration is sound and  
1018 consistent. These properties are necessary for the PP-Configuration to be suitable for use as the basis  
1019 for writing an ST.

1020 The class ACE is defined for the evaluation of a PP-Configuration composed of PPs and PP-Modules<sup>1</sup>.  
1021 The evaluation of PPs is addressed in Class APE. The class ACE defines the requirements for

1022 • Evaluating the PP-Modules in the framework of their base PPs/PP-Modules (components  
1023 ACE\_INT.1, ACE\_CCL.1, ACE\_SPD.1, ACE\_OBJ.1 or ACE\_OBJ.2, ACE\_REQ.1 or ACE\_REQ.2, and  
1024 ACE\_MCO.1.

1025 • Evaluating the consistency of the combination of all the PPs and PP-Modules that belong to the  
1026 PP-Configuration (component ACE\_CCO.1).

1027 Clause 8 should be used in conjunction with **Annexe C of ISO/IEC 15408-1**.

1028

1029 **Figure 5: ACE: Protection Profile Configuration evaluation class decomposition**

1030 **8.2 PP-Module introduction (ACE\_INT)**

1031 **8.2.1 Objectives**

1032 The objective of this family is to describe the TOE in a narrative way.

1033 The evaluation of the PP-Module introduction is required to demonstrate that the PP-Module is  
1034 correctly identified, and that the PP-Module reference and TOE overview are consistent with each  
1035 other.

1036 **8.2.2 ACE\_INT.1 PP-Module introduction**

1037 Dependencies: No dependencies.

---

<sup>1</sup> Two PP-Modules could define each other in their basis, which means that a PP-Configuration that contains one of them also contains the other.

- 1038 **8.2.2.1 Developer action elements**
- 1039 **8.2.2.1.1 ACE\_INT.1.1D**
- 1040 **The developer shall provide a PP-Module introduction.**
- 1041 **8.2.2.2 Content and presentation elements**
- 1042 **8.2.2.2.1 ACE\_INT.1.1C**
- 1043 **The PP-Module introduction shall contain a PP-Module reference, a base PPs/PP-Modules**  
 1044 **identification and a TOE overview.**
- 1045 **8.2.2.2.2 ACE\_INT.1.2C**
- 1046 **The PP-Module reference shall uniquely identify the PP-Module.**
- 1047 **8.2.2.2.3 ACE\_INT.1.3C**
- 1048 **The base PPs/PP-Modules identification shall consist of a list or potentially many alternative**  
 1049 **lists of references that uniquely identify the base PPs/PP-Modules on which the PP-Module**  
 1050 **depends.**
- 1051 **8.2.2.2.4 ACE\_INT.1.4C**
- 1052 **The base PPs/PP-Modules identification shall describe the dependency structure of the base**  
 1053 **PPs/PP-Modules.**
- 1054 **8.2.2.2.5 ACE\_INT.1.5C**
- 1055 **The PP-Module introduction shall contain as many TOE overviews as alternative sets of base**  
 1056 **PPs/PP-Modules.**
- 1057 **8.2.2.2.6 ACE\_INT.1.6C**
- 1058 **The TOE overview shall summarise the usage and major security features of the TOE.**
- 1059 **8.2.2.2.7 ACE\_INT.1.7C**
- 1060 **8.2.2.2.8 The TOE overview shall identify the TOE type.**
- 1061 **8.2.2.2.9 ACE\_INT.1.8C**
- 1062 **The TOE overview shall identify any non-TOE hardware/software/firmware available to the**  
 1063 **TOE.**
- 1064 **8.2.2.2.10 ACE\_INT.1.9C**
- 1065 **The TOE overview shall describe the differences of the TOE with regard to the TOEs defined in**  
 1066 **the base PPs/ PP-Modules.**

1067 **8.2.2.3 Evaluator action elements**

1068 **8.2.2.3.1 ACE\_INT.1.1E**

1069 **The evaluator shall confirm that the information provided meets all requirements for content**  
1070 **and presentation of evidence.**

1071 **8.3 PP-Module conformance claims (ACE\_CCL)**

1072 **8.3.1 Objectives**

1073 The objective of this family is to determine the validity of the conformance claim and conformance  
1074 statement. A PP-Module cannot claim conformance to any PP, PP-Configuration, or another PP-Module.

1075 **8.3.2 ACE\_CCL.1 PP-Module conformance claims**

1076 Dependencies: ACE\_INT.1 PP-Module introduction

1077 ACE\_ECD.1 PP-Module extended components definition

1078 ACE\_REQ.1 PP-Module stated security requirements or ACE\_REQ.2 PP-Module  
1079 security requirements

1080 **8.3.2.1.1 ACE\_CCL.1.1D**

1081 **The developer shall provide a conformance claim.**

1082 **8.3.2.1.2 ACE\_CCL.1.2D**

1083 **The developer shall provide a conformance statement.**

1084 **8.3.2.2 Content and presentation elements**

1085 **8.3.2.2.1 ACE\_CCL.1.1C**

1086 **The conformance claim shall contain an ISO/IEC 15408 series conformance claim that identifies**  
1087 **the ISO/IEC 15408-1 edition to which the PP-Module claims conformance.**

1088 **8.3.2.2.2 ACE\_CCL.1.2C**

1089 **The ISO/IEC 15408 series conformance claim shall describe the conformance of the PP-Module**  
1090 **to ISO/IEC 15408-2 as either ISO/IEC 15408-2 conformant or ISO/IEC 15408-2 extended.**

1091 **8.3.2.2.3 ACE\_CCL.1.3C**

1092 **The conformance statement shall describe the conformance type required of any ST to the PP-**  
1093 **Module (as part of a PP-Configuration) as one of exact, strict, or demonstrable.**

1094 **8.3.2.2.4 ACE\_CCL.1.4C**

1095 **The ISO/IEC 15408 series conformance claim shall describe the conformance of the PP-Module**  
1096 **to this document as either "ISO/IEC 15408-3 conformant" or ISO/IEC 15408-3 extended."**

- 1097 **8.3.2.2.5 ACE\_CCL.1.5C**
- 1098 **The ISO/IEC 15408 series conformance claim shall be consistent with the extended**  
 1099 **components definition.**
- 1100 **8.3.2.2.6 ACE\_CCL.1.6C**
- 1101 **The conformance claim shall identify all functional packages to which the PP-Module claims**  
 1102 **conformance.**
- 1103 **8.3.2.2.7 ACE\_CCL.1.7C**
- 1104 **The conformance claim shall describe any conformance of the PP-Module to a functional**  
 1105 **package as either package-name-conformant, package-augmented or package-name-tailored.**
- 1106 **8.3.2.2.8 ACE\_CCL.1.8C**
- 1107 **For strict and demonstrable conformance, the conformance claim shall identify all assurance**  
 1108 **packages to which the PP-Module claims conformance.**
- 1109 **8.3.2.2.9 ACE\_CCL.1.9C**
- 1110 **For strict and demonstrable conformance, the conformance claim shall describe any**  
 1111 **conformance of the PP-Module to an assurance package as either package-name-conformant or**  
 1112 **package-name-augmented.**
- 1113 **8.3.2.2.10 ACE\_CCL.1.10C**
- 1114 **For exact conformance, the PP-Module's conformance statement shall identify the set of PPs**  
 1115 **and PP-Modules to which, in combination with the PP-Module under evaluation, exact**  
 1116 **conformance is allowed to be claimed.**
- 1117 **8.3.2.2.11 ACE\_CCL.1.11C**
- 1118 **The conformance statement may identify the set of ISO/IEC 18045-derived Evaluation Methods**  
 1119 **and Evaluation Activities that shall be used with the PP-Module under evaluation. This list shall**  
 1120 **contain any Evaluation Methods and Evaluation Activities that are specified in the PP-Module**  
 1121 **but also any Evaluation Activities and Evaluation Methods specified in the base PPs/ PP-**  
 1122 **Modules and/or in the packages (if any) for which conformance is being claimed by the PP-**  
 1123 **Module under evaluation.**
- 1124 **8.3.2.3 Evaluator action elements**
- 1125 **8.3.2.3.1 ACE\_CCL.1.1E**
- 1126 **The evaluator shall confirm that the information provided meets all requirements for content**  
 1127 **and presentation of evidence.**
- 1128 **8.4 PP-Module security problem definition (ACE\_SPD)**
- 1129 **8.4.1 Objectives**
- 1130 **This part of the PP-Module defines the security problem to be addressed by the TOE and the**  
 1131 **operational environment of the TOE.**

1132 Evaluation of the security problem definition is required to demonstrate that the security problem  
1133 intended to be addressed by the TOE and its operational environment, is clearly defined.

#### 1134 **8.4.2 ACE\_SPD.1 PP-Module Security problem definition**

1135 Dependencies: No dependencies.

##### 1136 **8.4.2.1 Developer action elements**

###### 1137 **8.4.2.1.1 ACE\_SPD.1.1D**

1138 **The developer shall provide a security problem definition.**

##### 1139 **8.4.2.2 Content and presentation elements**

###### 1140 **8.4.2.2.1 ACE\_SPD.1.1C**

1141 **The security problem definition shall describe the threats.**

###### 1142 **8.4.2.2.2 ACE\_SPD.1.2C**

1143 **All threats shall be described in terms of a threat agent, an asset, and an adverse action.**

###### 1144 **8.4.2.2.3 ACE\_SPD.1.3C**

1145 **The security problem definition shall describe the OSPs.**

###### 1146 **8.4.2.2.4 ACE\_SPD.1.4C**

1147 **The security problem definition shall describe the assumptions about the operational**  
1148 **environment of the TOE.**

##### 1149 **8.4.2.3 Evaluator action elements**

###### 1150 **8.4.2.3.1 ACE\_SPD.1.1E**

1151 **The evaluator shall confirm that the information provided meets all requirements for content**  
1152 **and presentation of evidence.**

#### 1153 **8.5 PP-Module security objectives (ACE\_OBJ)**

##### 1154 **8.5.1 Objectives**

1155 The security objectives are a concise statement of the intended response to the security problem  
1156 defined through the Security problem definition (APE\_SPD) family.

1157 Evaluation of the security objectives is required to demonstrate that the security objectives adequately  
1158 and completely address the security problem definition and that the division of this problem between  
1159 the TOE and its operational environment is clearly defined.

##### 1160 **8.5.2 Component levelling**

1161 The components in this family are levelled on whether they prescribe only security objectives for the  
1162 operational environment (see ACE\_OBJ.1), or also security objectives for the TOE (see ACE\_OBJ.2).

- 1163 **8.5.3 ACE\_OBJ.1 Direct Rationale PP-Module security objectives**
- 1164 Dependencies: No dependencies.
- 1165 **8.5.3.1 Developer action elements**
- 1166 **8.5.3.1.1 ACE\_OBJ.1.1D**
- 1167 **The developer shall provide a statement of security objectives for the PP-Module.**
- 1168 **8.5.3.2 Content and presentation elements**
- 1169 **8.5.3.2.1 ACE\_OBJ.1.1C**
- 1170 **The statement of security objectives shall describe the security objectives for the operational**
- 1171 **environment.**
- 1172 **8.5.3.3 Evaluator action elements**
- 1173 **8.5.3.3.1 ACE\_OBJ.1.1E**
- 1174 **The evaluator shall confirm that the information provided meets all requirements for content**
- 1175 **and presentation of evidence.**
- 1176 **8.5.4 ACE\_OBJ.2 PP-Module Security objectives**
- 1177 Dependencies: ACE\_SPD.1 PP-Module security problem definition.
- 1178 **8.5.4.1 Developer action elements**
- 1179 **8.5.4.1.1 ACE\_OBJ.2.1D**
- 1180 **The developer shall provide a statement of security objectives for the PP-Module.**
- 1181 **8.5.4.1.2 ACE\_OBJ.2.2D**
- 1182 **The developer shall provide a security objectives rationale for the PP-Module.**
- 1183 **8.5.4.2 Content and presentation elements**
- 1184 **8.5.4.2.1 ACE\_OBJ.2.1C**
- 1185 **The statement of security objectives shall describe the security objectives for the TOE and the**
- 1186 **security objectives for the operational environment.**
- 1187 **8.5.4.2.2 ACE\_OBJ.2.2C**
- 1188 **The security objectives rationale shall trace each security objective for the TOE back to threats**
- 1189 **countered by that security objective and OSPs enforced by that security objective.**
- 1190 **8.5.4.2.3 ACE\_OBJ.2.3C**
- 1191 **The security objectives rationale shall trace each security objective for the operational**
- 1192 **environment back to threats countered by that security objective, OSPs enforced by that**
- 1193 **security objective, and assumptions upheld by that security objective.**

- 1194 **8.5.4.2.4 ACE\_OBJ.2.4C**
- 1195 **The security objectives rationale shall demonstrate that the security objectives counter all**  
1196 **threats.**
- 1197 **8.5.4.2.5 ACE\_OBJ.2.5C**
- 1198 **The security objectives rationale shall demonstrate that the security objectives enforce all**  
1199 **OSPs.**
- 1200 **8.5.4.2.6 ACE\_OBJ.2.6C**
- 1201 **The security objectives rationale shall demonstrate that the security objectives for the**  
1202 **operational environment uphold all assumptions.**
- 1203 **8.5.4.3 Evaluator action elements**
- 1204 **8.5.4.3.1 ACE\_OBJ.2.1E**
- 1205 **The evaluator shall confirm that the information provided meets all requirements for content**  
1206 **and presentation of evidence.**
- 1207 **8.6 PP-Module extended components definition (ACE\_ECD)**
- 1208 **8.6.1 Objectives**
- 1209 Extended security functional requirements are requirements that are not based on components from  
1210 **ISO/IEC 15408-2**, but are based on extended components: components defined by the PP-Module  
1211 author.
- 1212 Evaluation of the definition of extended functional components is necessary to determine that they are  
1213 clear and unambiguous, and that they are necessary, i.e. they may not be clearly expressed using  
1214 existing **ISO/IEC 15408-2** components.
- 1215 **8.6.2 ACE\_ECD.1 PP-Module extended components definition**
- 1216 Dependencies: No dependencies.
- 1217 **8.6.2.1 Developer action elements**
- 1218 **8.6.2.1.1 ACE\_ECD.1.1D**
- 1219 **The developer shall provide a statement of security requirements for the PP-Module.**
- 1220 **8.6.2.1.2 ACE\_ECD.1.2D**
- 1221 **The developer shall provide an extended components definition for the PP-Module.**
- 1222 **8.6.2.2 Content and presentation elements**
- 1223 **8.6.2.2.1 ACE\_ECD.1.1C**
- 1224 **The statement of security requirements shall identify all the extended security requirements.**



- 1225 **8.6.2.2.2 ACE\_ECD.1.2C**
- 1226 **The extended components definition shall define an extended component for each extended**  
 1227 **security requirement.**
- 1228 **8.6.2.2.3 ACE\_ECD.1.3C**
- 1229 **The extended components definition shall describe how each extended component is related to**  
 1230 **the existing ISO/IEC 15408 series components, families, and classes.**
- 1231 **8.6.2.2.4 ACE\_ECD.1.4C**
- 1232 **The extended components definition shall use the existing ISO/IEC 15408 series components,**  
 1233 **families, classes, and methodology as a model for presentation.**
- 1234 **8.6.2.2.5 ACE\_ECD.1.5C**
- 1235 **8.6.2.2.6 The extended components shall consist of measurable and objective elements such**  
 1236 **that conformance or nonconformance to these elements may be demonstrated.**
- 1237 **8.6.2.3 Evaluator action elements**
- 1238 **8.6.2.3.1 ACE\_ECD.1.1E**
- 1239 **The evaluator shall confirm that the information provided meets all requirements for content**  
 1240 **and presentation of evidence.**
- 1241 **8.6.2.3.2 ACE\_ECD.1.2E**
- 1242 **The evaluator shall confirm that no extended component may be clearly expressed using**  
 1243 **existing components.**
- 1244 **8.7 PP-Module security requirements (ACE\_REQ)**
- 1245 **8.7.1 Objectives**
- 1246 **The SFRs form a clear, unambiguous and well-defined description of the expected security behaviour**  
 1247 **of the TOE. The SARs form a clear, unambiguous and well-defined description of the expected activities**  
 1248 **that will be undertaken to gain assurance in the TOE.**
- 1249 **Evaluation of the security requirements is required to ensure that they are clear, unambiguous and**  
 1250 **well-defined.**
- 1251 **8.7.2 Component levelling**
- 1252 **The components in this family are levelled on whether they are stated as is (see ACE\_REQ.1), or**  
 1253 **whether the SFRs are derived from security objectives for the TOE (see ACE\_REQ.2.).**
- 1254 **8.7.3 ACE\_REQ.1 Direct rationale PP-Module security requirements**
- 1255 **Dependencies:** [APE\\_ECD.1 Extended components definition](#)~~[APE\\_ECD.1 Extended components definition](#)~~  
 1256 [ACE\\_SPD.1 PP-Module security problem definition](#)

- 1257 **8.7.3.1 Developer action elements**
- 1258 **8.7.3.1.1 ACE\_REQ.1.1D**
- 1259 **The developer shall provide a statement of security requirements for the PP-Module.**
- 1260 **8.7.3.1.2 ACE\_REQ.1.2D**
- 1261 **The developer shall provide a security requirements rationale for the PP-Module.**
- 1262 **8.7.3.2 Content and presentation elements**
- 1263 **8.7.3.2.1 ACE\_REQ.1.1C**
- 1264 **The statement of security requirements shall describe the SFRs. For the strict and**  
 1265 **demonstrable conformance cases, the statement of security requirements shall also describe**  
 1266 **and the SARs.<sup>2</sup>**
- 1267 **8.7.3.2.2 ACE\_REQ.1.2C**
- 1268 **All subjects, objects, operations, security attributes, external entities and other terms that are**  
 1269 **used in the SFRs and the SARs shall be defined.**
- 1270 **8.7.3.2.3 ACE\_REQ.1.3C**
- 1271 **The statement of security requirements shall include a natural language description, part of**  
 1272 **which describes how the SFRs combine together to provide security functionality in terms of**  
 1273 **the architecture that is observable to Administrators and other users, or in terms of internal**  
 1274 **features or properties.**
- 1275 **8.7.3.2.4 ACE\_REQ.1.4C**
- 1276 **The statement of security requirements shall identify all operations on the security**  
 1277 **requirements.**
- 1278 **8.7.3.2.5 ACE\_REQ.1.5C**
- 1279 **All operations shall be performed correctly.**
- 1280 **8.7.3.2.6 ACE\_REQ.1.6C**
- 1281 **Each dependency of the security requirements shall either be satisfied, or the security**  
 1282 **requirements rationale shall justify the dependency not being satisfied.**
- 1283 **8.7.3.2.7 ACE\_REQ.1.7C**
- 1284 **The security requirements rationale shall trace each SFR back to the threats countered by that**  
 1285 **SFR and the OSPs enforced by that SFR.**

---

<sup>2</sup> In exact conformance PP-Modules SARs are inherited from base PPs

- 1286 **8.7.3.2.8 ACE\_REQ.1.8C**
- 1287 **The security requirements rationale shall trace each security objective for the operational**  
 1288 **environment back to the threats countered by that security objective, the OSPs enforced by that**  
 1289 **security objective, and the assumptions upheld by that security objective.**
- 1290 **8.7.3.2.9 ACE\_REQ.1.9C**
- 1291 **The security requirements rationale shall demonstrate that the SFRs (in conjunction with the**  
 1292 **security objectives for the environment) counter all the threats for the TOE.**
- 1293 **8.7.3.2.10 ACE\_REQ.1.10C**
- 1294 **The security requirements rationale shall demonstrate that the SFRs (in conjunction with the**  
 1295 **security objectives for the environment) enforce all the OSPs for the TOE.**
- 1296 **8.7.3.2.11 ACE\_REQ.1.11C**
- 1297 **The security requirements rationale shall demonstrate that the security objectives for the**  
 1298 **operational environment uphold all assumptions.**
- 1299 **8.7.3.2.12 ACE\_REQ.1.12C**
- 1300 The statement of security requirements shall be internally consistent.
- 1301 **8.7.3.3 Evaluator action elements**
- 1302 **8.7.3.3.1 ACE\_REQ.1.1E**
- 1303 **The evaluator shall confirm that the information provided meets all requirements for content**  
 1304 **and presentation of evidence.**
- 1305 **8.7.4 ACE\_REQ.2 PP-Module derived security requirements**
- 1306 Dependencies: ACE\_ECD.1 PP-Module extended components definition
- 1307 ACE\_OBJ.2 PP-Module Security objectives
- 1308 **8.7.4.1 Developer action elements**
- 1309 **8.7.4.1.1 ACE\_REQ.2.1D**
- 1310 **The developer shall provide a statement of security requirements for the PP-Module.**
- 1311 **8.7.4.1.2 ACE\_REQ.2.2D**
- 1312 **The developer shall provide a security requirement rationale for the PP-Module.**

1313 **8.7.4.2 Content and presentation elements**

1314 **8.7.4.2.1 ACE\_REQ.2.1C**

1315 **The statement of security requirements shall describe the SFRs and the SARs. For the strict and**  
 1316 **demonstrable conformance cases, the statement of security requirements shall also describe**  
 1317 **the SARs<sup>3</sup>.**

1318 **8.7.4.2.2 ACE\_REQ.2.2C**

1319 **All subjects, objects, operations, security attributes, external entities and other terms that are**  
 1320 **used in the SFRs and the SARs shall be defined.**

1321 **8.7.4.2.3 ACE\_REQ.2.3C**

1322 **The statement of security requirements shall identify all operations on the security**  
 1323 **requirements.**

1324 **8.7.4.2.4 ACE\_REQ.2.4C**

1325 **All operations shall be performed correctly.**

1326 **8.7.4.2.5 ACE\_REQ.2.5C**

1327 **Each dependency of the security requirements shall either be satisfied, or the security**  
 1328 **requirements rationale shall justify the dependency not being satisfied.**

1329 **8.7.4.2.6 ACE\_REQ.2.6C**

1330 **The security requirements rationale shall trace each SFR back to the security objectives for the**  
 1331 **TOE and OSPs enforced by that SFR.**

1332 **8.7.4.2.7 ACE\_REQ.2.7C**

1333 **The security requirements rationale shall demonstrate that the SFRs meet all security**  
 1334 **objectives for the TOE.**

1335 **8.7.4.2.8 ACE\_REQ.2.8C**

1336 **The security requirements rationale shall demonstrate that the SFRs (in conjunction with the**  
 1337 **security objectives for the environment) enforce all OSPs.**

1338 **8.7.4.2.9 ACE\_REQ.2.9C**

1339 **The security requirements rationale shall explain why the SARs were chosen.**

1340 **8.7.4.2.10 ACE\_REQ.2.10C**

1341 **The statement of security requirements shall be internally consistent.**

---

<sup>3</sup> In exact conformance PP-Modules SARs are inherited from base PPs.

1342 **8.7.4.3 Evaluator action elements**

1343 **8.7.4.3.1 ACE\_REQ.2.1E**

1344 **The evaluator shall confirm that the information provided meets all requirements for content**  
1345 **and presentation of evidence.**

1346 **8.8 PP-Module consistency (ACE\_MCO)**

1347 **8.8.1 Objectives**

1348 The objective of this family is to determine the consistency of the PP-Module and to state the  
1349 correspondence between the PP-Module and its base PPs/PP-Modules.

1350 **8.8.2 ACE\_MCO.1 PP-Module consistency**

1351 Dependencies: ACE\_INT.1 PP-Module introduction

1352 ACE\_SPD.1 PP-Module Security problem definition

1353 ACE\_OBJ.1 Direct Rationale PP-Module Security objectives for the environment or  
1354 ACE\_OBJ.2 PP-Module Security objectives

1355 ACE\_REQ.1 PP-Module stated security requirements or ACE\_REQ.2 PP-Module  
1356 security requirements

1357 **8.8.2.1 Developer action elements**

1358 **8.8.2.1.1 ACE\_MCO.1.1D**

1359 **The developer shall provide a consistency rationale of the PP-Module for each of the alternative**  
1360 **sets of base PPs/ PP-Modules identified in the PP-Module introduction.**

1361 **8.8.2.1.2 ACE\_MCO.1.2D**

1362 **For the strict and demonstrable conformance cases, the developer shall provide an assurance**  
1363 **rationale of the PP-Module for each of the alternative sets of base PPs/PP-Modules identified in**  
1364 **the PP-Module introduction.**

1365 **8.8.2.2 Content and presentation elements**

1366 **8.8.2.2.1 ACE\_MCO.1.1C**

1367 **The consistency rationale shall demonstrate that the TOE type of the PP-Module and the TOE**  
1368 **types of its base PPs/ PP-Modules are consistent.**

1369 **8.8.2.2.2 ACE\_MCO.1.2C**

1370 **The consistency rationale shall identify the assets of the PP-Module's SPD that also belong to**  
1371 **some of its base PPs/PP-Modules and amongst them those for which the PP-Module and the**  
1372 **base PPs/ PP-Modules define different security problems.**

1373 **8.8.2.2.3 ACE\_MCO.1.3C**

1374 The consistency rationale shall demonstrate that the union of the security problem definition  
1375 of the PP-Module and the security problem definition of its base PPs/ PP-Modules are  
1376 consistent.

1377 **8.8.2.2.4 ACE\_MCO.1.4C**

1378 The consistency rationale shall demonstrate that the union of the security objectives of the PP-  
1379 Module and the security objectives of its base PPs/ PP-Modules are consistent.

1380 **8.8.2.2.5 ACE\_MCO.1.5C**

1381 The consistency rationale shall demonstrate that the union of the security functional  
1382 requirements of the PP-Module and the security functional requirements of its base PPs/ PP-  
1383 Modules are consistent.

1384 **8.8.2.2.6 ACE\_MCO.1.6C**

1385 For the strict and demonstrable conformance cases, the assurance rationale shall demonstrate  
1386 the internal consistency of the set of security assurance requirements of the PP-Module with  
1387 regard to its security problem definition.

1388 **8.8.2.2.7 ACE\_MCO.1.7C**

1389 For the strict and demonstrable conformance cases, the assurance rationale shall demonstrate  
1390 the consistency of the set of security assurance requirements of the PP-Module with regard to  
1391 the security assurance requirements of the base PPs/PP-Modules.

1392 **8.8.2.3 Evaluator action elements**

1393 **8.8.2.3.1 ACE\_MCO.1.1E**

1394 The evaluator shall confirm that the information provided meets all requirements for content  
1395 and presentation of evidence. If the PP-Module specifies alternative sets of base PPs and PP-  
1396 Modules, the evaluator shall perform this action for each consistency rationale.

1397 **8.9 PP-Configuration consistency (ACE\_CCO)**

1398 **8.9.1 Objectives**

1399 The objective of this family is to determine the well-formedness and the consistency of the PP-  
1400 Configuration.

1401 **8.9.2 ACE\_CCO.1 PP-Configuration consistency**

1402 Dependencies: ACE\_INT.1 PP-Module introduction

1403 ACE\_CCL.1 PP-Module conformance claims

1404 ACE\_SPD.1 PP-Module security problem definition

1405 ACE\_OBJ.1 Direct Rationale PP-Module security objectives for the environment or  
1406 ACE\_OBJ.2 PP-Module Security objectives

1407	ACE_ECD.1 PP-Module extended component definition
1408	ACE_REQ.1 PP-Module stated security requirements or ACE_REQ.2 PP-Module
1409	security requirements
1410	ACE_MCO.1 PP-Module consistency
1411	APE_* (all APE comments)
1412	<b>8.9.2.1 Developer action elements</b>
1413	<b>8.9.2.1.1 ACE_CCO.1.1D</b>
1414	<b>The developer shall provide the reference of the PP-Configuration.</b>
1415	<b>8.9.2.1.2 ACE_CCO.1.2D</b>
1416	<b>The developer shall provide a components statement.</b>
1417	<b>8.9.2.1.3 ACE_CCO.1.3D</b>
1418	<b>The developer shall provide a TOE overview.</b>
1419	<b>8.9.2.1.4 ACE_CCO.1.4D</b>
1420	<b>The developer shall provide a conformance claim.</b>
1421	<b>8.9.2.1.5 ACE_CCO.1.5D</b>
1422	<b>The developer shall provide a conformance statement within the conformance claim.</b>
1423	<b>8.9.2.1.6 ACE_CCO.1.6D</b>
1424	<b>The developer shall provide a consistency rationale.</b>
1425	<b>8.9.2.1.7 ACE_CCO.1.7D</b>
1426	<b>The developer shall provide a SAR statement.</b>
1427	<b>8.9.2.1.8 ACE_CCO.1.8D</b>
1428	<b>The developer shall provide the set of evaluation methods and/or activities that are applicable</b>
1429	<b>to the PP-Configuration.</b>
1430	<b>8.9.2.2 Content and presentation elements</b>
1431	<b>8.9.2.2.1 ACE_CCO.1.1C</b>
1432	<b>The PP-Configuration reference shall uniquely identify the PP-Configuration.</b>
1433	<b>8.9.2.2.2 ACE_CCO.1.2C</b>
1434	<b>The PP-Configuration components statement shall uniquely identify the PPs and PP-Modules</b>
1435	<b>that compose the PP-Configuration.</b>

1436 **8.9.2.2.3 ACE\_CCO.1.3C**

1437 **For each PP-Module identified in the PP-Configuration components statement, the components**  
 1438 **statement shall include the base PPs/PP-Modules required by the identified PP-Module. If the**  
 1439 **PP-Module specifies alternative sets of base PPs/PP-Modules, only one of these sets shall be**  
 1440 **referred to in the PP-Configuration.**

1441 **8.9.2.2.4 ACE\_CCO.1.4C**

1442 **For a multi-assurance PP-Configuration, the components statement shall describe the**  
 1443 **organisation of the TSF in terms of the sub-TSFs defined in the PPs and PP-Modules defined in**  
 1444 **the PP-Configuration.**

1445 **8.9.2.3 ACE\_CCO.1.5C**

1446 **The TOE overview shall identify the TOE type.**

1447 **8.9.2.4 ACE\_CCO.1.6C**

1448 **The TOE overview shall describe the usage and major security features of the TOE.**

1449 **8.9.2.5 ACE\_CCO.1.7C**

1450 **The TOE overview shall identify any non-TOE hardware/software/firmware available to the**  
 1451 **TOE.**

1452 **8.9.2.6 ACE\_CCO.1.8C**

1453 **The conformance claim shall contain an ISO/IEC 15408 series conformance claim that identifies**  
 1454 **the ISO/IEC 15408-1 edition(s) to which the PPs and PP-Modules that compose the PP-**  
 1455 **Configuration claim conformance.**

1456 **8.9.2.7 ACE\_CCO.1.9C**

1457 **The ISO/IEC 15408 series conformance claim shall describe the conformance of the PP-**  
 1458 **Configuration to ISO/IEC 15408-2 as either ISO/IEC 15408-2 conformant or ISO/IEC 15408-2**  
 1459 **extended.**

1460 **8.9.2.8 ACE\_CCO.1.10C**

1461 **The ISO/IEC 15408 series conformance claim shall describe the conformance of the PP-**  
 1462 **Configuration to this document as either “ISO/IEC 15408-3 conformant” or ISO/IEC 15408-3**  
 1463 **extended.”**

1464 **8.9.2.9 ACE\_CCO.1.11C**

1465 **The ISO/IEC 15408 series conformance claim shall be consistent with the composing PPs and**  
 1466 **PP-Modules.**

1467 **8.9.2.10 ACE\_CCO.1.12C**

1468 **The conformance claim of a PP-Configuration shall include an assurance package conformance**  
 1469 **claim consisting of statements describing any conformance of the PP-Configuration to an**  
 1470 **assurance package as either package-name-conformant or package-name-augmented.**



## 1471 8.9.2.11 ACE\_CCO.1.13C

1472 The conformance statement shall specify the required conformance to the PP-Configuration as  
 1473 one of exact, strict, demonstrable, or it shall provide the list of conformance types that are  
 1474 required by each of the PPs and PP-Modules composing the PP-Configuration.

## 1475 8.9.2.12 ACE\_CCO.1.14C

1476 For the exact conformance case, the conformance statement of each PP included in the  
 1477 components statement of the PP-Configuration shall identify the PPs and PP-Modules listed in  
 1478 the PP-Configuration's component statement as being allowed to be used in combination with  
 1479 the PP in a PP-Configuration.

## 1480 8.9.2.13 ACE\_CCO.1.15C

1481 For the exact conformance case, the conformance statement of each PP-Module included in the  
 1482 components statement of the PP-Configuration shall identify the PPs and PP-Modules listed in  
 1483 the PP-Configuration's component statement that are not base PPs/PP-Modules as being  
 1484 allowed to be used in combination with the PP-Module in a PP-Configuration.

## 1485 8.9.2.14 ACE\_CCO.1.16C

1486 The conformance statement of a PP-Configuration may include an Evaluation Methods and Evaluation  
 1487 Activities reference statement that identifies the set of ISO/IEC 18045-derived Evaluation  
 1488 Methods and Evaluation Activities that are applicable to the PP-Configuration under  
 1489 evaluation.

## 1490 8.9.2.15 ACE\_CCO.1.17C

1491 The consistency rationale shall demonstrate that the TOE type defined in the PP-Configuration  
 1492 is consistent with the TOE types defined in the PPs and PP-Modules that belong to the PP-  
 1493 Configuration components statement.

## 1494 8.9.2.16 ACE\_CCO.1.18C

1495 The consistency rationale shall demonstrate that the union of all the SPDs, security objectives  
 1496 and security functional requirements defined in the PPs and PP-Modules of the PP-  
 1497 Configuration components statement is consistent.

## 1498 8.9.2.17 ACE\_CCO.1.19C

1499 For a single-assurance PP-Configuration, the SAR statement shall define the applicable  
 1500 assurance requirements. The SAR statement shall define a single set of SARs for all the  
 1501 components in the PP-Configuration under evaluation. For exact conformance, the set of SARs  
 1502 shall be inherited from the PPs.

## 1503 8.9.2.18 ACE\_CCO.1.20C

1504 For a multi-assurance PP-Configuration, the SAR statement shall define the applicable  
 1505 assurance requirements. The SAR statement shall define the global set of SARs that applies to  
 1506 the entire TOE and the SARs that apply to each sub-TSF.

Kommentiert [OA1]: See Comment US 086

1507 **8.9.2.19 ACE\_CCO.1.21C**

1508 **The SAR statement of a PP-Configuration shall include an assurance rationale that**  
 1509 **demonstrates the consistency of the applicable set of SARs with those defined in the**  
 1510 **components of the PP-Configuration under evaluation. For PP-Configurations using the multi-**  
 1511 **assurance evaluation approach, the assurance rationale shall demonstrate:**

- 1512 • **the consistency of the global set of SARs/assurance package with regard to the threat**  
 1513 **models as defined in the SPDs of the component PPs and PP-Modules, and**
- 1514 • **the consistency of the global set of SARs/assurance package and all the sets of**  
 1515 **SARs/assurance packages for the sub-TSFs (TSF parts) with each other.**

1516 **8.9.2.20 Evaluator action elements**

1517 **8.9.2.20.1 ACE\_CCO.1.1E**

1518 **The evaluator shall confirm that the information provided meets all requirements for content**  
 1519 **and presentation of evidence.**

1520 **8.9.2.20.2 ACE\_CCO.1.2E**

1521 **The evaluator shall check that the PP-Configuration consisting of all the PPs and PP-Modules**  
 1522 **identified in the components list is consistent.**

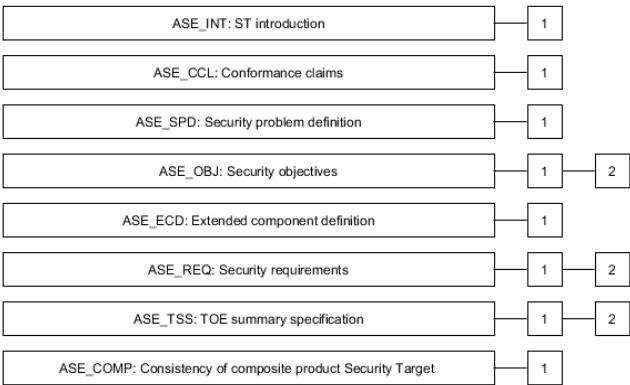
1523 **9 Class ASE: Security Target evaluation**

1524 **9.1 Introduction**

1525 Evaluating an ST is required to demonstrate that the ST is sound and internally consistent, and, if the  
 1526 ST is based on a PP-Configuration, or one or more PPs or packages, that the ST is a correct  
 1527 instantiation of the PP-Configuration, PPs, and packages. These properties are necessary for the ST to  
 1528 be suitable for use as the basis for a TOE evaluation.

1529 Clause 9 should be used in conjunction with Annexes B, C and D in ISO/IEC 15408-1:XXXX, as these  
 1530 annexes clarify the concepts here and provide many examples.

1531 Figure 6 shows the families within this class, and the hierarchy of components within the families.



1532

1533 **Figure 6 — ASE: Security Target evaluation class decomposition**

1534 **9.2 ST introduction (ASE\_INT)**

1535 **9.2.1 Objectives**

1536 The objective of this family is to describe the TOE in a narrative way on three levels of abstraction:  
1537 TOE reference, TOE overview and TOE description.

1538 Evaluation of the ST introduction is required to demonstrate that the ST and the TOE are correctly  
1539 identified, that the TOE is correctly described at three levels of abstraction and that these three  
1540 descriptions are consistent with each other.

1541 **9.2.2 ASE\_INT.1 ST introduction**

1542 Dependencies: No dependencies.

1543 **9.2.2.1 Developer action elements**

1544 **9.2.2.1.1 ASE\_INT.1.1D**

1545 **The developer shall provide an ST introduction.**

1546 **9.2.2.2 Content and presentation elements**

1547 **9.2.2.2.1 ASE\_INT.1.1C**

1548 **The ST introduction shall contain an ST reference, a TOE reference, a TOE overview and a TOE  
1549 description.**

1550 **9.2.2.2.2 ASE\_INT.1.2C**

1551 **The ST reference shall uniquely identify the ST.**

- 1552 **9.2.2.2.3 ASE\_INT.1.3C**
- 1553 **The TOE reference shall uniquely identify the TOE.**
- 1554 **9.2.2.2.4 ASE\_INT.1.4C**
- 1555 **The TOE overview shall summarise the usage and major security features of the TOE.**
- 1556 **9.2.2.2.5 ASE\_INT.1.5C**
- 1557 **The TOE overview shall identify the TOE type.**
- 1558 **9.2.2.2.6 ASE\_INT.1.6C**
- 1559 **The TOE overview shall identify any non-TOE hardware/software/firmware required by the**  
 1560 **TOE.**
- 1561 **9.2.2.2.7 ASE\_INT.1.7C**
- 1562 **For a multi-assurance ST, the TOE overview shall describe the TSF organization in terms of the**  
 1563 **sub-TSFs defined in the PP-Configuration the ST claims conformance to.**
- 1564 **9.2.2.2.8 ASE\_INT.1.8C**
- 1565 **The TOE description shall describe the physical scope of the TOE.**
- 1566 **9.2.2.2.9 ASE\_INT.1.9C**
- 1567 **The TOE description shall describe the logical scope of the TOE.**
- 1568 **9.2.2.3 Evaluator action elements**
- 1569 **9.2.2.3.1 ASE\_INT.1.1E**
- 1570 **The evaluator shall confirm that the information provided meets all requirements for content**  
 1571 **and presentation of evidence.**
- 1572 **9.2.2.3.2 ASE\_INT.1.2E**
- 1573 **The evaluator shall confirm that the TOE reference, the TOE overview, and the TOE description**  
 1574 **are consistent with each other.**
- 1575 **9.3 Conformance claims (ASE\_CCL)**
- 1576 **9.3.1 Objectives**
- 1577 **The objective of this family is to determine the validity of the conformance claim. In addition, this**  
 1578 **family specifies how STs are to claim conformance with the PP or PP-Configuration.**
- 1579 **9.3.2 ASE\_CCL.1 Conformance claims**
- 1580 Dependencies: ASE\_INT.1 ST introduction
- 1581 ASE\_ECD.1 Extended components definition
- 1582 [ASE\\_REQ.1 Direct Rationale Stated security requirements](#) [ASE\\_REQ.1 Stated security requirements](#)

1583 **9.3.2.1 Developer action elements**

1584 **9.3.2.1.1 ASE\_CCL.1.1D**

1585 **The developer shall provide a conformance claim.**

1586 **9.3.2.1.2 ASE\_CCL.1.2D**

1587 **The developer shall provide a conformance claim rationale.**

1588 **9.3.2.2 Content and presentation elements**

1589 **9.3.2.2.1 ASE\_CCL.1.1C**

1590 **The conformance claim shall contain an ISO/IEC 15408 series conformance claim that identifies**  
1591 **the edition of ISO/IEC 15408 to which the ST and the TOE claim conformance.**

1592 **9.3.2.2.2 ASE\_CCL.1.2C**

1593 **The ISO/IEC 15408 series conformance claim shall describe the conformance of the ST to**  
1594 **ISO/IEC 15408-2 as either ISO/IEC 15408-2 conformant or ISO/IEC 15408-2 extended.**

1595 **9.3.2.2.3 ASE\_CCL.1.3C**

1596 **The ISO/IEC 15408 series conformance claim shall describe the conformance of the ST to this**  
1597 **document as either "ISO/IEC 15408-3 conformant" or ISO/IEC 15408-3 extended."**

1598 **9.3.2.2.4 ASE\_CCL.1.4C**

1599 **The ISO/IEC 15408 series conformance claim shall be consistent with the extended**  
1600 **components definition.**

1601 **9.3.2.2.5 ASE\_CCL.1.5C**

1602 **The conformance claim shall identify a PP-Configuration, or all PPs and security requirement**  
1603 **packages to which the ST claims conformance.**

1604 **9.3.2.2.6 ASE\_CCL.1.6C**

1605 **The conformance claim shall describe any conformance of the ST to a package as either**  
1606 **package-conformant or package-augmented.**

1607 **9.3.2.2.7 ASE\_CCL.1.7C**

1608 **The conformance claim shall describe any conformance of the ST to a PP as PP-Conformant.**

1609 **9.3.2.2.8 ASE\_CCL.1.8C**

1610 **The conformance claim shall include any conformance of the ST to a PP-Configuration. In the**  
1611 **exact conformance case and in the multi-assurance case, the ST shall claim conformance to only**  
1612 **one PP-Configuration and no other PP or PP-Configuration. In the multi-assurance case the PP-**  
1613 **Configuration to which the ST claims conformance shall be a multi-assurance PP-Configuration.**

1614 9.3.2.2.9 ASE\_CCL.1.9C

1615 The conformance claim rationale shall demonstrate that the TOE type is consistent with the  
1616 TOE type in the PP-Configuration or PPs for which conformance is being claimed.

1617 9.3.2.2.10 ASE\_CCL.1.10C

1618 The conformance claim rationale shall demonstrate that the statement of the security problem  
1619 definition is consistent with the statement of the security problem definition in the PP-  
1620 Configuration<sup>4</sup> or PPs for which conformance is being claimed.

1621 9.3.2.2.11 ASE\_CCL.1.11C

1622 The conformance claim rationale shall demonstrate that the statement of security objectives is  
1623 consistent with the statement of security objectives in the PP-Configuration<sup>5</sup> or PPs for which  
1624 conformance is being claimed.

1625 9.3.2.2.12 ASE\_CCL.1.12C

1626 The conformance claim rationale shall demonstrate that the statement of security  
1627 requirements is consistent with the statement of security requirements in the PP-  
1628 Configuration<sup>6</sup> or PPs for which conformance is being claimed.

1629 9.3.2.2.13 ASE\_CCL.1.13C

1630 The conformance claim for PP(s) and PP-Configuration(s) shall be exact, strict, or  
1631 demonstrable or a list of conformance types.

1632 9.3.2.2.14 ASE\_CCL.1.14C

1633 The conformance statement may identify the set of ISO/IEC-18045-derived Evaluation Methods  
1634 and Evaluation Activities that shall be used. The identified Evaluation Methods and Evaluation  
1635 Activities shall be only those that are included in a package, a PP, or a PP-Module in a PP-  
1636 Configuration claimed by the ST.

1637 9.3.2.3 Evaluator action elements

1638 9.3.2.3.1 ASE\_CCL.1.1E

1639 The evaluator shall confirm that the information provided meets all requirements for content  
1640 and presentation of evidence.

1641 9.4 Security problem definition (ASE\_SPD)

1642 9.4.1 Objectives

1643 This part of the ST defines the security problem to be addressed by the TOE and the operational  
1644 environment of the TOE.

---

<sup>4</sup> In practice, this refers to the union of SPDs of the PPs and the PP-Modules defined in the PP-Configuration.

<sup>5</sup> In practice, this refers to the union of security objectives of the PPs and the PP-Modules defined in the PP-Configuration.

<sup>6</sup> In practice, this refers to the union of SFRs of the PPs and the PP-Modules defined in the PP-Configuration.

1645 Evaluation of the security problem definition is required to demonstrate that the security problem  
1646 intended to be addressed by the TOE and its operational environment, is clearly defined.

#### 1647 **9.4.2 ASE\_SPD.1 Security problem definition**

1648 Dependencies: No dependencies.

##### 1649 **9.4.2.1 Developer action elements**

###### 1650 **9.4.2.1.1 ASE\_SPD.1.1D**

1651 **The developer shall provide a security problem definition.**

##### 1652 **9.4.2.2 Content and presentation elements**

###### 1653 **9.4.2.2.1 ASE\_SPD.1.1C**

1654 **The security problem definition shall describe the threats.**

###### 1655 **9.4.2.2.2 ASE\_SPD.1.2C**

1656 **All threats shall be described in terms of a threat agent, an asset, and an adverse action.**

###### 1657 **9.4.2.2.3 ASE\_SPD.1.3C**

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

###### 1659 **9.4.2.2.4 ASE\_SPD.1.4C**

1660 **The security problem definition shall describe the assumptions about the operational**  
1661 **environment of the TOE.**

##### 1662 **9.4.2.3 Evaluator action elements**

###### 1663 **9.4.2.3.1 ASE\_SPD.1.1E**

1664 **The evaluator shall confirm that the information provided meets all requirements for content**  
1665 **and presentation of evidence.**

#### 1666 **9.5 Security objectives (ASE\_OBJ)**

##### 1667 **9.5.1 Objectives**

1668 The security objectives are a concise statement of the intended response to the security problem  
1669 defined through the Security problem definition (ASE\_SPD) family.

1670 Evaluation of the security objectives is required to demonstrate that the security objectives adequately  
1671 and completely address the security problem definition, that the division of this problem between the  
1672 TOE and its operational environment is clearly defined.

##### 1673 **9.5.2 Component levelling**

1674 The components in this family are levelled on whether they prescribe only security objectives for the  
1675 operational environment (ASE\_OBJ.1), or also security objectives for the TOE (ASE\_OBJ.2).

1676           **9.5.3 ASE\_OBJ.1 Direct rationale Security objectives for the operational environment**

1677   Dependencies:   No dependencies

1678   **9.5.3.1 Developer action elements**

1679   **9.5.3.1.1 ASE\_OBJ.1.1D**

1680   **The developer shall provide a statement of security objectives.**

1681   **9.5.3.2 Content and presentation elements**

1682   **9.5.3.2.1 ASE\_OBJ.1.1C**

1683   **The statement of security objectives shall describe the security objectives for the operational**  
1684   **environment.**

1685   **9.5.3.3 Evaluator action elements**

1686   **9.5.3.3.1 ASE\_OBJ.1.1E**

1687   **The evaluator shall confirm that the information provided meets all requirements for content**  
1688   **and presentation of evidence.**

1689           **9.5.4 ASE\_OBJ.2 Security objectives**

1690   Dependencies:   ASE\_SPD.1 Security problem definition

1691   **9.5.4.1 Developer action elements**

1692   **9.5.4.1.1 ASE\_OBJ.2.1D**

1693   **The developer shall provide a statement of security objectives.**

1694   **9.5.4.1.2 ASE\_OBJ.2.2D**

1695   **The developer shall provide a security objectives rationale.**

1696   **9.5.4.2 Content and presentation elements**

1697   **9.5.4.2.1 ASE\_OBJ.2.1C**

1698   **The statement of security objectives shall describe the security objectives for the TOE and the**  
1699   **security objectives for the operational environment.**

1700   **9.5.4.2.2 ASE\_OBJ.2.2C**

1701   **The security objectives rationale shall trace each security objective for the TOE back to threats**  
1702   **countered by that security objective and OSPs enforced by that security objective.**

1703   **9.5.4.2.3 ASE\_OBJ.2.3C**

1704   **The security objectives rationale shall trace each security objective for the operational**  
1705   **environment back to threats countered by that security objective, OSPs enforced by that**  
1706   **security objective, and assumptions upheld by that security objective.**



1707 **9.5.4.2.4 ASE\_OBJ.2.4C**

1708 The security objectives **rationale** shall **demonstrate that** the security objectives **counter all threats**.

1709 **9.5.4.2.5 ASE\_OBJ.2.5C**

1710 **The security objectives rationale shall demonstrate that the security objectives enforce all**  
1711 **OSPs.**

1712 **9.5.4.2.6 ASE\_OBJ.2.6C**

1713 **The security objectives rationale shall demonstrate that the security objectives for the**  
1714 **operational environment uphold all assumptions.**

1715 **9.5.4.3 Evaluator action elements**

1716 **9.5.4.3.1 ASE\_OBJ.2.1E**

1717 **The evaluator shall confirm that the information provided meets all requirements for content**  
1718 **and presentation of evidence.**

1719 **9.6 Extended components definition (ASE\_ECD)**

1720 **9.6.1 Objectives**

1721 Extended security requirements are requirements that are not based on components from ISO/IEC  
1722 15408-2 or this document, but are based on extended components: components defined by the ST  
1723 author.

1724 Evaluation of the definition of extended components is necessary to determine that they are clear and  
1725 unambiguous, and that they are necessary, i.e. they may not be clearly expressed using existing  
1726 ISO/IEC 15408-2 or this document components.

1727 **9.6.2 ASE\_ECD.1 Extended components definition**

1728 Dependencies: No dependencies.

1729 **9.6.2.1 Developer action elements**

1730 **9.6.2.1.1 ASE\_ECD.1.1D**

1731 **The developer shall provide a statement of security requirements.**

1732 **9.6.2.1.2 ASE\_ECD.1.2D**

1733 **The developer shall provide an extended components definition.**

1734 **9.6.2.2 Content and presentation elements**

1735 **9.6.2.2.1 ASE\_ECD.1.1C**

1736 **The statement of security requirements shall identify all extended security requirements.**

- 1737 **9.6.2.2.2 ASE\_ECD.1.2C**
- 1738 **The extended components definition shall define an extended component for each extended**  
 1739 **security requirement.**
- 1740 **9.6.2.2.3 ASE\_ECD.1.3C**
- 1741 **The extended components definition shall describe how each extended component is related to**  
 1742 **the existing ISO/IEC 15408 series components, families, and classes.**
- 1743 **9.6.2.2.4 ASE\_ECD.1.4C**
- 1744 **The extended components definition shall use the existing ISO/IEC 15408 series components,**  
 1745 **families, classes, and methodology as a model for presentation.**
- 1746 **9.6.2.2.5 ASE\_ECD.1.5C**
- 1747 **The extended components shall consist of measurable and objective elements such that**  
 1748 **conformance or nonconformance to these elements may be demonstrated.**
- 1749 **9.6.2.3 Evaluator action elements**
- 1750 **9.6.2.3.1 ASE\_ECD.1.1E**
- 1751 **The evaluator shall confirm that the information provided meets all requirements for content**  
 1752 **and presentation of evidence.**
- 1753 **9.6.2.3.2 ASE\_ECD.1.2E**
- 1754 **The evaluator shall confirm that no extended component may be clearly expressed using**  
 1755 **existing components.**
- 1756 **9.7 Security requirements (ASE\_REQ)**
- 1757 **9.7.1 Objectives**
- 1758 **The SFRs form a clear, unambiguous and well-defined description of the expected security behaviour**  
 1759 **of the TOE. The SARs form a clear, unambiguous and canonical description of the expected activities**  
 1760 **that will be undertaken to gain assurance in the TOE.**
- 1761 **Evaluation of the security requirements is required to ensure that they are clear, unambiguous and**  
 1762 **well-defined.**
- 1763 **9.7.2 Component levelling**
- 1764 **The components in this family are levelled on whether they are stated as is (see ASE\_REQ.1), or**  
 1765 **whether the SFRs are derived from security objectives for the TOE (see ASE\_REQ.2.).**
- 1766 **9.7.3 ASE\_REQ.1Direct Rationale Stated security requirements**
- 1767 **Dependencies: ASE\_ECD.1 Extended components definition**

- 1768 **9.7.3.1 Developer action elements**
- 1769 **9.7.3.1.1 ASE\_REQ.1.1D**
- 1770 **The developer shall provide a statement of security requirements.**
- 1771 **9.7.3.1.2 ASE\_REQ.1.2D**
- 1772 **The developer shall provide a security requirements rationale.**
- 1773 **9.7.3.2 Content and presentation elements**
- 1774 **9.7.3.2.1 ASE\_REQ.1.1C**
- 1775 **The statement of security requirements shall describe the SFRs and the SARs.**
- 1776 **9.7.3.2.2 ASE\_REQ.1.2C**
- 1777 **For a single-assurance ST, the statement of security requirements shall define the global set of**  
 1778 **SARs that apply to the entire TOE. The sets of SARs shall be consistent with the PPs or PP-**  
 1779 **Configuration to which the ST claims conformance.**
- 1780 **9.7.3.2.3 ASE\_REQ.1.3C**
- 1781 **For a multi-assurance ST, the statement of security requirements shall define the global set of**  
 1782 **SARs that apply to the entire TOE. The statement of security requirements shall additionally**  
 1783 **define the SARs that apply to each sub-TSF. The sets of SARs shall be consistent with the multi-**  
 1784 **assurance PP-Configuration to which the ST claims conformance.**
- 1785 **9.7.3.2.4 ASE\_REQ.1.4C**
- 1786 **All subjects, objects, operations, security attributes, external entities and other terms that are**  
 1787 **used in the SFRs and the SARs shall be defined.**
- 1788 **9.7.3.2.5 ASE\_REQ.1.5C**
- 1789 **The statement of security requirements shall include a natural language description, part of**  
 1790 **which describes how the SFRs combine together to provide security functionality in terms of**  
 1791 **the architecture that is observable to Administrators and other users, or in terms of internal**  
 1792 **features or properties.**
- 1793 **9.7.3.2.6 ASE\_REQ.1.6C**
- 1794 **The statement of security requirements shall identify all operations on the security**  
 1795 **requirements.**
- 1796 **9.7.3.2.7 ASE\_REQ.1.7C**
- 1797 **All operations shall be performed correctly.**
- 1798 **9.7.3.2.8 ASE\_REQ.1.8C**
- 1799 **Each dependency of the security requirements shall either be satisfied, or the security**  
 1800 **requirements rationale shall justify the dependency not being satisfied.**

- 1801 **9.7.3.2.9 ASE\_REQ.1.9C**
- 1802 **The security requirements rationale shall trace each SFR back to the security objectives threats**  
 1803 **countered by that SFR and OSPs enforced by that SFR.**
- 1804 **9.7.3.2.10 ASE\_REQ.1.10C**
- 1805 **The security requirements rationale shall trace each security objective for the operational**  
 1806 **environment back to threats countered by that security objective, OSPs enforced by that**  
 1807 **security objective, and assumptions upheld by that security objective.**
- 1808 **9.7.3.2.11 ASE\_REQ.1.11C**
- 1809 **The security requirements rationale shall demonstrate that the SFRs (in conjunction with the**  
 1810 **security objectives for the environment) counter all threats for the TOE.**
- 1811 **9.7.3.2.12 ASE\_REQ.1.12C**
- 1812 **The security requirements rationale shall demonstrate that the SFRs (in conjunction with the**  
 1813 **security objectives for the environment) enforce all OSPs.**
- 1814 **9.7.3.2.13 ASE\_REQ.1.13C**
- 1815 **The security requirements rationale shall demonstrate that the security objectives for the**  
 1816 **operational environment uphold all assumptions.**
- 1817 **9.7.3.2.14 ASE\_REQ.1.14C**
- 1818 **The statement of security requirements shall be internally consistent.**
- 1819 **9.7.3.2.15 ASE\_REQ.1.15C**
- 1820 **In the case of a multi-assurance ST that extends the sets of SARs of the PP-Configuration it**  
 1821 **claims conformance to, the security requirements rationale shall include an assurance**  
 1822 **rationale that justifies the consistency of the extension.**
- 1823 **9.7.3.3 Evaluator action elements**
- 1824 **9.7.3.3.1 ASE\_REQ.1.1E**
- 1825 **The evaluator shall confirm that the information provided meets all requirements for content**  
 1826 **and presentation of evidence.**
- 1827 **9.7.4 ASE\_REQ.2 Derived security requirements**
- 1828 Dependencies: ASE\_OBJ.2 Security objectives
- 1829 ASE\_ECD.1 Extended components definition
- 1830 **9.7.4.1 Developer action elements**
- 1831 **9.7.4.1.1 ASE\_REQ.2.1D**
- 1832 **The developer shall provide a statement of security requirements.**

1833 **9.7.4.1.2 ASE\_REQ.2.2D**

1834 The developer shall provide a security requirements rationale.

1835 **9.7.4.2 Content and presentation elements**

1836 **9.7.4.2.1 ASE\_REQ.2.1C**

1837 The statement of security requirements shall describe the SFRs and the SARs.

1838 **9.7.4.2.2 ASE\_REQ.2.2C**

1839 **For a single-assurance ST, the statement of security requirements shall define the global set of**  
 1840 **SARs that apply to the entire TOE. The sets of SARs shall be consistent with the PPs or PP-**  
 1841 **Configuration to which the ST claims conformance.**

1842 **9.7.4.2.3 ASE\_REQ.2.3C**

1843 **For a multi-assurance ST, the statement of security requirements shall define the global set of**  
 1844 **SARs that apply to the entire TOE. The statement of security requirements shall additionally**  
 1845 **define the SARs that apply to each sub-TSF. The sets of SARs shall be consistent with the multi-**  
 1846 **assurance PP-Configuration to which the ST claims conformance.**

1847 **9.7.4.2.4 ASE\_REQ.2.4C**

1848 All subjects, objects, operations, security attributes, external entities and other terms that are used in  
 1849 the SFRs and the SARs shall be defined.

1850 **9.7.4.2.5 ASE\_REQ.2.5C**

1851 The statement of security requirements shall identify all operations on the security requirements.

1852 **9.7.4.2.6 ASE\_REQ.2.6C**

1853 All operations shall be performed correctly.

1854 **9.7.4.2.7 ASE\_REQ.2.7C**

1855 Each dependency of the security requirements shall either be satisfied, or the security requirements  
 1856 rationale shall justify the dependency not being satisfied.

1857 **9.7.4.2.8 ASE\_REQ.2.8C**

1858 The security requirements rationale shall trace each SFR back to the SPD elements **for the TOE**.

1859 **9.7.4.2.9 ASE\_REQ.2.9C**

1860 **The security requirements rationale shall demonstrate that the SFRs meet all security**  
 1861 **objectives for the TOE.**

1862 **9.7.4.2.10 ASE\_REQ.2.10C**

1863 **The security requirements rationale shall explain why the SARs were chosen.**

1864 **9.7.4.2.11 ASE\_REQ.2.11C**

1865 The statement of security requirements shall be internally consistent.

1866 **9.7.4.2.12 ASE\_REQ.2.12C**

1867 **9.7.4.3** The security requirements rationale shall demonstrate that the security objectives for the  
1868 operational environment uphold all assumptions.

1869 **9.7.4.4 Evaluator action elements**

1870 **9.7.4.4.1 ASE\_REQ.2.1E**

1871 The evaluator shall confirm that the information provided meets all requirements for content and  
1872 presentation of evidence.

1873 **9.8 TOE summary specification (ASE\_TSS)**

1874 **9.8.1 Objectives**

1875 The TOE summary specification enables evaluators and potential consumers to gain a general  
1876 understanding of how the TOE is implemented.

1877 Evaluation of the TOE summary specification is necessary to determine whether it is adequately  
1878 described how the TOE:

- 1879 • meets its SFRs;
- 1880 • protects itself against interference, logical tampering and bypass;
- 1881 and whether the TOE summary specification is consistent with other narrative descriptions of the  
1882 TOE.

1883 **9.8.2 Component levelling**

1884 The components in this family are levelled on whether the TOE summary specification only needs to  
1885 describe how the TOE meets the SFRs, or whether the TOE summary specification also needs to  
1886 describe how the TOE protects itself against logical tampering and bypass. This additional description  
1887 may be used in special circumstances where there might be a specific concern regarding the TOE  
1888 security architecture.

1889 **9.8.3 ASE\_TSS.1 TOE summary specification**

1890 Dependencies: ASE\_INT.1 ST introduction

1891 [ASE\\_REQ.1DirectRationaleStated security requirementsASE\\_REQ.1Stated security requirements](#)

1892 ADV\_FSP.1 Basic functional specification

1893 **9.8.3.1 Developer action elements**

1894 **9.8.3.1.1 ASE\_TSS.1.1D**

1895 **The developer shall provide a TOE summary specification.**

- 1896 **9.8.3.2 Content and presentation elements**
- 1897 **9.8.3.2.1 ASE\_TSS.1.1C**
- 1898 **The TOE summary specification shall describe how the TOE meets each SFR.**
- 1899 **9.8.3.3 Evaluator action elements**
- 1900 **9.8.3.3.1 ASE\_TSS.1.1E**
- 1901 **The evaluator shall confirm that the information provided meets all requirements for content**
- 1902 **and presentation of evidence.**
- 1903 **9.8.3.3.2 ASE\_TSS.1.2E**
- 1904 **The evaluator shall confirm that the TOE summary specification is consistent with the TOE**
- 1905 **overview and the TOE description.**
- 1906 **9.8.4 ASE\_TSS.2 TOE summary specification with architectural design summary**
- 1907 Dependencies: ASE\_INT.1 ST introduction
- 1908 [ASE\\_REQ.1Direct Rationale Stated security requirementsASE\\_REQ.1 Stated security requirements](#)
- 1909 ADV\_ARC.1 Security architecture description
- 1910 **9.8.4.1 Developer action elements**
- 1911 **9.8.4.1.1 ASE\_TSS.2.1D**
- 1912 The developer shall provide a TOE summary specification.
- 1913 **9.8.4.2 Content and presentation elements**
- 1914 **9.8.4.2.1 ASE\_TSS.2.1C**
- 1915 The TOE summary specification shall describe how the TOE meets each SFR.
- 1916 **9.8.4.2.2 ASE\_TSS.2.2C**
- 1917 **The TOE summary specification shall describe how the TOE protects itself against interference**
- 1918 **and logical tampering.**
- 1919 **9.8.4.2.3 ASE\_TSS.2.3C**
- 1920 **The TOE summary specification shall describe how the TOE protects itself against bypass.**
- 1921 **9.8.4.3 Evaluator action elements**
- 1922 **9.8.4.3.1 ASE\_TSS.2.1E**
- 1923 The evaluator shall confirm that the information provided meets all requirements for content and
- 1924 presentation of evidence.

1925 **9.8.4.3.2 ASE\_TSS.2.2E**

1926 The evaluator shall confirm that the TOE summary specification is consistent with the TOE overview  
1927 and the TOE description.

1928 **9.9 Consistency of composite product Security Target (ASE\_COMP)**1929 **9.9.1 Objectives**

1930 The aim of this activity is to determine whether the Security Target of the composite product<sup>7</sup> does not  
1931 contradict the Security Target of the underlying platform<sup>8</sup>.

1932 **9.9.2 ASE\_COMP.1 Consistency of Security Target**

1933 Dependencies: No dependencies

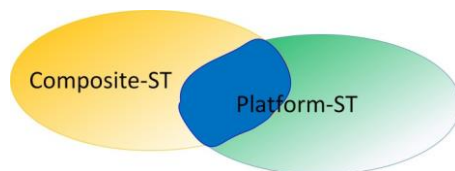
1934 **9.9.2.1 Application notes**

1935 These application notes aid the developer to create as well as the evaluator to analyse a composite  
1936 Security Target and describe a general methodology for it. For detailed information / guidance please  
1937 refer to the single work units below.

1938 In order to create a composite Security Target, the developer should perform the following steps:

1939 Step 1: The developer formulates a preliminary Security Target for the composite product (the  
1940 Composite-ST) using the standard code of practice. The Composite-ST can be formulated  
1941 independently of the Security Target of the underlying platform (Platform-ST) – at least as long as  
1942 there are no formal PP conformance claims.

1943 Step 2: The developer determines the overlap between Platform-ST and Composite-ST through  
1944 analysing and comparing their TOE Security Functionality (TSF) <sup>910</sup>:



1945

1946 **Figure 7 - Overlap between Platform-ST and Composite-ST**

1947 Step 3: The developer determines under which conditions he can trust in and rely on the Platform-  
1948 TSF being used by the Composite-ST without a new examination.

---

<sup>7</sup> denoted by Composite-ST in the following

<sup>8</sup> denoted by Platform-ST in the following. Generally, a Security Target expresses a security policy for the TOE defined.

<sup>9</sup> because the TSF enforce the Security Target (together with organisational measures enforcing security objectives for the operational environment of the TOE).

<sup>10</sup> 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.



1949 Having undertaken these steps the developer completes the preliminary Security Target for the  
1950 composite product.

1951 It is not mandatory that the platform and the composite TOE are being certified according to same  
1952 edition of 15408. It is due to the fact that the application can rely on some security services of the  
1953 platform, if (i) the assurance level of the platform covers the intended assurance level of the composite  
1954 TOE and (ii) the platform's security certificate is valid and up-to-date. Equivalence of single assurance  
1955 components (and, hence, of assurance levels) belonging to different **ISO/IEC 15408 series editions**  
1956 have to be established / acknowledged by the Composite Product Certification Body, cf. chapter X (to  
1957 **be completed After CEM has been updated**) in ISO/IEC 18045.

1958 If a PP conformance is claimed (e.g. composite ST claim conformance to a PP that claims conformance  
1959 to a hardware PP), the consistency check can be reduced to the elements of the Security Target having  
1960 not already been covered by these Protection Profiles.

1961 The fact of compliance to a PP is not sufficient to avoid inconsistencies. Assume the following situation,  
1962 where → stands for "complies with"

1963 Composite-ST → SW PP → HW PP ← platform-ST

1964 The SW PP may require any kind of conformance<sup>11</sup>, but this does not change the 'additional elements'  
1965 that the platform-ST may introduce to the HW PP. In conclusion, these additions are not necessarily  
1966 consistent with the composite-ST/SW PP additions: There is no scenario that ensures the consistency  
1967 'by construction'.

1968 Note that consistency may not be direct matching: e.g. objectives for the platform environment may  
1969 become objectives for the composite TOE.

## 1970 9.9.2.2 Developer action elements

### 1971 9.9.2.2.1 ASE\_COMP.1.1D

1972 **The developer shall provide a statement of compatibility between the Composite Security**  
1973 **Target and the Platform Security Target. This statement may be provided within the Composite**  
1974 **Product Security Target.**

## 1975 9.9.2.3 Content and presentation elements

### 1976 9.9.2.3.1 ASE\_COMP.1.1C

1977 **The statement of compatibility shall describe the separation of the Platform-TSF into relevant**  
1978 **Platform-TSF being used by the Composite-ST and others.**

### 1979 9.9.2.3.2 ASE\_COMP.1.2C

1980 **The statement of compatibility between the Composite Security Target and the Platform**  
1981 **Security Target shall show (e.g. in form of a mapping) that the Security Targets of the composite**  
1982 **product and of the underlying platform match, i.e. that there is no conflict between security**  
1983 **environments, security objectives, and security requirements of the Composite Security Target**  
1984 **and the Platform Security Target. It may be provided by indicating of the concerned elements**

---

<sup>11</sup> e.g. "strict", "exact" or "demonstrable" according to ISO/IEC 15408.

1985 **directly in the Security Target for the composite product followed by explanatory text, if**  
 1986 **necessary.**

1987 **9.9.2.4 Evaluator action elements**

1988 **9.9.2.4.1 ASE\_COMP.1.1E**

1989 **The evaluator shall confirm that the information provided meets all requirements for content**  
 1990 **and presentation of evidence.**

1991 **10 Class ADV: Development**

1992 **10.1 Introduction**

1993 The requirements of the Development class provide information about the TOE. The knowledge  
 1994 obtained by this information is used as the basis for conducting vulnerability analysis and testing upon  
 1995 the TOE, as described in the AVA and ATE classes.

1996 The Development class encompasses seven families of requirements for structuring and representing  
 1997 the TSF at various levels and varying forms of abstraction. These families include:

- 1998 • requirements for the description (at the various levels of abstraction) of the design and  
 1999 implementation of the SFRs (ADV\_FSP, ADV\_TDS, ADV\_IMP and ADV\_COMP)
- 2000 • requirements for the description of the architecture-oriented features of domain separation, TSF  
 2001 self-protection and non-bypassability of the security functionality (ADV\_ARC)
- 2002 • requirements for a security policy model and for correspondence mappings between security  
 2003 policy model and the functional specification (ADV\_SPM)
- 2004 • requirements on the internal structure of the TSF, which covers aspects such as modularity,  
 2005 layering, and minimisation of complexity (ADV\_INT)

2006 When documenting the security functionality of a TOE, there are two properties that need to be  
 2007 demonstrated. The first property is that the security functionality works correctly; that is, it performs  
 2008 as specified. The second property, and one that is arguably harder to demonstrate, is that the TOE  
 2009 cannot be used in a way such that the security functionality can be corrupted or bypassed. These two  
 2010 properties require somewhat different approaches in analysis, and so the families in ADV are  
 2011 structured to support these different approaches. The families Functional specification (ADV\_FSP),  
 2012 TOE design (ADV\_TDS), Implementation representation (ADV\_IMP), and Security policy modelling  
 2013 (ADV\_SPM) deal with the first property: the specification of the security functionality. The families  
 2014 Security Architecture (ADV\_ARC) and TSF internals (ADV\_INT) deal with the second property: the  
 2015 specification of the design of the TOE demonstrating the security functionality cannot be corrupted or  
 2016 bypassed. It should be noted that both properties need to be realised: the more confidence one has  
 2017 that the properties are satisfied, the more trustworthy the TOE is. The TSF of a composite product are  
 2018 represented at various levels of abstraction in the families of the development class ADV. The family  
 2019 Composite design compliance (ADV\_COMP) determines whether the requirements on the application,  
 2020 imposed by the underlying platform, are fulfilled in a composite product. Due to the distribution of the  
 2021 TSF of a composite product to various levels in the families of the class ADV, this family is not  
 2022 represented in Figure 8. The components in the families are designed so that more assurance can be  
 2023 gained as the components hierarchically increase.

2024 The paradigm for the families targeted at the first property is one of design decomposition. At the  
2025 highest level, there is a functional specification of the TSF in terms of its interfaces (describing *what*  
2026 the TSF does in terms of requests to the TSF for services and resulting responses), decomposing the  
2027 TSF into smaller units (dependent on the assurance desired and the complexity of the TOE) and  
2028 describing *how* the TSF accomplishes its functions (to a level of detail commensurate with the  
2029 assurance level), and showing the implementation of the TSF. A formal model of the security  
2030 behaviour also may be given. All levels of decomposition are used in determining the completeness  
2031 and accuracy of all other levels, ensuring that the levels are mutually supportive. The requirements for  
2032 the various TSF representations are separated into different families, to allow the PP/ST author to  
2033 specify which TSF representations are required. The level chosen will dictate the assurance  
2034 desired/gained.

2035 Figure 8 indicates the relationships among the various TSF representations of the ADV class, as well as  
2036 their relationships with other classes. As the figure indicates, the APE and ASE classes define the  
2037 requirements for the correspondence between the SFRs and the security objectives for the TOE. Class  
2038 ASE also defines requirements for the correspondence between both the security objectives and SFRs,  
2039 and for the TOE summary specification which explains how the TOE meets its SFRs. The activities of  
2040 ALC\_CMC.5.2E include the verification that the TSF that is tested under the ATE and AVA classes is in  
2041 fact the one described by all of the ADV decomposition levels.

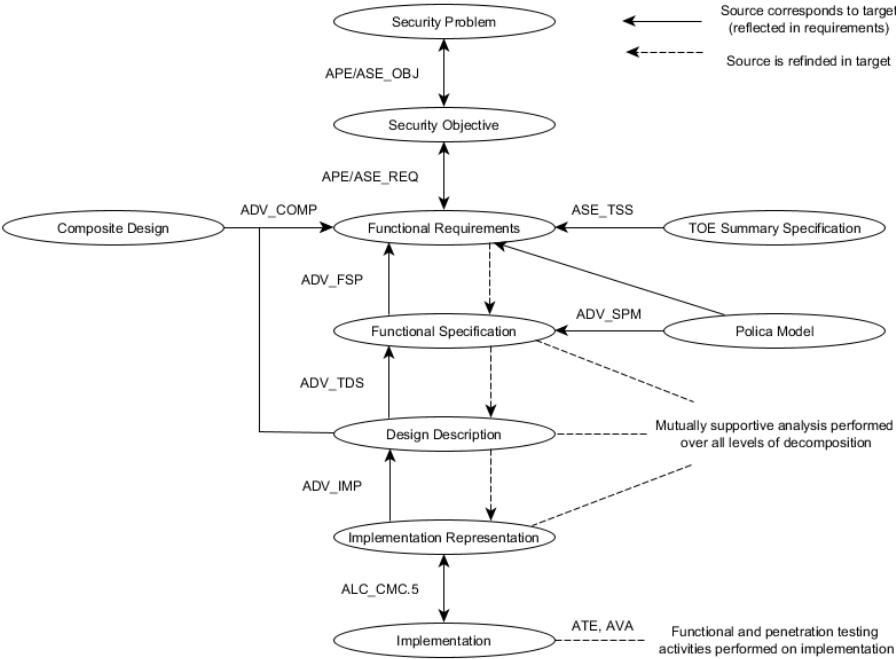


Figure 8 — Relationships of ADV constructs to one another and to other families

The requirements for all other correspondence shown in Figure 8 are defined in the ADV class for the TOE. The Security policy modelling (ADV\_SPM) family defines the requirements for formally modelling selected SFRs, and providing correspondence between the functional specification and the formal model. Each assurance family specific to a TSF representation (i.e., Functional specification (ADV\_FSP), TOE design (ADV\_TDS) and Implementation representation (ADV\_IMP)) defines requirements relating that TSF representation to the SFRs. All decompositions must accurately reflect all other decompositions (i.e., be mutually supportive); the developer supplies the tracings in the last .C elements of the components. Assurance relating to this factor is obtained during the analysis for each of the levels of decomposition by referring to other levels of decomposition (in a recursive fashion) while the analysis of a particular level of decomposition is being performed; the evaluator verifies the correspondence as part of the second E element. The understanding gained from these levels of decomposition form the basis of the functional and penetration testing efforts.

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

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

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

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

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

All portions of the TSF are *security relevant*, meaning that they must preserve the security of the TOE as expressed by the SFRs and requirements for domain separation and non-bypassability. One aspect of security relevance is the degree to which a portion of the TSF enforces a security requirement. Since

different portions of the TOE play different roles (or no apparent role at all) in enforcing security requirements, this creates a continuum of SFR relevance: at one end of this continuum are portions of the TOE that are termed *SFR-enforcing*. Such portions play a direct role in implementing any SFR on the TOE. Such SFRs refer to any functionality provided by one of the SFRs contained in the ST. It should be noted that the definition of *plays a role in* for SFR-enforcing functionality is impossible to express quantitatively. For example, in the implementation of a Discretionary Access Control (DAC) mechanism, a very narrow view of *SFR-enforcing* might be the several lines of code that actually perform the check of a subject's attributes against the object's attributes. A broader view would include the software entity (e.g., C function) that contained the several lines of code. A broader view still would include callers of the C function, since they would be responsible for enforcing the decision returned by the attribute check. A still broader view would include any code in the call tree (or programming equivalent for the implementation language used) for that C function (e.g., a sort function that sorted access control list entries in a first-match algorithm implementation). At some point, the component is not so much *enforcing* the security policy but rather plays a *supporting* role; such components are termed *SFR supporting*. One of the characteristics of SFR-supporting functionality is that it is trusted to preserve the correctness of the SFR implementation by operating without error. Such functionality may be depended on by SFR-enforcing functionality, but the dependence is generally at a functional level; for example, memory management, buffer management, etc. Further down on the security relevance continuum is functionality termed *SFR non-interfering*. Such functionality has no role in implementing the SFRs, and is likely part of the TSF because of its environment; for example, any code running in a privileged hardware mode on an operating system. It needs to be considered part of the TSF because, if compromised (or replaced by malicious code), it 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

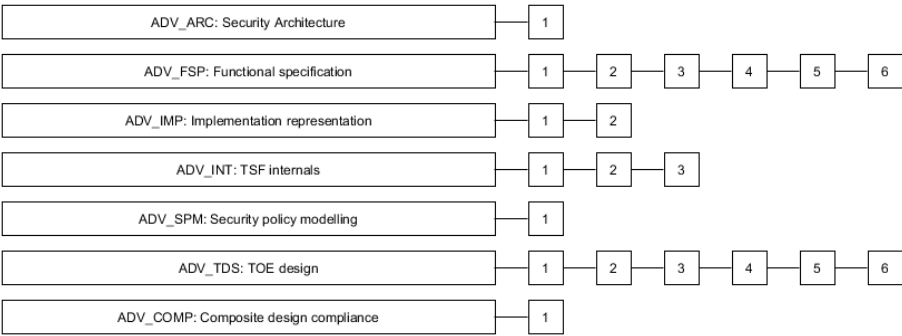
2140 semiformal style. A semiformal style reduces the ambiguity in these documents over an informal  
2141 presentation. A formal specification may also be required *in addition to* the semi-formal presentation;  
2142 the value is that a description of the TSF in more than one way will add increased assurance that the  
2143 TSF has been completely and accurately specified.

2144 An informal specification is written as prose in natural language. Natural language is used here as  
2145 meaning communication in any commonly spoken tongue (e.g. Spanish, German, French, English,  
2146 Dutch). An informal specification is not subject to any notational or special restrictions other than  
2147 those required as ordinary conventions for that language (e.g. grammar and syntax). While no  
2148 notational restrictions apply, the informal specification is also required to provide defined meanings  
2149 for terms that are used in a context other than that accepted by normal usage.

2150 The difference between semiformal and informal documents is only a matter of formatting or  
2151 presentation: a semiformal notation includes such things as an explicit glossary of terms, a  
2152 standardised presentation format, etc. A semiformal specification is written to a standard presentation  
2153 template. The presentation should use terms consistently if written in a natural language. The  
2154 presentation may also use more structured languages/diagrams (e.g. data-flow diagrams, state  
2155 transition diagrams, entity-relationship diagrams, data structure diagrams, and process or program  
2156 structure diagrams). Whether based on diagrams or natural language, a set of conventions must be  
2157 used in the presentation. The glossary explicitly identifies the words that are being used in a precise  
2158 and constant manner; similarly, the standardised format implies that extreme care has been taken in  
2159 methodically preparing the document in a manner that maximises clarity. It should be noted that  
2160 fundamentally different portions of the TSF may have different semiformal notation conventions and  
2161 presentation styles (as long as the number of different “semiformal notations” is small); this still  
2162 conforms to the concept of a *semiformal presentation*.

2163 A formal specification is written in a notation based upon well-established mathematical concepts, and  
2164 is typically accompanied by supporting explanatory (informal) prose. These mathematical concepts  
2165 are used to define the syntax and semantics of the notation and the proof rules that support logical  
2166 reasoning. The syntactic and semantic rules supporting a formal notation should define how to  
2167 recognise constructs unambiguously and determine their meaning. There needs to be evidence that it  
2168 is impossible to derive contradictions, and all rules supporting the notation need to be defined or  
2169 referenced.

2170 Figure 9 shows the families within this class, and the hierarchy of components within the families.



2171

2172 **Figure 9 — ADV: Development class decomposition**

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

## 10.2 Security Architecture (ADV\_ARC)

### 10.2.1 Objectives

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

### 10.2.2 Component levelling

This family contains only one component.

### 10.2.3 Application notes

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

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

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

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

2215 This family consists of requirements for a security architecture description that describes the self-  
2216 protection, domain separation, non-bypassability principles, including a description of how these  
2217 principles are supported by the parts of the TOE that are used for TSF initialisation.

2218 In case of a **multi-assurance evaluation** the properties of self-protection, domain separation, and  
2219 non-bypassability may also be described for boundaries between the **sub-TSF**.

2220 Additional information on the security architecture properties of self-protection, domain separation,  
2221 and non-bypassability can be found in Annex A.1, ADV\_ARC: Supplementary material on security  
2222 architectures.

#### 2223 10.2.4 ADV\_ARC.1 Security architecture description

2224 Dependencies: ADV\_FSP.1 Basic functional specification

2225 ADV\_TDS.1 Basic design

##### 2226 10.2.4.1 Developer action elements

###### 2227 10.2.4.1.1 ADV\_ARC.1.1D

2228 The developer shall design and implement the TOE so that the security features of the TSF  
2229 cannot be bypassed.

###### 2230 10.2.4.1.2 ADV\_ARC.1.2D

2231 The developer shall design and implement the TSF so that it is able to protect itself from  
2232 tampering by untrusted active entities.

###### 2233 10.2.4.1.3 ADV\_ARC.1.3D

2234 The developer shall provide a security architecture description of the TSF.

##### 2235 10.2.4.2 Content and presentation elements

###### 2236 10.2.4.2.1 ADV\_ARC.1.1C

2237 The security architecture description shall be at a level of detail commensurate with the  
2238 description of the SFR-enforcing abstractions described in the TOE design document.

###### 2239 10.2.4.2.2 ADV\_ARC.1.2C

2240 The security architecture description shall describe the security domains maintained by the  
2241 TSF consistently with the SFRs.

###### 2242 10.2.4.2.3 ADV\_ARC.1.3C

2243 The security architecture description shall describe how the TSF initialisation process is  
2244 secure.

###### 2245 10.2.4.2.4 ADV\_ARC.1.4C

2246 The security architecture description shall demonstrate that the TSF protects itself from  
2247 tampering.



2248 **10.2.4.2.5 ADV\_ARC.1.5C**

2249 **The security architecture description shall demonstrate that the TSF prevents bypass of the**  
 2250 **SFR-enforcing functionality.**

2251 **10.2.4.3 Evaluator action elements**2252 **10.2.4.3.1 ADV\_ARC.1.1E**

2253 **The evaluator shall confirm that the information provided meets all requirements for content**  
 2254 **and presentation of evidence.**

2255 **10.3 Functional specification (ADV\_FSP)**2256 **10.3.1 Objectives**

2257 This family levies requirements upon the functional specification, which describes the TSF interfaces  
 2258 (TSFIs). The TSFI consists of all means by which external entities (or subjects in the TOE but outside of  
 2259 the TSF) supply data to the TSF, receive data from the TSF and invoke services from the TSF. It does  
 2260 *not* describe how the TSF processes those service requests, nor does it describe the communication  
 2261 when the TSF invokes services from its operational environment; this information is addressed by the  
 2262 TOE design (ADV\_TDS) and Reliance of dependent component (ACO\_REL) families, respectively.

2263 This family provides assurance directly by allowing the evaluator to understand how the TSF meets  
 2264 the claimed SFRs. It also provides assurance indirectly, as input to other assurance families and  
 2265 classes:

- 2266 • ADV\_ARC, where the description of the TSFIs may be used to gain better understanding of how the  
 2267 TSF is protected against corruption (i.e. subversion of self-protection or domain separation)  
 2268 and/or bypass;
- 2269 • ATE, where the description of the TSFIs is an important input for both developer and evaluator  
 2270 testing;
- 2271 • AVA, where the description of the TSFIs is used to search for vulnerabilities.

2272 **10.3.2 Component levelling**

2273 The components in this family are levelled on the degree of detail required of the description of the  
 2274 TSFIs, and the degree of formalism required of the description of the TSFIs.

2275 **10.3.3 Application notes**

2276 Once the TSFIs are determined (see XXX for guidance and examples of determining TSFI), they are  
 2277 described. At lower-level components, developers focus their documentation (and evaluators focus  
 2278 their analysis) on the more security-relevant aspects of the TOE. Three categories of TSFIs are defined,  
 2279 based upon the relevance the services available through them have to the SFRs being claimed:

- 2280 • If a service available through an interface can be traced to one of the SFRs levied on the TSF,  
 2281 then that interface is termed *SFR-enforcing*. Note that it is possible that an interface may have  
 2282 various services and results, some of which may be SFR-enforcing and some of which may not.

Kommentiert [NC2]: Verweis einfügen

- Interfaces to (or services available through an interface relating to) services that SFR-enforcing functionality depends upon, but need only to function correctly in order for the security policies of the TOE to be preserved, are termed *SFR-supporting*.

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

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

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

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

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

#### 10.3.3.1 Detail about the Interfaces

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

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

The interface's *method of use* describes how the interface is supposed to be used. This description should be built around the various interactions available at that interface. For instance, if the interface were a Unix command shell, *ls*, *mv* and *cp* would be interactions for that interface. For each interaction the method of use describes what the interaction does, both for behaviour seen at the interface (e.g.

the programmer calling the API, the Windows users changing a setting in the registry, etc.) as well as behaviour at other interfaces (e.g. generating an audit record).

*Parameters* are explicit inputs to and outputs from an interface that control the behaviour of that interface. For example, parameters are the arguments supplied to an API; the various fields in a packet for a given network protocol; the individual key values in the Windows Registry; the signals across a set of pins on a chip; the flags that can be set for the *ls*, etc. The parameters are “identified” with a simple list of what they are.

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

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

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

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

An example functional specification is provided in A.2.4.

### 10.3.3.2 Components of this Family

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

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

At ADV\_FSP.2 Security-enforcing functional specification, the developer is required to provide the purpose, method of use, parameters, and parameter descriptions for all TSFIs. Additionally, for the

2367 SFR-enforcing TSFIs the developer has to describe the SFR-enforcing actions and direct error  
2368 messages.

2369 At ADV\_FSP.3 Functional specification with complete summary, the developer must now, in addition to  
2370 the information required at ADV\_FSP.2, provide enough information about the SFR-supporting and  
2371 SFR-non-interfering actions to show that they are not SFR-enforcing. Further, the developer must now  
2372 document all of the direct error messages resulting from the invocation of SFR-enforcing TSFIs.

2373 At ADV\_FSP.4 Complete functional specification, all TSFIs - whether SFR-enforcing, SFR-supporting or  
2374 SFR-non-interfering - must be described to the same degree, including all of the direct error messages.

2375 At ADV\_FSP.5 Complete semi-formal functional specification with additional error information, the  
2376 TSFIs descriptions also include error messages that do not result from an invocation of a TSFI.

2377 At ADV\_FSP.6 Complete semi-formal functional specification with additional formal specification, in  
2378 addition to the information required by ADV\_FSP.5, all remaining error messages are included. The  
2379 developer must also provide a formal description of the TSFI. This provides an alternative view of the  
2380 TSFI that may expose inconsistencies or incomplete specification.

#### 2381 **10.3.4 ADV\_FSP.1 Basic functional specification**

2382 Dependencies: No dependencies.

##### 2383 **10.3.4.1 Developer action elements**

###### 2384 **10.3.4.1.1 ADV\_FSP.1.1D**

2385 **The developer shall provide a functional specification.**

###### 2386 **10.3.4.1.2 ADV\_FSP.1.2D**

2387 **The developer shall provide a tracing from the functional specification to the SFRs.**

##### 2388 **10.3.4.2 Content and presentation elements**

###### 2389 **10.3.4.2.1 ADV\_FSP.1.1C**

2390 **The functional specification shall describe the purpose and method of use for each SFR-**  
2391 **enforcing and SFR-supporting TSFI.**

###### 2392 **10.3.4.2.2 ADV\_FSP.1.2C**

2393 **The functional specification shall identify all parameters associated with each SFR-enforcing**  
2394 **and SFR-supporting TSFI.**

###### 2395 **10.3.4.2.3 ADV\_FSP.1.3C**

2396 **The functional specification shall provide rationale for the implicit categorisation of interfaces**  
2397 **as SFR-non-interfering.**

###### 2398 **10.3.4.2.4 ADV\_FSP.1.4C**

2399 **The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.**

2400	<b>10.3.4.3 Evaluator action elements</b>
2401	<b>10.3.4.3.1 ADV_FSP.1.1E</b>
2402	<b>The evaluator shall confirm that the information provided meets all requirements for content</b>
2403	<b>and presentation of evidence.</b>
2404	<b>10.3.4.3.2 ADV_FSP.1.2E</b>
2405	<b>The evaluator shall determine that the functional specification is an accurate and complete</b>
2406	<b>instantiation of the SFRs.</b>
2407	<b>10.3.5 ADV_FSP.2 Security-enforcing functional specification</b>
2408	Dependencies: ADV_TDS.1 Basic design
2409	<b>10.3.5.1 Developer action elements</b>
2410	<b>10.3.5.1.1 ADV_FSP.2.1D</b>
2411	The developer shall provide a functional specification.
2412	<b>10.3.5.1.2 ADV_FSP.2.2D</b>
2413	The developer shall provide a tracing from the functional specification to the SFRs.
2414	<b>10.3.5.2 Content and presentation elements</b>
2415	<b>10.3.5.2.1 ADV_FSP.2.1C</b>
2416	<b>The functional specification shall completely represent the TSF.</b>
2417	<b>10.3.5.2.2 ADV_FSP.2.2C</b>
2418	The functional specification shall describe the purpose and method of use for <b>all</b> TSFI.
2419	<b>10.3.5.2.3 ADV_FSP.2.3C</b>
2420	The functional specification shall identify <b>and describe</b> all parameters associated with each TSFI.
2421	<b>10.3.5.2.4 ADV_FSP.2.4C</b>
2422	<b>For each SFR-enforcing TSFI, the functional specification shall describe the SFR-enforcing actions</b>
2423	<b>associated with the TSFI.</b>
2424	<b>10.3.5.2.5 ADV_FSP.2.5C</b>
2425	<b>For each SFR-enforcing TSFI, the functional specification shall describe direct error messages</b>
2426	<b>resulting from processing associated with the SFR-enforcing actions.</b>
2427	<b>10.3.5.2.6 ADV_FSP.2.6C</b>
2428	The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

2429 **10.3.5.3 Evaluator action elements**

2430 **10.3.5.3.1 ADV\_FSP.2.1E**

2431 The evaluator shall confirm that the information provided meets all requirements for content and  
2432 presentation of evidence.

2433 **10.3.5.3.2 ADV\_FSP.2.2E**

2434 The evaluator shall determine that the functional specification is an accurate and complete  
2435 instantiation of the SFRs.

2436 **10.3.6 ADV\_FSP.3 Functional specification with complete summary**

2437 Dependencies: ADV\_TDS.1 Basic design

2438 **10.3.6.1 Developer action elements**

2439 **10.3.6.1.1 ADV\_FSP.3.1D**

2440 The developer shall provide a functional specification.

2441 **10.3.6.1.2 ADV\_FSP.3.2D**

2442 The developer shall provide a tracing from the functional specification to the SFRs.

2443 **10.3.6.2 Content and presentation elements**

2444 **10.3.6.2.1 ADV\_FSP.3.1C**

2445 The functional specification shall completely represent the TSF.

2446 **10.3.6.2.2 ADV\_FSP.3.2C**

2447 The functional specification shall describe the purpose and method of use for all TSFI.

2448 **10.3.6.2.3 ADV\_FSP.3.3C**

2449 The functional specification shall identify and describe all parameters associated with each TSFI.

2450 **10.3.6.2.4 ADV\_FSP.3.4C**

2451 For each SFR-enforcing TSFI, the functional specification shall describe the SFR-enforcing actions  
2452 associated with the TSFI.

2453 **10.3.6.2.5 ADV\_FSP.3.5C**

2454 For each SFR-enforcing TSFI, the functional specification shall describe direct error messages resulting  
2455 from **SFR-enforcing actions and exceptions** associated with **invocation of the TSFI**.

2456 **10.3.6.2.6 ADV\_FSP.3.6C**

2457 **The functional specification shall summarise the SFR-supporting and SFR-non-interfering**  
2458 **actions associated with each TSFI.**

- 2459 **10.3.6.2.7 ADV\_FSP.3.7C**
- 2460 The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.
- 2461 **10.3.6.3 Evaluator action elements**
- 2462 **10.3.6.3.1 ADV\_FSP.3.1E**
- 2463 The evaluator shall confirm that the information provided meets all requirements for content and  
2464 presentation of evidence.
- 2465 **10.3.6.3.2 ADV\_FSP.3.2E**
- 2466 The evaluator shall determine that the functional specification is an accurate and complete  
2467 instantiation of the SFRs.
- 2468 **10.3.7 ADV\_FSP.4 Complete functional specification**
- 2469 Dependencies: ADV\_TDS.1 Basic design
- 2470 **10.3.7.1 Developer action elements**
- 2471 **10.3.7.1.1 ADV\_FSP.4.1D**
- 2472 The developer shall provide a functional specification.
- 2473 **10.3.7.1.2 ADV\_FSP.4.2D**
- 2474 The developer shall provide a tracing from the functional specification to the SFRs.
- 2475 **10.3.7.2 Content and presentation elements**
- 2476 **10.3.7.2.1 ADV\_FSP.4.1C**
- 2477 The functional specification shall completely represent the TSF.
- 2478 **10.3.7.2.2 ADV\_FSP.4.2C**
- 2479 The functional specification shall describe the purpose and method of use for all TSFI.
- 2480 **10.3.7.2.3 ADV\_FSP.4.3C**
- 2481 The functional specification shall identify and describe all parameters associated with each TSFI.
- 2482 **10.3.7.2.4 ADV\_FSP.4.4C**
- 2483 The functional specification shall **describe all** actions associated with each TSFI.
- 2484 **10.3.7.2.5 ADV\_FSP.4.5C**
- 2485 **The functional specification shall describe all direct error messages that may result from an**  
2486 **invocation of each TSFI.**
- 2487 **10.3.7.2.6 ADV\_FSP.4.6C**
- 2488 The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

2489 **10.3.7.3 Evaluator action elements**

2490 **10.3.7.3.1 ADV\_FSP.4.1E**

2491 The evaluator shall confirm that the information provided meets all requirements for content and  
2492 presentation of evidence.

2493 **10.3.7.3.2 ADV\_FSP.4.2E**

2494 The evaluator shall determine that the functional specification is an accurate and complete  
2495 instantiation of the SFRs.

2496 **10.3.8 ADV\_FSP.5 Complete semi-formal functional specification with additional error**  
2497 **information**

2498 Dependencies: ADV\_TDS.1 Basic design

2499 ADV\_IMP.1 Implementation representation of the TSF

2500 **10.3.8.1 Developer action elements**

2501 **10.3.8.1.1 ADV\_FSP.5.1D**

2502 The developer shall provide a functional specification.

2503 **10.3.8.1.2 ADV\_FSP.5.2D**

2504 The developer shall provide a tracing from the functional specification to the SFRs.

2505 **10.3.8.2 Content and presentation elements**

2506 **10.3.8.2.1 ADV\_FSP.5.1C**

2507 The functional specification shall completely represent the TSF.

2508 **10.3.8.2.2 ADV\_FSP.5.2C**

2509 **The functional specification shall describe the TSFI using a semi-formal style.**

2510 **10.3.8.2.3 ADV\_FSP.5.3C**

2511 The functional specification shall describe the purpose and method of use for all TSFI.

2512 **10.3.8.2.4 ADV\_FSP.5.4C**

2513 The functional specification shall identify and describe all parameters associated with each TSFI.

2514 **10.3.8.2.5 ADV\_FSP.5.5C**

2515 The functional specification shall describe all actions associated with each TSFI.

2516 **10.3.8.2.6 ADV\_FSP.5.6C**

2517 The functional specification shall describe all direct error messages that may result from an invocation  
2518 of each TSFI.



2519	<b>10.3.8.2.7 ADV_FSP.5.7C</b>
2520	<b>The functional specification shall describe all error messages that do not result from an</b>
2521	<b>invocation of a TSFI.</b>
2522	<b>10.3.8.2.8 ADV_FSP.5.8C</b>
2523	<b>The functional specification shall provide a rationale for each error message contained in the</b>
2524	<b>TSF implementation yet does not result from an invocation of a TSFI.</b>
2525	<b>10.3.8.2.9 ADV_FSP.5.9C</b>
2526	The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.
2527	<b>10.3.8.3 Evaluator action elements</b>
2528	<b>10.3.8.3.1 ADV_FSP.5.1E</b>
2529	The evaluator shall confirm that the information provided meets all requirements for content and
2530	presentation of evidence.
2531	<b>10.3.8.3.2 ADV_FSP.5.2E</b>
2532	The evaluator shall determine that the functional specification is an accurate and complete
2533	instantiation of the SFRs.
2534	<b>10.3.9 ADV_FSP.6 Complete semi-formal functional specification with additional formal</b>
2535	<b>specification</b>
2536	Dependencies: ADV_TDS.1 Basic design
2537	ADV_IMP.1 Implementation representation of the TSF
2538	<b>10.3.9.1 Developer action elements</b>
2539	<b>10.3.9.1.1 ADV_FSP.6.1D</b>
2540	The developer shall provide a functional specification.
2541	<b>10.3.9.1.2 ADV_FSP.6.2D</b>
2542	<b>The developer shall provide a formal presentation of the functional specification of the TSF.</b>
2543	<b>10.3.9.1.3 ADV_FSP.6.3D</b>
2544	The developer shall provide a tracing from the functional specification to the SFRs.
2545	<b>10.3.9.2 Content and presentation elements</b>
2546	<b>10.3.9.2.1 ADV_FSP.6.1C</b>
2547	The functional specification shall completely represent the TSF.
2548	<b>10.3.9.2.2 ADV_FSP.6.2C</b>
2549	The functional specification shall describe the TSFI using a <b>formal</b> style.

2550	<b>10.3.9.2.3 ADV_FSP.6.3C</b>
2551	The functional specification shall describe the purpose and method of use for all TSFI.
2552	<b>10.3.9.2.4 ADV_FSP.6.4C</b>
2553	The functional specification shall identify and describe all parameters associated with each TSFI.
2554	<b>10.3.9.2.5 ADV_FSP.6.5C</b>
2555	The functional specification shall describe all actions associated with each TSFI.
2556	<b>10.3.9.2.6 ADV_FSP.6.6C</b>
2557	The functional specification shall describe all direct error messages that may result from an invocation
2558	of each TSFI.
2559	<b>10.3.9.2.7 ADV_FSP.6.7C</b>
2560	The functional specification shall describe all error messages <b>contained in the TSF implementation</b>
2561	<b>representation.</b>
2562	<b>10.3.9.2.8 ADV_FSP.6.8C</b>
2563	The functional specification shall provide a rationale for each error message contained in the TSF
2564	implementation <b>that is not otherwise described in the functional specification justifying why it is</b>
2565	<b>not associated with</b> a TSFI.
2566	<b>10.3.9.2.9 ADV_FSP.6.9C</b>
2567	<b>The formal presentation of the functional specification of the TSF shall describe the TSFI using</b>
2568	<b>a formal style, supported by informal, explanatory text where appropriate.</b>
2569	<b>10.3.9.2.10 ADV_FSP.6.10C</b>
2570	The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.
2571	<b>10.3.9.3 Evaluator action elements</b>
2572	<b>10.3.9.3.1 ADV_FSP.6.1E</b>
2573	The evaluator shall confirm that the information provided meets all requirements for content and
2574	presentation of evidence.
2575	<b>10.3.9.3.2 ADV_FSP.6.2E</b>
2576	The evaluator shall determine that the functional specification is an accurate and complete
2577	instantiation of the SFRs.
2578	<b>10.4 Implementation representation (ADV_IMP)</b>
2579	<b>10.4.1 Objectives</b>
2580	The function of the Implementation representation (ADV_IMP) family is for the developer to make
2581	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

##### **10.4.4.1 Developer action elements**

###### **10.4.4.1.1 ADV\_IMP.1.1D**

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

###### **10.4.4.1.2 ADV\_IMP.1.2D**

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

##### **10.4.4.2 Content and presentation elements**

###### **10.4.4.2.1 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.**

###### **10.4.4.2.2 ADV\_IMP.1.2C**

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

###### **10.4.4.2.3 ADV\_IMP.1.3C**

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

2661	<b>10.4.4.3 Evaluator action elements</b>
2662	<b>10.4.4.3.1 ADV_IMP.1.1E</b>
2663	<b>The evaluator shall confirm that, for the selected sample of the implementation representation, the information provided meets all requirements for content and presentation of evidence.</b>
2664	
2665	<b>10.4.5 ADV_IMP.2 Complete mapping of the implementation representation of the TSF</b>
2666	Dependencies: ADV_TDS.3 Basic modular design
2667	ALC_TAT.1 Well-defined development tools
2668	ALC_CMC.5 Advanced support
2669	<b>10.4.5.1 Developer action elements</b>
2670	<b>10.4.5.1.1 ADV_IMP.2.1D</b>
2671	The developer shall make available the implementation representation for the entire TSF.
2672	<b>10.4.5.1.2 ADV_IMP.2.2D</b>
2673	The developer shall provide a mapping between the TOE design description and the <b>entire</b> implementation representation.
2674	
2675	<b>10.4.5.2 Content and presentation elements</b>
2676	<b>10.4.5.2.1 ADV_IMP.2.1C</b>
2677	The implementation representation shall define the TSF to a level of detail such that the TSF may be generated without further design decisions.
2678	
2679	<b>10.4.5.2.2 ADV_IMP.2.2C</b>
2680	The implementation representation shall be in the form used by the development personnel.
2681	<b>10.4.5.2.3 ADV_IMP.2.3C</b>
2682	The mapping between the TOE design description and the <b>entire</b> implementation representation shall demonstrate their correspondence.
2683	
2684	<b>10.4.5.3 Evaluator action elements</b>
2685	<b>10.4.5.3.1 ADV_IMP.2.1E</b>
2686	The evaluator shall confirm that the information provided meets all requirements for content and presentation of evidence.
2687	
2688	<b>10.5 TSF internals (ADV_INT)</b>
2689	<b>10.5.1 Objectives</b>
2690	This family addresses the assessment of the internal structure of the TSF. A TSF whose internals are well-structured is easier to implement and less likely to contain flaws that could lead to vulnerabilities; it is also easier to maintain without the introduction of flaws.
2691	
2692	

### 10.5.2 Component levelling

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

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

### 10.5.3 Application notes

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

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

### 10.5.4 ADV\_INT.1 Well-structured subset of TSF internals

Dependencies: ADV\_IMP.1 Implementation representation of the TSF

ADV\_TDS.3 Basic modular design

ALC\_TAT.1 Well-defined development tools

#### 10.5.4.1 Objectives

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

#### 10.5.4.2 Application notes

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

- a) the structural elements of the TSF as identified in the TOE design (e.g. “The developer shall design and implement *the audit subsystem* such that it has well-structured internals.”)
- b) the implementation (e.g. “The developer shall design and implement *the encrypt.c and decrypt.c files* such that it has well-structured internals.” or “The developer shall design and implement *the 6227 IC chip* such that it has well-structured internals.”)

2730 It is likely this would not be readily accomplished by referencing the claimed SFRs (e.g. “The developer  
2731 shall design and implement *the portion of the TSF that provide anonymity as defined in FPR\_ANO.2* such  
2732 that it has well-structured internals.”) because this does not indicate where to focus the analysis.

2733 This component has limited value and would be suitable in cases where potentially-malicious  
2734 users/subjects have limited or strictly controlled access to the TSFIs or where there is another means  
2735 of protection (e.g., domain separation) that ensures the chosen subset of the TSF cannot be adversely  
2736 affected by the rest of the TSF (e.g., the cryptographic functionality, which is isolated from the rest of  
2737 the TSF, is well-structured).

#### 2738 10.5.4.3 Developer action elements

##### 2739 10.5.4.3.1 ADV\_INT.1.1D

2740 The developer shall design and implement [assignment: *subset of the TSF*] such that it has well-  
2741 structured internals.

##### 2742 10.5.4.3.2 ADV\_INT.1.2D

2743 The developer shall provide an internals description and justification.

#### 2744 10.5.4.4 Content and presentation elements

##### 2745 10.5.4.4.1 ADV\_INT.1.1C

2746 The justification shall explain the characteristics used to judge the meaning of “well-  
2747 structured”.

##### 2748 10.5.4.4.2 ADV\_INT.1.2C

2749 The TSF internals description shall demonstrate that the assigned subset of the TSF is well-  
2750 structured.

#### 2751 10.5.4.5 Evaluator action elements

##### 2752 10.5.4.5.1 ADV\_INT.1.1E

2753 The evaluator shall confirm that the information provided meets all requirements for content  
2754 and presentation of evidence.

##### 2755 10.5.4.5.2 ADV\_INT.1.2E

2756 The evaluator shall perform an internals analysis on the assigned subset of the TSF.

#### 2757 10.5.5 ADV\_INT.2 Well-structured internals

2758 Dependencies: ADV\_IMP.1 Implementation representation of the TSF

2759 ADV\_TDS.3 Basic modular design

2760 ALC\_TAT.1 Well-defined development tools

2761 **10.5.5.1 Objectives**

2762 The objective of this component is to provide a means for requiring the TSF to be well-structured. The  
2763 intent is that the entire TSF has been designed and implemented using sound engineering principles.

2764 **10.5.5.2 Application notes**

2765 Judgements on the adequacy of the structure are expected to be derived from the specific technologies  
2766 used in the TOE. This component calls for identifying the standards for measuring the characteristic of  
2767 being well-structured.

2768 **10.5.5.3 Developer action elements**

2769 **10.5.5.3.1 ADV\_INT.2.1D**

2770 The developer shall design and implement the **entire TSF** such that it has well-structured internals.

2771 **10.5.5.3.2 ADV\_INT.2.2D**

2772 The developer shall provide an internals description and justification.

2773 **10.5.5.4 Content and presentation elements**

2774 **10.5.5.4.1 ADV\_INT.2.1C**

2775 The justification shall **describe** the characteristics used to judge the meaning of “well-structured”.

2776 **10.5.5.4.2 ADV\_INT.2.2C**

2777 The TSF internals description shall demonstrate that the **entire TSF** is well-structured.

2778 **10.5.5.5 Evaluator action elements**

2779 **10.5.5.5.1 ADV\_INT.2.1E**

2780 The evaluator shall confirm that the information provided meets all requirements for content and  
2781 presentation of evidence.

2782 **10.5.5.5.2 ADV\_INT.2.2E**

2783 The evaluator shall perform an internals analysis on the TSF.

2784 **10.5.6 ADV\_INT.3 Minimally complex internals**

2785 Dependencies: ADV\_IMP.1 Implementation representation of the TSF

2786 ADV\_TDS.3 Basic modular design

2787 ALC\_TAT.1 Well-defined development tools

2788 **10.5.6.1 Objectives**

2789 The objective of this component is to provide a means for requiring the TSF to be well-structured and  
2790 of minimal complexity. The intent is that the entire TSF has been designed and implemented using  
2791 sound engineering principles.



2792 **10.5.6.2 Application notes**

2793 Judgements on the adequacy of the structure and complexity are expected to be derived from the  
2794 specific technologies used in the TOE. This component calls for identifying the standards for  
2795 measuring the structure and complexity.

2796 **10.5.6.3 Developer action elements**

2797 **10.5.6.3.1 ADV\_INT.3.1D**

2798 The developer shall design and implement the entire TSF such that it has well-structured internals.

2799 **10.5.6.3.2 ADV\_INT.3.2D**

2800 The developer shall provide an internals description and justification.

2801 **10.5.6.4 Content and presentation elements**

2802 **10.5.6.4.1 ADV\_INT.3.1C**

2803 The justification shall describe the characteristics used to judge the meaning of “well-structured” **and**  
2804 **“complex”**.

2805 **10.5.6.4.2 ADV\_INT.3.2C**

2806 The TSF internals description shall demonstrate that the entire TSF is well-structured **and is not**  
2807 **overly complex**.

2808 **10.5.6.5 Evaluator action elements**

2809 **10.5.6.5.1 ADV\_INT.3.1E**

2810 The evaluator shall confirm that the information provided meets all requirements for content and  
2811 presentation of evidence.

2812 **10.5.6.5.2 ADV\_INT.3.2E**

2813 The evaluator shall perform an internals analysis on the **entire** TSF.

2814 **10.6 Security policy modelling (ADV\_SPM)**

2815 **10.6.1 Objectives**

2816 It is the objective of this family to provide additional assurance from the development of a formal  
2817 *security policy model* of the TSFI behaviour of the TSF, and establishing a correspondence between the  
2818 functional specification and this security policy model. Preserving internal consistency the security  
2819 policy model is expected to formally establish the security principles from its characteristics by means  
2820 of a mathematical proof.

2821 **10.6.2 Component levelling**

2822 This family contains only one component.

### 10.6.3 Application notes

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

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

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

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

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

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

2868 **10.6.4 ADV\_SPM.1 Formal TOE security policy model**

2869 Dependencies: ADV\_FSP.5 Complete semi-formal functional specification with additional error  
2870 information

2871 ADV\_FSP.6 Complete semi-formal functional specification with additional formal  
2872 specification

2873 **10.6.4.1 Developer action elements**

2874 **10.6.4.1.1 ADV\_SPM.1.1D**

2875 **The developer shall provide a formal security policy model for the TSFI behaviour of the TOE.**

2876 **10.6.4.1.2 ADV\_SPM.1.2D**

2877 **The developer shall determine all TSFIs and analyze for each TSFI whether its behaviour can be**  
2878 **modelled by the formal security policy model. If a TSFI cannot be modelled, for example caused**  
2879 **by technical limitations, the developer shall analyze the impact of not modelling the TSFI**  
2880 **behaviour on the security of the TOE. If parts of the TSFI behaviour cannot be modelled due to**  
2881 **technical limitations, the remaining parts shall nevertheless be covered by the formal model.**

2882 **10.6.4.1.3 ADV\_SPM.1.3D**

2883 **The formal security policy model shall identify the modelled TSFIs. For each TSFI covered by**  
2884 **the formal security policy model, the model shall identify the related SFRs and security policies**  
2885 **in the ST. For each SFR covered by the formal security policy model, the model shall identify the**  
2886 **relevant portions of the statement of SFRs.**

2887 **10.6.4.1.4 ADV\_SPM.1.4D**

2888 **For all TSFIs that are not modelled by the formal security policy model, the developer shall**  
2889 **identify the affected SFRs and security policies in the ST.**

2890 **10.6.4.1.5 ADV\_SPM.1.5D**

2891 **The developer shall provide a formal proof of correspondence between the model and any**  
2892 **formal functional specification. The proof of correspondence shall relate model elements and**  
2893 **TSFIs. With ADV\_SPM.1.3D, the given proof of correspondence thereby implicitly provides a**  
2894 **correspondence between model elements and SFRs, as well as model elements and security**  
2895 **policies. The developer defines a structured process for identifying and presenting**  
2896 **corresponding items formally.**

2897 **10.6.4.1.6 ADV\_SPM.1.6D**

2898 **The developer shall provide a demonstration of correspondence between the model and the**  
2899 **functional specification. This item shall demonstrate the correspondence between model**  
2900 **elements and TSFIs.**

2901 **10.6.4.2 Content and presentation elements**

2902 **10.6.4.2.1 ADV\_SPM.1.1C**

2903 **The model shall define security for the TOE and provide a formal proof that the TOE cannot**  
2904 **reach a state that is not secure.**

2905 **10.6.4.2.2 ADV\_SPM.1.2C**2906 **The developer shall provide an analysis why the chosen modelling formalism is appropriate.**2907 **10.6.4.2.3 ADV\_SPM.1.3C**

2908 **If tool support is used, the developer shall identify the tool chain used to verify the formal**  
 2909 **security policy model, including environments and version numbers. The developer shall**  
 2910 **provide arguments why the tool chain is suited and trustworthy.**

2911 **10.6.4.2.4 ADV\_SPM.1.4C**

2912 **The developer shall define how the formal analysis of the formal security policy model may be**  
 2913 **reproduced (for example, applying an interactive theorem prover to prove correctness of the**  
 2914 **formal security policy model).**

2915 **10.6.4.2.5 ADV\_SPM.1.5C**

2916 **The model shall be in a formal style, supported by explanatory text as required, and identify**  
 2917 **the TSFIs that are modelled. Additionally, the SFRs and security policies of the TSF that are**  
 2918 **modelled via the TSFI behaviour shall be presented. The model shall identify all TSFIs that are**  
 2919 **not modelled (compare ADV\_SPM.1.2D) and present the affected SFRs and security policies. The**  
 2920 **model shall explain the reason for not modelling TSFIs and provide an impact analysis which**  
 2921 **shows that correctness of the formal model is not affected.**

2922 **10.6.4.2.6 ADV\_SPM.1.6C**

2923 **The correspondence between the model and the functional specification shall be at the correct**  
 2924 **level of formality. The developer shall describe the correspondence analysis process and define**  
 2925 **the applied understanding of correspondence. If a semi-formal functional specification is**  
 2926 **provided, the correspondence must be shown semi-formally. If a formal functional**  
 2927 **specification is provided, the correspondence must be shown formally.**

2928 **10.6.4.2.7 ADV\_SPM.1.7C**

2929 **The correspondence shall show that the model is consistent and complete with respect to the**  
 2930 **functional specification.**

2931 **10.6.4.3 Evaluator action elements**2932 **10.6.4.3.1 ADV\_SPM.1.1E**

2933 **The evaluator shall confirm that the information provided meets all requirements for content**  
 2934 **and presentation of evidence.**

2935 **10.7 TOE design (ADV\_TDS)**2936 **10.7.1 Objectives**

2937 **The design description of a TOE provides both context for a description of the TSF, and a thorough**  
 2938 **description of the TSF. As assurance needs increase, the level of detail provided in the description also**  
 2939 **increases. As the size and complexity of the TSF increase, multiple levels of decomposition are**  
 2940 **appropriate. The design requirements are intended to provide information (commensurate with the**  
 2941 **given assurance level) so that a determination can be made that the security functional requirements**  
 2942 **are realised.**

## 10.7.2 Component levelling

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

## 10.7.3 Application notes

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

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

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

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

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

### 10.7.3.1 Detail about the Subsystems and Modules

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

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

- 2984 c) A subsystem's *behaviour* is what it does. The behaviour may also be categorised as SFR-enforcing,  
 2985 SFR-supporting, or SFR-non-interfering. The behaviour of the subsystem is never categorised as  
 2986 more SFR-relevant than the category of the subsystem itself. For example, an SFR-enforcing  
 2987 subsystem can have SFR-enforcing behaviour as well as SFR-supporting or SFR-non-interfering  
 2988 behaviour.
- 2989 d) A *behaviour summary* of a subsystem is an overview of the actions it performs (e.g. "The TCP  
 2990 subsystem assembles IP datagrams into reliable byte streams").
- 2991 e) A *behaviour description* of a subsystem is an explanation of everything it does. This description  
 2992 should be at a level of detail that one can readily determine whether the behaviour has any  
 2993 relevance to the enforcement of the SFRs.
- 2994 f) A *description of interactions* among or between subsystems or modules identifies the reason that  
 2995 subsystems or modules communicate, and characterises the information that is passed. It need not  
 2996 define the information to the same level of detail as an interface specification. For example, it  
 2997 would be sufficient to say "subsystem X requests a block of memory from the memory manager,  
 2998 which responds with the location of the allocated memory.
- 2999 g) A *description of interfaces* provides the details of how the interactions among modules are  
 3000 achieved. Rather than describing the reason the modules are communicating or the purpose of  
 3001 their communication (that is, the description of interactions), the description of interfaces  
 3002 describes the details of how that communication is accomplished, in terms of the structure and  
 3003 contents of the messages, semaphores, internal process communications, etc.
- 3004 h) The *purpose* describes how a module provides their functionality. It provides sufficient detail that  
 3005 no further design decisions are needed. The correspondence between the implementation  
 3006 representation that implements the module, and the purpose of the module should be readily  
 3007 apparent.
- 3008 i) A module is otherwise *described* in terms of whatever is identified in the element.
- 3009 Subsystems and modules, and "SFR-enforcing", etc. are all further explained in greater detail in A.4,  
 3010 ADV\_TDS: Subsystems and Modules.

#### 3011 10.7.4 ADV\_TDS.1 Basic design

3012 Dependencies: ADV\_FSP.2 Security-enforcing functional specification

##### 3013 10.7.4.1 Developer action elements

###### 3014 10.7.4.1.1 ADV\_TDS.1.1D

3015 The developer shall provide the design of the TOE.

###### 3016 10.7.4.1.2 ADV\_TDS.1.2D

3017 The developer shall provide a mapping from the TSFI of the functional specification to the  
 3018 lowest level of decomposition available in the TOE design.

- 3019 **10.7.4.2 Content and presentation elements**
- 3020 **10.7.4.2.1 ADV\_TDS.1.1C**
- 3021 The design shall describe the structure of the TOE in terms of subsystems.
- 3022 **10.7.4.2.2 ADV\_TDS.1.2C**
- 3023 The design shall identify all subsystems of the TSF.
- 3024 **10.7.4.2.3 ADV\_TDS.1.3C**
- 3025 The design shall provide the behaviour summary of each SFR-supporting or SFR-non-  
3026 interfering TSF subsystem.
- 3027 **10.7.4.2.4 ADV\_TDS.1.4C**
- 3028 The design shall summarise the SFR-enforcing behaviour of the SFR-enforcing subsystems.
- 3029 **10.7.4.2.5 ADV\_TDS.1.5C**
- 3030 The design shall provide a description of the interactions among SFR-enforcing subsystems of  
3031 the TSF, and between the SFR-enforcing subsystems of the TSF and other subsystems of the TSF.
- 3032 **10.7.4.2.6 ADV\_TDS.1.6C**
- 3033 The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE  
3034 design that they invoke.
- 3035 **10.7.4.3 Evaluator action elements**
- 3036 **10.7.4.3.1 ADV\_TDS.1.1E**
- 3037 The evaluator shall confirm that the information provided meets all requirements for content  
3038 and presentation of evidence.
- 3039 **10.7.4.3.2 ADV\_TDS.1.2E**
- 3040 The evaluator shall determine that the design is an accurate and complete instantiation of all  
3041 security functional requirements.
- 3042 **10.7.5 ADV\_TDS.2 Architectural design**
- 3043 Dependencies: ADV\_FSP.3 Functional specification with complete summary
- 3044 **10.7.5.1 Developer action elements**
- 3045 **10.7.5.1.1 ADV\_TDS.2.1D**
- 3046 The developer shall provide the design of the TOE.
- 3047 **10.7.5.1.2 ADV\_TDS.2.2D**
- 3048 The developer shall provide a mapping from the TSFI of the functional specification to the lowest level  
3049 of decomposition available in the TOE design.

3050	<b>10.7.5.2 Content and presentation elements</b>
3051	<b>10.7.5.2.1 ADV_TDS.2.1C</b>
3052	The design shall describe the structure of the TOE in terms of subsystems.
3053	<b>10.7.5.2.2 ADV_TDS.2.2C</b>
3054	The design shall identify all subsystems of the TSF.
3055	<b>10.7.5.2.3 ADV_TDS.2.3C</b>
3056	<b>The design shall provide the behaviour summary of each SFR non-interfering subsystem of the</b>
3057	<b>TSF.</b>
3058	<b>10.7.5.2.4 ADV_TDS.2.4C</b>
3059	The design shall <b>describe</b> the SFR-enforcing behaviour of the SFR-enforcing subsystems.
3060	<b>10.7.5.2.5 ADV_TDS.2.5C</b>
3061	The design shall summarise the <b>SFR-supporting and SFR-non-interfering</b> behaviour of the SFR-
3062	enforcing subsystems.
3063	<b>10.7.5.2.6 ADV_TDS.2.6C</b>
3064	The design shall summarise the behaviour of the <b>SFR-supporting</b> subsystems.
3065	<b>10.7.5.2.7 ADV_TDS.2.7C</b>
3066	<b>The design shall provide a description of the interactions among all subsystems of the TSF.</b>
3067	<b>10.7.5.2.8 ADV_TDS.2.8C</b>
3068	The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE design that
3069	they invoke.
3070	<b>10.7.5.3 Evaluator action elements</b>
3071	<b>10.7.5.3.1 ADV_TDS.2.1E</b>
3072	The evaluator shall confirm that the information provided meets all requirements for content and
3073	presentation of evidence.
3074	<b>10.7.5.3.2 ADV_TDS.2.2E</b>
3075	The evaluator shall determine that the design is an accurate and complete instantiation of all security
3076	functional requirements.
3077	<b>10.7.6 ADV_TDS.3 Basic modular design</b>
3078	Dependencies: ADV_FSP.4 Complete functional specification



- 3079 **10.7.6.1 Developer action elements**
- 3080 **10.7.6.1.1 ADV\_TDS.3.1D**
- 3081 The developer shall provide the design of the TOE.
- 3082 **10.7.6.1.2 ADV\_TDS.3.2D**
- 3083 The developer shall provide a mapping from the TSFI of the functional specification to the lowest level  
3084 of decomposition available in the TOE design.
- 3085 **10.7.6.2 Content and presentation elements**
- 3086 **10.7.6.2.1 ADV\_TDS.3.1C**
- 3087 The design shall describe the structure of the TOE in terms of subsystems.
- 3088 **10.7.6.2.2 ADV\_TDS.3.2C**
- 3089 **The design shall describe the TSF in terms of modules.**
- 3090 **10.7.6.2.3 ADV\_TDS.3.3C**
- 3091 The design shall identify all subsystems of the TSF.
- 3092 **10.7.6.2.4 ADV\_TDS.3.4C**
- 3093 The design shall **provide a description of each subsystem of the TSF.**
- 3094 **10.7.6.2.5 ADV\_TDS.3.5C**
- 3095 The design shall provide a description of the interactions among all subsystems of the TSF.
- 3096 **10.7.6.2.6 ADV\_TDS.3.6C**
- 3097 **The design shall provide a mapping from the subsystems of the TSF to the modules of the TSF.**
- 3098 **10.7.6.2.7 ADV\_TDS.3.7C**
- 3099 The design shall describe each **SFR-enforcing module in terms of its purpose and relationship**  
3100 **with other modules.**
- 3101 **10.7.6.2.8 ADV\_TDS.3.8C**
- 3102 **The design shall describe each SFR-enforcing module in terms of its SFR-related interfaces,**  
3103 **return values from those interfaces, interaction with other modules and called SFR-related**  
3104 **interfaces to other SFR-enforcing modules.**
- 3105 **10.7.6.2.9 ADV\_TDS.3.9C**
- 3106 The design shall describe each **SFR-supporting and SFR-non-interfering module in terms of its**  
3107 **purpose and interaction with other modules.**

3108 **10.7.6.2.10 ADV\_TDS.3.10C**

3109 The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE design that  
3110 they invoke.

3111 **10.7.6.3 Evaluator action elements**

3112 **10.7.6.3.1 ADV\_TDS.3.1E**

3113 The evaluator shall confirm that the information provided meets all requirements for content and  
3114 presentation of evidence.

3115 **10.7.6.3.2 ADV\_TDS.3.2E**

3116 The evaluator shall determine that the design is an accurate and complete instantiation of all security  
3117 functional requirements.

3118 **10.7.7 ADV\_TDS.4 Semiformal modular design**

3119 Dependencies: ADV\_FSP.5 Complete semi-formal functional specification with additional error  
3120 information

3121 **10.7.7.1 Developer action elements**

3122 **10.7.7.1.1 ADV\_TDS.4.1D**

3123 The developer shall provide the design of the TOE.

3124 **10.7.7.1.2 ADV\_TDS.4.2D**

3125 The developer shall provide a mapping from the TSFI of the functional specification to the lowest level  
3126 of decomposition available in the TOE design.

3127 **10.7.7.2 Content and presentation elements**

3128 **10.7.7.2.1 ADV\_TDS.4.1C**

3129 The design shall describe the structure of the TOE in terms of subsystems.

3130 **10.7.7.2.2 ADV\_TDS.4.2C**

3131 The design shall describe the TSF in terms of modules, **designating each module as SFR-enforcing,**  
3132 **SFR-supporting, or SFR-non-interfering.**

3133 **10.7.7.2.3 ADV\_TDS.4.3C**

3134 The design shall identify all subsystems of the TSF.

3135 **10.7.7.2.4 ADV\_TDS.4.4C**

3136 The design shall provide a **semiformal** description of each subsystem of the TSF, **supported by**  
3137 **informal, explanatory text where appropriate.**

3138 **10.7.7.2.5 ADV\_TDS.4.5C**

3139 The design shall provide a description of the interactions among all subsystems of the TSF.

3140 **10.7.7.2.6 ADV\_TDS.4.6C**

3141 The design shall provide a mapping from the subsystems of the TSF to the modules of the TSF.

3142 **10.7.7.2.7 ADV\_TDS.4.7C**

3143 The design shall describe each SFR-enforcing **and SFR-supporting** module in terms of its purpose and  
3144 relationship with other modules.

3145 **10.7.7.2.8 ADV\_TDS.4.8C**

3146 The design shall describe each SFR-enforcing **and SFR-supporting** module in terms of its SFR-related  
3147 interfaces, return values from those interfaces, interaction with other modules and called SFR-related  
3148 interfaces to other SFR-enforcing **or SFR-supporting** modules.

3149 **10.7.7.2.9 ADV\_TDS.4.9C**

3150 The design shall describe each SFR-non-interfering module in terms of its purpose and interaction  
3151 with other modules.

3152 **10.7.7.2.10 ADV\_TDS.4.10C**

3153 The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE design that  
3154 they invoke.

3155 **10.7.7.3 Evaluator action elements**

3156 **10.7.7.3.1 ADV\_TDS.4.1E**

3157 The evaluator shall confirm that the information provided meets all requirements for content and  
3158 presentation of evidence.

3159 **10.7.7.3.2 ADV\_TDS.4.2E**

3160 The evaluator shall determine that the design is an accurate and complete instantiation of all security  
3161 functional requirements.

3162 **10.7.8 ADV\_TDS.5 Complete semiformal modular design**

3163 Dependencies: ADV\_FSP.5 Complete semi-formal functional specification with additional error  
3164 information

3165 **10.7.8.1 Developer action elements**

3166 **10.7.8.1.1 ADV\_TDS.5.1D**

3167 The developer shall provide the design of the TOE.

3168 **10.7.8.1.2 ADV\_TDS.5.2D**

3169 The developer shall provide a mapping from the TSFI of the functional specification to the lowest level  
3170 of decomposition available in the TOE design.

3171 **10.7.8.2 Content and presentation elements**

3172 **10.7.8.2.1 ADV\_TDS.5.1C**

3173 The design shall describe the structure of the TOE in terms of subsystems.

3174 **10.7.8.2.2 ADV\_TDS.5.2C**

3175 The design shall describe the TSF in terms of modules, designating each module as SFR-enforcing, SFR-  
3176 supporting, or SFR-non-interfering.

3177 **10.7.8.2.3 ADV\_TDS.5.3C**

3178 The design shall identify all subsystems of the TSF.

3179 **10.7.8.2.4 ADV\_TDS.5.4C**

3180 The design shall provide a semiformal description of each subsystem of the TSF, supported by  
3181 informal, explanatory text where appropriate.

3182 **10.7.8.2.5 ADV\_TDS.5.5C**

3183 The design shall provide a description of the interactions among all subsystems of the TSF.

3184 **10.7.8.2.6 ADV\_TDS.5.6C**

3185 The design shall provide a mapping from the subsystems of the TSF to the modules of the TSF.

3186 **10.7.8.2.7 ADV\_TDS.5.7C**

3187 The design shall **provide a semiformal description of** each module in terms of its **purpose**,  
3188 **interaction**, interfaces, return values from those interfaces, and called interfaces to other modules,  
3189 **supported by informal, explanatory text where appropriate.**

3190 **10.7.8.2.8 ADV\_TDS.5.8C**

3191 The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE design that  
3192 they invoke.

3193 **10.7.8.3 Evaluator action elements**

3194 **10.7.8.3.1 ADV\_TDS.5.1E**

3195 The evaluator shall confirm that the information provided meets all requirements for content and  
3196 presentation of evidence.

3197 **10.7.8.3.2 ADV\_TDS.5.2E**

3198 The evaluator shall determine that the design is an accurate and complete instantiation of all security  
3199 functional requirements.

3200 **10.7.9 ADV\_TDS.6 Complete semiformal modular design with formal high-level design**  
3201 **presentation**

3202 Dependencies: ADV\_FSP.6 Complete semi-formal functional specification with additional formal  
3203 specification

- 3204 **10.7.9.1 Developer action elements**
- 3205 **10.7.9.1.1 ADV\_TDS.6.1D**
- 3206 The developer shall provide the design of the TOE.
- 3207 **10.7.9.1.2 ADV\_TDS.6.2D**
- 3208 The developer shall provide a mapping from the TSFI of the functional specification to the lowest level  
3209 of decomposition available in the TOE design.
- 3210 **10.7.9.1.3 ADV\_TDS.6.3D**
- 3211 **The developer shall provide a formal specification of the TSF subsystems.**
- 3212 **10.7.9.1.4 ADV\_TDS.6.4D**
- 3213 **The developer shall provide a proof of correspondence between the formal specifications of the**  
3214 **TSF subsystems and of the functional specification.**
- 3215 **10.7.9.2 Content and presentation elements**
- 3216 **10.7.9.2.1 ADV\_TDS.6.1C**
- 3217 The design shall describe the structure of the TOE in terms of subsystems.
- 3218 **10.7.9.2.2 ADV\_TDS.6.2C**
- 3219 The design shall describe the TSF in terms of modules, designating each module as SFR-enforcing, SFR-  
3220 supporting, or SFR-non-interfering.
- 3221 **10.7.9.2.3 ADV\_TDS.6.3C**
- 3222 The design shall identify all subsystems of the TSF.
- 3223 **10.7.9.2.4 ADV\_TDS.6.4C**
- 3224 The design shall provide a semiformal description of each subsystem of the TSF, supported by  
3225 informal, explanatory text where appropriate.
- 3226 **10.7.9.2.5 ADV\_TDS.6.5C**
- 3227 The design shall provide a description of the interactions among all subsystems of the TSF.
- 3228 **10.7.9.2.6 ADV\_TDS.6.6C**
- 3229 The design shall provide a mapping from the subsystems of the TSF to the modules of the TSF.
- 3230 **10.7.9.2.7 ADV\_TDS.6.7C**
- 3231 The design shall **describe** each module in **semiformal style in** terms of its purpose, interaction,  
3232 interfaces, return values from those interfaces, and called interfaces to other modules, supported by  
3233 informal, explanatory text where appropriate.

3234 **10.7.9.2.8 ADV\_TDS.6.8C**

3235 **The formal specification of the TSF subsystems shall describe the TSF using a formal style,**  
3236 **supported by informal, explanatory text where appropriate.**

3237 **10.7.9.2.9 ADV\_TDS.6.9C**

3238 The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE design that  
3239 they invoke.

3240 **10.7.9.2.10 ADV\_TDS.6.10C**

3241 **The proof of correspondence between the formal specifications of the TSF subsystems and of**  
3242 **the functional specification shall demonstrate that all behaviour described in the TOE design is**  
3243 **a correct and complete refinement of the TSFI that invoked it.**

3244 **10.7.9.3 Evaluator action elements**

3245 **10.7.9.3.1 ADV\_TDS.6.1E**

3246 The evaluator shall confirm that the information provided meets all requirements for content and  
3247 presentation of evidence.

3248 **10.7.9.3.2 ADV\_TDS.6.2E**

3249 The evaluator shall determine that the design is an accurate and complete instantiation of all security  
3250 functional requirements.

3251 **10.8 Composite design compliance (ADV\_COMP)**

3252 **10.8.1 Objectives**

3253 The aim of this activity is to determine whether the requirements on the application, imposed by the  
3254 underlying platform, are fulfilled in the composite product.

3255 **10.8.2 Component levelling**

3256 This family contains only one component.

3257 **10.8.3 Application notes**

3258 The requirements on the application, imposed by the underlying platform, can be formulated in the  
3259 relevant certification report (e.g. in form of constraints and recommendations), user guidance and  
3260 ETR\_COMP (in form of observations and recommendations) for the platform. The developer of the  
3261 composite product shall regard each of these sources, if available (cf. Table D2, chapter **Fehler!**  
3262 **Verweisquelle konnte nicht gefunden werden.****D.4.7**), and implement the composite product in such a  
3263 way that the applicable requirements are fulfilled.

3264 The TSF of the composite product is represented at various levels of abstraction in the families of the  
3265 development class ADV. Experiential, the appropriate levels of design representation for examining,  
3266 whether the requirements of the platform are fulfilled by the composite product, are the TOE design  
3267 (ADV\_TDS), security architecture (ADV\_ARC) and the implementation (ADV\_IMP). In case, these design  
3268 representation levels are not available (e.g. due to the assurance package chosen is EAL1), the current  
3269 activity is not applicable (see the next paragraph for the reason).

3270 Due to the definition of the composite TOE (cf. Clause 13.3.2.2 in ISO/IEC 15408-1) the interface  
3271 between the underlying platform and the application is the internal one, hence, a functional  
3272 specification (ADV\_FSP) as representation level is not appropriate for analysing the design  
3273 compliance.

3274 Security architecture ADV\_ARC as assurance family is dedicated to ensure that integrative security  
3275 services like domain separation, self-protection and non-bypassability properly work. It is impossible  
3276 and not the sense of the composite evaluation to have an insight into the architectural internals of the  
3277 underlying platform (it is a matter of the platform evaluation). What the Composite Evaluator has to  
3278 do in the context of ADV\_ARC is

3279 i. to determine whether the application uses services of the underlying platform within its own  
3280 Composite-ST to provide domain separation, self-protection, non-bypassability and protected  
3281 start-up; if no, there is no further composite activities for ADV\_ARC; if yes, then

3282 ii. the evaluator has to determine, whether the application uses these platform-services in an  
3283 appropriate/secure way (please refer to the platform user guidance, cf. item #3 in Table D1,  
3284 chapter [Fehler! Verweisquelle konnte nicht gefunden werden.D.1.7](#)).

3285 Since consistency of the composite product security policy has already been considered in the context  
3286 of the Security Target in the assurance family ASE\_COMP, there is no necessity to consider non-  
3287 contradictoriness of the security policy model (ADV\_SPM) of the composite TOE and the security  
3288 policy model of the underlying platform.

#### 3289 **10.8.4 ADV\_COMP.1 Design compliance with the platform certification report, guidance** 3290 **and ETR\_COMP**

3291 Dependencies: No dependencies

##### 3292 **10.8.4.1**

##### 3293 **10.8.4.2 Developer action elements**

##### 3294 **10.8.4.2.1 ADV\_COMP.1.1D**

3295 **The developer shall provide a design compliance justification; cf. item #6 as well as items #3,**  
3296 **#4, #5 in Table D1, chapter [Fehler! Verweisquelle konnte nicht gefunden werden.D.1.7](#).**

##### 3297 **10.8.4.3 Content and presentation elements**

##### 3298 **10.8.4.3.1 ADV\_COMP.1.1C**

3299 **The design compliance justification shall provide a rationale for design compliance – on an**  
3300 **appropriate representation level – of how the requirements on the application, imposed by the**  
3301 **underlying platform, are fulfilled in the composite product.**

##### 3302 **10.8.4.4 Evaluator action elements**

##### 3303 **10.8.4.4.1 ADV\_COMP.1.1E**

3304 **The evaluator shall confirm that the rationale for design compliance is complete, coherent, and**  
3305 **internally consistent.**

3306 **11 Class AGD: Guidance documents**

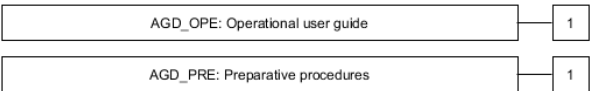
3307 **11.1 Introduction**

3308 The guidance documents class provides the requirements for guidance documentation for all user  
3309 roles. For the secure preparation and operation of the TOE it is necessary to describe all relevant  
3310 aspects for the secure handling of the TOE. The class also addresses the possibility of unintended  
3311 incorrect configuration or handling of the TOE.

3312 In many cases it may be appropriate that guidance is provided in separate documents for preparation  
3313 and operation of the TOE, or even separate for different user roles as end-users, administrators,  
3314 application programmers using software or hardware interfaces, etc.

3315 The guidance documents class is subdivided into two families which are concerned with the  
3316 preparative user guidance (what has to be done to transform the delivered TOE into its evaluated  
3317 configuration in the operational environment as described in the ST) and with the operational user  
3318 guidance (what has to be done during the operation of the TOE in its evaluated configuration).

3319 Figure 10 shows the families within this class, and the hierarchy of components within the families.



3320

3321 **Figure 10 — AGD: Guidance documents class decomposition**

3322 **11.2 Operational user guidance (AGD\_OPE)**

3323 **11.2.1 Objectives**

3324 Operational user guidance refers to written material that is intended to be used by all types of users of  
3325 the TOE in its evaluated configuration: end-users, persons responsible for maintaining and  
3326 administering the TOE in a correct manner for maximum security, and by others (e.g. programmers)  
3327 using the TOE's external interfaces. Operational user guidance describes the security functionality  
3328 provided by the TSF, provides instructions and guidelines (including warnings), helps to understand  
3329 the TSF and includes the security-critical information, and the security-critical actions required, for its  
3330 secure use. Misleading and unreasonable guidance should be absent from the guidance  
3331 documentation, and secure procedures for all modes of operation should be addressed. Insecure states  
3332 should be easy to detect.

3333 The operational user guidance provides a measure of confidence that non-malicious users,  
3334 administrators, application providers and others exercising the external interfaces of the TOE will  
3335 understand the secure operation of the TOE and will use it as intended. The evaluation of the user  
3336 guidance includes investigating whether the TOE can be used in a manner that is insecure but that the  
3337 user of the TOE would reasonably believe to be secure. The objective is to minimise the risk of human  
3338 or other errors in operation that may deactivate, disable, or fail to activate security functionality,  
3339 resulting in an undetected insecure state.



**11.2.2 Component levelling**

This family contains only one component.

**11.2.3 Application notes**

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

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

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

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

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

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

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

**11.2.4 AGD\_OPE.1 Operational user guidance**

Dependencies: ADV\_FSP.1 Basic functional specification

**11.2.4.1 Developer action elements****11.2.4.1.1 AGD\_OPE.1.1D**

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

**11.2.4.2 Content and presentation elements****11.2.4.2.1 AGD\_OPE.1.1C**

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

3376 **11.2.4.2.2 AGD\_OPE.1.2C**

3377 The operational user guidance shall describe, for each user role, how to use the available  
3378 interfaces provided by the TOE in a secure manner.

3379 **11.2.4.2.3 AGD\_OPE.1.3C**

3380 The operational user guidance shall describe, for each user role, the available functions and  
3381 interfaces, in particular all security parameters under the control of the user, indicating secure  
3382 values as appropriate.

3383 **11.2.4.2.4 AGD\_OPE.1.4C**

3384 The operational user guidance shall, for each user role, clearly present each type of security-  
3385 relevant event relative to the user-accessible functions that need to be performed, including  
3386 changing the security characteristics of entities under the control of the TSF.

3387 **11.2.4.2.5 AGD\_OPE.1.5C**

3388 The operational user guidance shall identify all possible modes of operation of the TOE  
3389 (including operation following failure or operational error), their consequences and  
3390 implications for maintaining secure operation.

3391 **11.2.4.2.6 AGD\_OPE.1.6C**

3392 The operational user guidance shall, for each user role, describe the security controls to be  
3393 followed in order to fulfil the security objectives for the operational environment as described  
3394 in the ST.

3395 **11.2.4.2.7 AGD\_OPE.1.7C**

3396 The operational user guidance shall be clear and reasonable.

3397 **11.2.4.3 Evaluator action elements**

3398 **11.2.4.3.1 AGD\_OPE.1.1E**

3399 The evaluator shall confirm that the information provided meets all requirements for content  
3400 and presentation of evidence.

3401 **11.3 Preparative procedures (AGD\_PRE)**

3402 **11.3.1 Objectives**

3403 Preparative procedures are useful for ensuring that the TOE has been received and installed in a  
3404 secure manner as intended by the developer. The requirements for preparation call for a secure  
3405 transition from the delivered TOE to its initial operational environment. This includes investigating  
3406 whether the TOE can be configured or installed in a manner that is insecure but that the user of the  
3407 TOE would reasonably believe to be secure.

3408 **11.3.2 Component levelling**

3409 This family contains only one component.

3410 **11.3.3 Application notes**

3411 It is recognised that the application of these requirements will vary depending on aspects such as  
3412 whether the TOE is delivered in an operational state, or whether it has to be installed at the TOE  
3413 owner's site, etc.

3414 The first process covered by the preparative procedures is the consumer's secure acceptance of the  
3415 received TOE in accordance with the developer's delivery procedures. If the developer has not defined  
3416 delivery procedures, security of the acceptance has to be ensured otherwise.

3417 Installation of the TOE includes transforming its operational environment into a state that conforms to  
3418 the security objectives for the operational environment provided in the ST.

3419 It might also be the case that no installation is necessary, for example a smart card. In this case it may  
3420 be inappropriate to require and analyse installation procedures.

3421 The requirements in this assurance family are presented separately from those in the Operational user  
3422 guidance (AGD\_OPE) family, due to the infrequent, possibly one-time use of the preparative  
3423 procedures.

3424 **11.3.4 AGD\_PRE.1 Preparative procedures**

3425 Dependencies: No dependencies.

3426 **11.3.4.1 Developer action elements**

3427 **11.3.4.1.1 AGD\_PRE.1.1D**

3428 **The developer shall provide the TOE including its preparative procedures.**

3429 **11.3.4.2 Content and presentation elements**

3430 **11.3.4.2.1 AGD\_PRE.1.1C**

3431 **The preparative procedures shall describe all the steps necessary for secure acceptance of the**  
3432 **delivered TOE in accordance with the developer's delivery procedures.**

3433 **11.3.4.2.2 AGD\_PRE.1.2C**

3434 **The preparative procedures shall describe all the steps necessary for secure installation of the**  
3435 **TOE and for the secure preparation of the operational environment in accordance with the**  
3436 **security objectives for the operational environment as described in the ST.**

3437 **11.3.4.3 Evaluator action elements**

3438 **11.3.4.3.1 AGD\_PRE.1.1E**

3439 **The evaluator shall confirm that the information provided meets all requirements for content**  
3440 **and presentation of evidence.**

3441 **11.3.4.3.2 AGD\_PRE.1.2E**

3442 **The evaluator shall apply the preparative procedures to confirm that the TOE can be prepared**  
3443 **securely for operation.**

## 3444 12 Class ALC: Life-cycle support

### 3445 12.1 Introduction

3446 Life-cycle support is an aspect of establishing appropriate security controls in the development,  
3447 production, delivery and maintenance of the TOE. Confidence in the correspondence between the TOE  
3448 security requirements and the TOE is greater if security analysis and the production of the evidence  
3449 are done on a regular basis as an integral part of the development, production, delivery and  
3450 maintenance activities.

3451 During the life-cycle of the TOE it is distinguished whether the TOE is under the responsibility of the  
3452 TOE developer or the user rather than whether it is located in the development or the user  
3453 environment. The point of transition is when the TOE is accepted by the user. User in this context  
3454 relates to the end-user as well as product- and system integrators.

3455 The ALC class consists of nine families:

- 3456 • Development Life-cycle definition (ALC\_LCD) provides requirements for the developer's  
3457 description of the life-cycle model used in the development, production, delivery and maintenance  
3458 life-cycle of the TOE;
- 3459 • CM capabilities (ALC\_CMC) provides requirements for the management of the configuration items;
- 3460 • CM scope (ALC\_CMS) requires a minimum set of configuration items to be managed in the defined  
3461 way;
- 3462 • ~~Developer environment security (ALC\_DVS)~~ ~~Developer environment security (ALC\_DVS)~~ is concerned with the developer's physical, logical,  
3463 procedural, personnel, and other security controls;
- 3464 • Tools and techniques (ALC\_TAT) provides requirements for the development tools and  
3465 implementation standards used by the developer;
- 3466 • Flaw remediation (ALC\_FLR) provides requirements for the handling of security flaws.
- 3467 • Delivery (ALC\_DEL) provides requirements for the procedures used for the delivery of the TOE to  
3468 the downstream user. Delivery processes occurring during the development of the TOE are  
3469 denoted rather as transfers, and are handled in the context of integration and acceptance  
3470 procedures in other families of this class.
- 3471 • ALC\_TDA is concerned with the generation of certain artefacts during the development process.
- 3472 • ALC\_COMP is concerned with the integration of composition parts and a consistency check of  
3473 delivery procedures.

3474 Throughout this class, development and related terms (developer, develop) are meant in the more  
3475 general sense to comprise development and production, whereas production specifically means the  
3476 process of transforming the implementation representation into the final TOE.

3477 Figure 11 shows the families within this class, and the hierarchy of components within the families.

Figure 11 — 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:

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

**12.2.2 Component levelling**

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

**12.2.3 Application notes**

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

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

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

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

3515 ALC\_CMC.2.3C introduces a requirement that the CM system uniquely identify all configuration items.  
3516 This also requires that modifications to configuration items result in a new, unique identifier being  
3517 assigned to the configuration item.

3518 ALC\_CMC.3.8C introduces the requirement that the evidence shall demonstrate that the CM system  
3519 operates in accordance with the CM plan. Examples of such evidence might be documentation such as  
3520 screen snapshots or audit trail output from the CM system, or a detailed demonstration of the CM  
3521 system by the developer. The evaluator is responsible for determining that this evidence is sufficient  
3522 to show that the CM system operates in accordance with the CM plan.

3523 ALC\_CMC.4.5C introduces a requirement that the CM system provide an automated means to support  
3524 the production of the TOE. This requires that the CM system provide an automated means to assist in  
3525 determining that the correct configuration items are used in generating the TOE.

3526 ALC\_CMC.5.10C introduces a requirement that the CM system provide an automated means to  
3527 ascertain the changes between the TOE and its preceding version. If no previous version of the TOE  
3528 exists, the developer still needs to provide an automated means to ascertain the changes between the  
3529 TOE and a future version of the TOE.

#### 3530 **12.2.4 ALC\_CMC.1 Labelling of the TOE**

3531 Dependencies: ALC\_CMS.1 TOE CM coverage

##### 3532 **12.2.4.1 Objectives**

3533 A unique reference is required to ensure that there is no ambiguity in terms of which instance of the  
3534 TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be  
3535 aware of which instance of the TOE they are using.

##### 3536 **12.2.4.2 Developer action elements**

###### 3537 **12.2.4.2.1 ALC\_CMC.1.1D**

3538 **The developer shall provide the TOE and a unique reference for the TOE.**

##### 3539 **12.2.4.3 Content and presentation elements**

###### 3540 **12.2.4.3.1 ALC\_CMC.1.1C**

3541 **The TOE shall be labelled with its unique reference.**

##### 3542 **12.2.4.4 Evaluator action elements**

###### 3543 **12.2.4.4.1 ALC\_CMC.1.1E**

3544 **The evaluator shall confirm that the information provided meets all requirements for content**  
3545 **and presentation of evidence.**

#### 3546 **12.2.5 ALC\_CMC.2 Use of the CM system**

3547 Dependencies: ALC\_CMS.1 TOE CM coverage

3548 **12.2.5.1 Objectives**

3549 A unique reference is required to ensure that there is no ambiguity in terms of which instance of the  
3550 TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be  
3551 aware of which instance of the TOE they are using.

3552 Unique identification of the configuration items leads to a clearer understanding of the composition of  
3553 the TOE, which in turn helps to determine those items which are subject to the evaluation  
3554 requirements for the TOE.

3555 The use of a CM system increases assurance that the configuration items are maintained in a  
3556 controlled manner.

3557 **12.2.5.2 Developer action elements**

3558 **12.2.5.2.1 ALC\_CMC.2.1D**

3559 The developer shall provide the TOE and a unique reference for the TOE.

3560 **12.2.5.2.2 ALC\_CMC.2.2D**

3561 **The developer shall provide the CM documentation.**

3562 **12.2.5.2.3 ALC\_CMC.2.3D**

3563 **The developer shall use a CM system.**

3564 **12.2.5.3 Content and presentation elements**

3565 **12.2.5.3.1 ALC\_CMC.2.1C**

3566 The TOE shall be labelled with its unique reference.

3567 **12.2.5.3.2 ALC\_CMC.2.2C**

3568 **The CM documentation shall describe the method used to uniquely identify the configuration**  
3569 **items.**

3570 **12.2.5.3.3 ALC\_CMC.2.3C**

3571 **The CM system shall uniquely identify all configuration items.**

3572 **12.2.5.4 Evaluator action elements**

3573 **12.2.5.4.1 ALC\_CMC.2.1E**

3574 The evaluator shall confirm that the information provided meets all requirements for content and  
3575 presentation of evidence.

3576 **12.2.6 ALC\_CMC.3 Authorisation controls**

3577 Dependencies: ALC\_CMS.1 TOE CM coverage

3578 ALC\_DVS.1 Identification of security measures

3579 A life-cycle model encompasses the procedures, tools and techniques used to develop  
 3580 and maintain the TOE. Aspects of the process that may be covered by such a model include design  
 3581 methods, review procedures, project management controls, change control procedures, test methods  
 3582 and acceptance procedures. An effective life-cycle model will address these aspects of the  
 3583 development and maintenance process within an overall management structure that assigns  
 3584 responsibilities and monitors progress.

3585 There are different types of acceptance situations that are dealt with at different locations in the  
 3586 criteria:

- 3587 • acceptance of parts delivered by subcontractors ("integration") should be treated in this family
- 3588 • Development Life-cycle definition (ALC LCD).
- 3589 • acceptance subsequent to internal transportations in Developer environment security  
3590 (ALC DVS).
- 3591 • acceptance of parts into the CM system in CM capabilities (ALC CMC), and
- 3592 • acceptance of the delivered TOE by the consumer in Delivery (ALC DEL).

3593 The first three types may overlap.

3594 Although life-cycle definition deals with the maintenance of the TOE and hence with aspects becoming  
 3595 relevant after the completion of the evaluation, its evaluation adds assurance through an analysis of  
 3596 the life-cycle information for the TOE provided at the time of the evaluation.

3597 A life-cycle model provides for the necessary control over the development and maintenance of the  
 3598 TOE, if the model enables sufficient minimisation of the danger that the TOE will not meet its security  
 3599 requirement.

3600 A measurable life-cycle model is a model using some quantitative valuation (arithmetic parameters  
 3601 and/or metrics) of the managed product in order to measure development properties of the product.  
 3602 Typical metrics are source code complexity metrics, defect density (errors per size of code) or mean  
 3603 time to failure. For the security evaluation all those metrics are of relevance, which are used to  
 3604 increase quality by decreasing the probability of faults and thereby in turn increasing assurance in the  
 3605 security of the TOE.

3606 One should take into account that there exist standardised life-cycle models on the one hand (like the  
 3607 waterfall model) and standardised metrics on the other hand (like error density), which may be  
 3608 combined. The ISO/IEC 15408 series does not require the life-cycle to follow exactly one standard  
 3609 defining both aspects.

3610 ALC LCD.1 Developer defined life-cycle processes~~ALC\_LCD.1 Developer defined life-cycle processes~~

#### 3611 **12.2.6.1 Objectives**

3612 A unique reference is required to ensure that there is no ambiguity in terms of which instance of the  
 3613 TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be  
 3614 aware of which instance of the TOE they are using.



3615 Unique identification of the configuration items leads to a clearer understanding of the composition of  
3616 the TOE, which in turn helps to determine those items which are subject to the evaluation  
3617 requirements for the TOE.

3618 The use of a CM system increases assurance that the configuration items are maintained in a  
3619 controlled manner.

3620 Providing controls to ensure that unauthorised modifications are not made to the TOE ("CM access  
3621 control"), and ensuring proper functionality and use of the CM system, helps to maintain the integrity  
3622 of the TOE.

#### 3623 **12.2.6.2 Developer action elements**

##### 3624 **12.2.6.2.1 ALC\_CMC.3.1D**

3625 The developer shall provide the TOE and a unique reference for the TOE.

##### 3626 **12.2.6.2.2 ALC\_CMC.3.2D**

3627 The developer shall provide the CM documentation.

##### 3628 **12.2.6.2.3 ALC\_CMC.3.3D**

3629 The developer shall use a CM system.

#### 3630 **12.2.6.3 Content and presentation elements**

##### 3631 **12.2.6.3.1 ALC\_CMC.3.1C**

3632 The TOE shall be labelled with its unique reference.

##### 3633 **12.2.6.3.2 ALC\_CMC.3.2C**

3634 The CM documentation shall describe the method used to uniquely identify the configuration items.

##### 3635 **12.2.6.3.3 ALC\_CMC.3.3C**

3636 The CM system shall uniquely identify all configuration items.

##### 3637 **12.2.6.3.4 ALC\_CMC.3.4C**

3638 **The CM system shall provide controls such that only authorised changes are made to the**  
3639 **configuration items.**

##### 3640 **12.2.6.3.5 ALC\_CMC.3.5C**

3641 **The CM documentation shall include a CM plan.**

##### 3642 **12.2.6.3.6 ALC\_CMC.3.6C**

3643 **The CM plan shall describe how the CM system is used for the development of the TOE.**

##### 3644 **12.2.6.3.7 ALC\_CMC.3.7C**

3645 **The evidence shall demonstrate that all configuration items are being maintained under the CM**  
3646 **system.**

3647 **12.2.6.3.8 ALC\_CMC.3.8C**

3648 **The evidence shall demonstrate that the CM system is being operated in accordance with the**  
3649 **CM plan.**

3650 **12.2.6.4 Evaluator action elements**

3651 **12.2.6.4.1 ALC\_CMC.3.1E**

3652 The evaluator shall confirm that the information provided meets all requirements for content and  
3653 presentation of evidence.

3654 **12.2.7 ALC\_CMC.4 Production support, acceptance procedures and automation**

3655 Dependencies: ALC\_CMS.1 TOE CM coverage

3656 ALC\_DVS.1 Identification of security measures

3657 A life-cycle model encompasses the procedures, tools and techniques used to develop  
3658 and maintain the TOE. Aspects of the process that may be covered by such a model include design  
3659 methods, review procedures, project management controls, change control procedures, test methods  
3660 and acceptance procedures. An effective life-cycle model will address these aspects of the  
3661 development and maintenance process within an overall management structure that assigns  
3662 responsibilities and monitors progress.

3663 There are different types of acceptance situations that are dealt with at different locations in the  
3664 criteria:

- 3665 • acceptance of parts delivered by subcontractors ("integration") should be treated in this family
- 3666 • Development Life-cycle definition (ALC LCD).
- 3667 • acceptance subsequent to internal transportations in Developer environment security  
3668 (ALC DVS).
- 3669 • acceptance of parts into the CM system in CM capabilities (ALC CMC), and
- 3670 • acceptance of the delivered TOE by the consumer in Delivery (ALC DEL).

3671 The first three types may overlap.

3672 Although life-cycle definition deals with the maintenance of the TOE and hence with aspects becoming  
3673 relevant after the completion of the evaluation, its evaluation adds assurance through an analysis of  
3674 the life-cycle information for the TOE provided at the time of the evaluation.

3675 A life-cycle model provides for the necessary control over the development and maintenance of the  
3676 TOE, if the model enables sufficient minimisation of the danger that the TOE will not meet its security  
3677 requirement.

3678 A measurable life-cycle model is a model using some quantitative valuation (arithmetic parameters  
3679 and/or metrics) of the managed product in order to measure development properties of the product.  
3680 Typical metrics are source code complexity metrics, defect density (errors per size of code) or mean  
3681 time to failure. For the security evaluation all those metrics are of relevance, which are used to

3682 increase quality by decreasing the probability of faults and thereby in turn increasing assurance in the  
3683 security of the TOE.

3684 One should take into account that there exist standardised life-cycle models on the one hand (like the  
3685 waterfall model) and standardised metrics on the other hand (like error density), which may be  
3686 combined. The ISO/IEC 15408 series does not require the life-cycle to follow exactly one standard  
3687 defining both aspects.

3688 ALC\_LCD.1 Developer defined life-cycle processes~~ALC\_LCD.1 Developer defined life-cycle processes~~

#### 3689 **12.2.7.1 Objectives**

3690 A unique reference is required to ensure that there is no ambiguity in terms of which instance of the  
3691 TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be  
3692 aware of which instance of the TOE they are using.

3693 Unique identification of the configuration items leads to a clearer understanding of the composition of  
3694 the TOE, which in turn helps to determine those items which are subject to the evaluation  
3695 requirements for the TOE.

3696 The use of a CM system increases assurance that the configuration items are maintained in a  
3697 controlled manner.

3698 Providing access controls to help ensure that unauthorised modifications are not made to the TOE  
3699 ("CM access control"), and ensuring proper functionality and use of the CM system, helps to maintain  
3700 the integrity of the TOE.

3701 The purpose of the acceptance procedures is to ensure that the parts of the TOE are of adequate  
3702 quality and to confirm that any creation or modification of configuration items is authorised.  
3703 Acceptance procedures are an essential element in integration processes and in the life-cycle  
3704 management of the TOE.

3705 In a CM system where the quantity and organization of configuration items is complex, it is difficult to  
3706 control changes without the support of automated tools. In particular, these automated tools need to  
3707 be able to support the numerous changes that occur during development and ensure that those  
3708 changes are authorised. It is an objective of this component to ensure that the configuration items are  
3709 controlled through automated means. In the case where the overall CM system includes more than one  
3710 CM application then automated tools can also support integration between the CM applications and of  
3711 the TOE.

3712 Production support procedures help to ensure that the generation of the TOE from a managed set of  
3713 configuration items is correctly performed in an authorised manner, particularly in the case when  
3714 different developers are involved and integration processes have to be carried out.

#### 3715 **12.2.7.2 Developer action elements**

##### 3716 **12.2.7.2.1 ALC\_CMC.4.1D**

3717 The developer shall provide the TOE and a unique reference for the TOE.

##### 3718 **12.2.7.2.2 ALC\_CMC.4.2D**

3719 The developer shall provide the CM documentation.

- 3720 **12.2.7.2.3 ALC\_CMC.4.3D**
- 3721 The developer shall use a CM system.
- 3722 **12.2.7.3 Content and presentation elements**
- 3723 **12.2.7.3.1 ALC\_CMC.4.1C**
- 3724 The TOE shall be labelled with its unique reference.
- 3725 **12.2.7.3.2 ALC\_CMC.4.2C**
- 3726 The CM documentation shall describe the method or methods used to uniquely identify the  
3727 configuration items.
- 3728 **12.2.7.3.3 ALC\_CMC.4.3C**
- 3729 The CM system shall uniquely identify all configuration items.
- 3730 **12.2.7.3.4 ALC\_CMC.4.4C**
- 3731 The CM system shall provide **automated** controls such that only authorised changes are made to the  
3732 configuration items.
- 3733 **12.2.7.3.5 ALC\_CMC.4.5C**
- 3734 **The CM system shall support the production of the TOE by automated means.**
- 3735 **12.2.7.3.6 ALC\_CMC.4.6C**
- 3736 The CM documentation shall include a CM plan.
- 3737 **12.2.7.3.7 ALC\_CMC.4.7C**
- 3738 The CM plan shall describe how the CM system is used for the development of the TOE.
- 3739 **12.2.7.3.8 ALC\_CMC.4.8C**
- 3740 **The CM plan shall describe the procedures used to accept modified or newly created**  
3741 **configuration items as part of the TOE.**
- 3742 **12.2.7.3.9 ALC\_CMC.4.9C**
- 3743 The evidence shall demonstrate that all configuration items are being maintained under the CM  
3744 system.
- 3745 **12.2.7.3.10 ALC\_CMC.4.10C**
- 3746 The evidence shall demonstrate that the CM system is being operated in accordance with the CM plan.
- 3747 **12.2.7.4 Evaluator action elements**
- 3748 **12.2.7.4.1 ALC\_CMC.4.1E**
- 3749 The evaluator shall confirm that the information provided meets all requirements for content and  
3750 presentation of evidence.

**12.2.8 ALC\_CMC.5 Advanced support**

Dependencies: ALC\_CMS.1 TOE CM coverage

ALC\_DVS.2 Sufficiency of security measures

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

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

- acceptance of parts delivered by subcontractors ("integration") should be treated in this family
- Development Life-cycle definition (ALC LCD).
- acceptance subsequent to internal transportations in Developer environment security (ALC DVS).
- acceptance of parts into the CM system in CM capabilities (ALC CMC), and
- acceptance of the delivered TOE by the consumer in Delivery (ALC DEL).

The first three types may overlap.

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

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

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

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

ALC\_LCD.1 Developer defined life-cycle processes~~ALC\_LCD.1 Developer defined life-cycle processes~~

3786 **12.2.8.1 Objectives**

3787 A unique reference is required to ensure that there is no ambiguity in terms of which instance of the  
3788 TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be  
3789 aware of which instance of the TOE they are using.

3790 Unique identification of the configuration items leads to a clearer understanding of the composition of  
3791 the TOE, which in turn helps to determine those items which are subject to the evaluation  
3792 requirements for the TOE.

3793 The use of a CM system increases assurance that the configuration items are maintained in a  
3794 controlled manner.

3795 Providing controls to ensure that unauthorised modifications are not made to the TOE ("CM access  
3796 control"), and ensuring proper functionality and use of the CM system, helps to maintain the integrity  
3797 of the TOE.

3798 The purpose of the acceptance procedures is to ensure that the parts of the TOE meet defined criteria  
3799 in regard to the integrity of the TOE. Criteria for acceptance procedures may include code review,  
3800 checking for vulnerabilities, authenticity checking, and functional testing to confirm that any creation  
3801 or modification of configuration items is authorised. Acceptance procedures are an essential element  
3802 in integration processes and in the life-cycle management of the TOE.

3803 In development environments where the configuration items are complex, it is difficult to control  
3804 changes without the support of automated tools. In particular, these automated tools need to be able  
3805 to support the numerous changes that occur during development and ensure that those changes are  
3806 authorised. It is an objective of this component to ensure that the configuration items are controlled  
3807 through automated means. If the TOE is developed by multiple developers, i.e. integration has to take  
3808 place, the use of automatic tools is adequate.

3809 Production support procedures help to ensure that the generation of the TOE from a managed set of  
3810 configuration items is correctly performed in an authorised manner, particularly in the case when  
3811 different developers are involved and integration processes have to be carried out.

3812 Requiring that the CM system be able to identify the version of the implementation representation  
3813 from which the TOE is generated helps to ensure that the integrity of this material is preserved by the  
3814 appropriate technical, physical and procedural safeguards.

3815 Providing an automated means of ascertaining changes between versions of the TOE and identifying  
3816 which configuration items are affected by modifications to other configuration items assists in  
3817 determining the impact of the changes between successive versions of the TOE. This in turn can  
3818 provide valuable information in determining whether changes to the TOE result in all configuration  
3819 items being consistent with one another.

3820 **12.2.8.2 Developer action elements**

3821 **12.2.8.2.1 ALC\_CMC.5.1D**

3822 The developer shall provide the TOE and a unique reference for the TOE.

3823 **12.2.8.2.2 ALC\_CMC.5.2D**

3824 The developer shall provide the CM documentation.

- 3825 **12.2.8.2.3 ALC\_CMC.5.3D**
- 3826 The developer shall use a CM system.
- 3827 **12.2.8.3 Content and presentation elements**
- 3828 **12.2.8.3.1 ALC\_CMC.5.1C**
- 3829 The TOE shall be labelled with its unique reference.
- 3830 **12.2.8.3.2 ALC\_CMC.5.2C**
- 3831 The CM documentation shall describe the method used to uniquely identify the configuration items.
- 3832 **12.2.8.3.3 ALC\_CMC.5.3C**
- 3833 **The CM documentation shall justify that the acceptance procedures provide for an adequate**  
 3834 **and appropriate review of changes to all configuration items.**
- 3835 **12.2.8.3.4 ALC\_CMC.5.4C**
- 3836 The CM system shall uniquely identify all configuration items.
- 3837 **12.2.8.3.5 ALC\_CMC.5.5C**
- 3838 The CM system shall provide automated controls such that only authorised changes are made to the  
 3839 configuration items.
- 3840 **12.2.8.3.6 ALC\_CMC.5.6C**
- 3841 The CM system shall support the production of the TOE by automated means.
- 3842 **12.2.8.3.7 ALC\_CMC.5.7C**
- 3843 **The CM system shall ensure that the person responsible for accepting a configuration item into**  
 3844 **CM is not the person who developed it.**
- 3845 **12.2.8.3.8 ALC\_CMC.5.8C**
- 3846 **The CM system shall identify the configuration items that comprise the TSF.**
- 3847 **12.2.8.3.9 ALC\_CMC.5.9C**
- 3848 **The CM system shall support the audit of all changes to the TOE by automated means, including**  
 3849 **the originator, date, and time in the audit trail.**
- 3850 **12.2.8.3.10 ALC\_CMC.5.10C**
- 3851 **The CM system shall provide an automated means to identify all other configuration items that**  
 3852 **are affected by the change of a given configuration item.**
- 3853 **12.2.8.3.11 ALC\_CMC.5.11C**
- 3854 **The CM system shall be able to identify the version of the implementation representation from**  
 3855 **which the TOE is generated.**

3856 **12.2.8.3.12 ALC\_CMC.5.12C**

3857 The CM documentation shall include a CM plan.

3858 **12.2.8.3.13 ALC\_CMC.5.13C**

3859 The CM plan shall describe how the CM system is used for the development of the TOE.

3860 **12.2.8.3.14 ALC\_CMC.5.14C**

3861 The CM plan shall describe the procedures used to accept modified or newly created configuration  
3862 items as part of the TOE.

3863 **12.2.8.3.15 ALC\_CMC.5.15C**

3864 The evidence shall demonstrate that all configuration items are being maintained under the CM  
3865 system.

3866 **12.2.8.3.16 ALC\_CMC.5.16C**

3867 The evidence shall demonstrate that the CM system is being operated in accordance with the CM plan.

3868 **12.2.8.4 Evaluator action elements**

3869 **12.2.8.4.1 ALC\_CMC.5.1E**

3870 The evaluator shall confirm that the information provided meets all requirements for content and  
3871 presentation of evidence.

3872 **12.2.8.4.2 ALC\_CMC.5.2E**

3873 **The evaluator shall determine that the application of the production support procedures**  
3874 **results in a TOE as provided by the developer for testing activities.**

3875 **12.3 CM scope (ALC\_CMS)**

3876 **12.3.1 Objectives**

3877 The objective of this family is to identify items to be included as configuration items and hence placed  
3878 under the CM requirements of CM capabilities (ALC\_CMC). Applying configuration management to  
3879 these additional items provides additional assurance that the integrity of TOE is maintained.

3880 **12.3.2 Component levelling**

3881 The components in this family are levelled on the basis of which of the following are required to be  
3882 included as configuration items: the TOE and the evaluation evidence required by the SARs; the parts  
3883 of the TOE; the implementation representation; security flaws; and development tools and related  
3884 information.

3885 **12.3.3 Application notes**

3886 While CM scope (ALC\_CMS) mandates a list of configuration items and that each item on this list be  
3887 under CM, CM capabilities (ALC\_CMC) leaves the contents of the configuration list to the discretion of  
3888 the developer. CM scope (ALC\_CMS) narrows this discretion by identifying items that must be included  
3889 in the configuration list, and hence come under the CM requirements of CM capabilities (ALC\_CMC).



3890 **12.3.4 ALC\_CMS.1 TOE CM coverage**

3891 Dependencies: No dependencies.

3892 **12.3.4.1 Objectives**

3893 A CM system can control changes only to those items that have been placed under CM (i.e., the  
3894 configuration items identified in the configuration list). Placing the TOE itself and the evaluation  
3895 evidence required by the other SARs in the ST under CM provides assurance that they have been  
3896 modified in a controlled manner with proper authorisations.

3897 **12.3.4.2 Application notes**

3898 ALC\_CMS.1.1C introduces the requirement that the TOE itself and the evaluation evidence required by  
3899 the other SARs in the ST be included in the configuration list and hence be subject to the CM  
3900 requirements of CM capabilities (ALC\_CMC).

3901 **12.3.4.3 Developer action elements**

3902 **12.3.4.3.1 ALC\_CMS.1.1D**

3903 **The developer shall provide a configuration list for the TOE.**

3904 **12.3.4.4 Content and presentation elements**

3905 **12.3.4.4.1 ALC\_CMS.1.1C**

3906 **The configuration list shall include the following: the TOE itself; and the evaluation evidence**  
3907 **required by the SARs.**

3908 **12.3.4.4.2 ALC\_CMS.1.2C**

3909 **The configuration list shall uniquely identify the configuration items.**

3910 **12.3.4.5 Evaluator action elements**

3911 **12.3.4.5.1 ALC\_CMS.1.1E**

3912 **The evaluator shall confirm that the information provided meets all requirements for content**  
3913 **and presentation of evidence.**

3914 **12.3.5 ALC\_CMS.2 Parts of the TOE CM coverage**

3915 Dependencies: No dependencies.

3916 **12.3.5.1 Objectives**

3917 A CM system can control changes only to those items that have been placed under CM (i.e., the  
3918 configuration items identified in the configuration list). Placing the TOE itself, the parts that comprise  
3919 the TOE, and the evaluation evidence required by the other SARs under CM provides assurance that  
3920 they have been modified in a controlled manner with proper authorisations.

3921 **12.3.5.2 Application notes**

3922 ALC\_CMS.2.1C introduces the requirement that the parts that comprise the TOE (all parts that are  
3923 delivered to the consumer, for example hardware parts or executable files) be included in the  
3924 configuration list and hence be subject to the CM requirements of CM capabilities (ALC\_CMC).

3925 ALC\_CMS.2.3C introduces the requirement that the configuration list indicate the developer of each  
3926 TSF relevant configuration item.

3927 **12.3.5.3 Developer action elements**

3928 **12.3.5.3.1 ALC\_CMS.2.1D**

3929 The developer shall provide a configuration list for the TOE.

3930 **12.3.5.4 Content and presentation elements**

3931 **12.3.5.4.1 ALC\_CMS.2.1C**

3932 The configuration list shall include the following: the TOE itself; the evaluation evidence required by  
3933 the SARs; **and the parts that comprise the TOE.**

3934 **12.3.5.4.2 ALC\_CMS.2.2C**

3935 The configuration list shall uniquely identify the configuration items.

3936 **12.3.5.4.3 ALC\_CMS.2.3C**

3937 **For each TSF relevant configuration item, the configuration list shall indicate the developer of**  
3938 **the item.**

3939 **12.3.5.5 Evaluator action elements**

3940 **12.3.5.5.1 ALC\_CMS.2.1E**

3941 The evaluator shall confirm that the information provided meets all requirements for content and  
3942 presentation of evidence.

3943 **12.3.6 ALC\_CMS.3 Implementation representation CM coverage**

3944 Dependencies: No dependencies.

3945 **12.3.6.1 Objectives**

3946 A CM system can control changes only to those items that have been placed under CM (i.e., the  
3947 configuration items identified in the configuration list). Placing the TOE itself, the parts that comprise  
3948 the TOE, the TOE implementation representation and the evaluation evidence required by the other  
3949 SARs under CM provides assurance that they have been modified in a controlled manner with proper  
3950 authorisations.

3951 **12.3.6.2 Application notes**

3952 ALC\_CMS.3.1C introduces the requirement that the TOE implementation representation be included in  
3953 the list of configuration items and hence be subject to the CM requirements of CM capabilities  
3954 (ALC\_CMC).

3955 **12.3.6.3 Developer action elements**

3956 **12.3.6.3.1 ALC\_CMS.3.1D**

3957 The developer shall provide a configuration list for the TOE.

3958 **12.3.6.4 Content and presentation elements**

3959 **12.3.6.4.1 ALC\_CMS.3.1C**

3960 The configuration list shall include the following: the TOE itself; the evaluation evidence required by  
3961 the SARs; the parts that comprise the TOE; **and the implementation representation.**

3962 **12.3.6.4.2 ALC\_CMS.3.2C**

3963 The configuration list shall uniquely identify the configuration items.

3964 **12.3.6.4.3 ALC\_CMS.3.3C**

3965 For each TSF relevant configuration item, the configuration list shall indicate the developer of the item.

3966 **12.3.6.5 Evaluator action elements**

3967 **12.3.6.5.1 ALC\_CMS.3.1E**

3968 The evaluator shall confirm that the information provided meets all requirements for content and  
3969 presentation of evidence.

3970 **12.3.7 ALC\_CMS.4 Problem tracking CM coverage**

3971 Dependencies: No dependencies.

3972 **12.3.7.1 Objectives**

3973 A CM system can control changes only to those items that have been placed under CM (i.e., the  
3974 configuration items identified in the configuration list). Placing the TOE itself, the parts that comprise  
3975 the TOE, the TOE implementation representation and the evaluation evidence required by the other  
3976 SARs under CM provides assurance that they have been modified in a controlled manner with proper  
3977 authorisations.

3978 Placing security flaw reports under CM ensures that the integrity of the reports is maintained and that  
3979 access to them is managed, further, it may support developers in tracking security flaws to their  
3980 resolution.

3981 **12.3.7.2 Application notes**

3982 ALC\_CMS.4.1C introduces the requirement that reports of identified security flaws be included in the  
3983 configuration list and hence be subject to the CM requirements of CM capabilities (ALC\_CMC). This  
3984 requires that information regarding previously identified security flaw reports and their resolution be  
3985 maintained.

3986 **12.3.7.3 Developer action elements**

3987 **12.3.7.3.1 ALC\_CMS.4.1D**

3988 The developer shall provide a configuration list for the TOE.

3989 **12.3.7.4 Content and presentation elements**

3990 **12.3.7.4.1 ALC\_CMS.4.1C**

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; the implementation representation; **and security flaw**  
3993 **reports and resolution status.**

3994 **12.3.7.4.2 ALC\_CMS.4.2C**

3995 The configuration list shall uniquely identify the configuration items.

3996 **12.3.7.4.3 ALC\_CMS.4.3C**

3997 For each TSF relevant configuration item, the configuration list shall indicate the developer of the item.

3998 **12.3.7.5 Evaluator action elements**

3999 **12.3.7.5.1 ALC\_CMS.4.1E**

4000 The evaluator shall confirm that the information provided meets all requirements for content and  
4001 presentation of evidence.

4002 **12.3.8 ALC\_CMS.5 Development tools CM coverage**

4003 Dependencies: No dependencies.

4004 **12.3.8.1 Objectives**

4005 A CM system can control changes only to those items that have been placed under CM (i.e., the  
4006 configuration items identified in the configuration list). Placing the TOE itself, the parts that comprise  
4007 the TOE, the TOE implementation representation and the evaluation evidence required by the other  
4008 SARs under CM provides assurance that they have been modified in a controlled manner with proper  
4009 authorisations.

4010 Placing security flaw reports under CM ensures that the integrity of the reports is maintained and that  
4011 access to them is managed, further, it may support developers in tracking security flaws to their  
4012 resolution.

4013 Development tools play an important role in ensuring the production of a quality version of the TOE.  
4014 Therefore, it is important to control modifications to these tools.

4015 **12.3.8.2 Application notes**

4016 ALC\_CMS.5.1C introduces the requirement that development tools and other related information be  
4017 included in the list of configuration items and hence be subject to the CM requirements of CM  
4018 capabilities (ALC\_CMC). Examples of development tools are programming languages and compilers.  
4019 Information pertaining to TOE generation items (such as compiler options, generation options, and  
4020 build options) is an example of information relating to development tools.

4021 **12.3.8.3 Developer action elements**

4022 **12.3.8.3.1 ALC\_CMS.5.1D**

4023 The developer shall provide a configuration list for the TOE.

4024 **12.3.8.4 Content and presentation elements**

4025 **12.3.8.4.1 ALC\_CMS.5.1C**

4026 The configuration list shall include the following: the TOE itself; the evaluation evidence required by  
4027 the SARs; the parts that comprise the TOE; the implementation representation; security flaw reports  
4028 and resolution status; **and development tools and related information.**

4029 **12.3.8.4.2 ALC\_CMS.5.2C**

4030 The configuration list shall uniquely identify the configuration items.

4031 **12.3.8.4.3 ALC\_CMS.5.3C**

4032 For each TSF relevant configuration item, the configuration list shall indicate the developer of the item.

4033 **12.3.8.5 Evaluator action elements**

4034 **12.3.8.5.1 ALC\_CMS.5.1E**

4035 The evaluator shall confirm that the information provided meets all requirements for content and  
4036 presentation of evidence.

4037 **12.4 Delivery (ALC\_DEL)**

4038 **12.4.1 Objectives**

4039 The concern of this family is the secure transfer of the finished TOE from the development  
4040 environment into the responsibility of the user.

4041 The requirements for delivery call for system control and distribution facilities and procedures that  
4042 detail the controls necessary to provide assurance that the security of the TOE is maintained during  
4043 distribution of the TOE to the user. For a valid distribution of the TOE, the procedures used for the  
4044 distribution of the TOE address the implied or identified objectives identified in the PP/ST relating to  
4045 the security of the TOE during delivery.

4046 **12.4.2 Component levelling**

4047 This family contains only one component. An increasing level of protection for the TOE is established  
4048 by requiring that the delivery procedures are commensurate with the assumed attack potential in the  
4049 family Vulnerability analysis (AVA\_VAN) specified in the ST.

4050 **12.4.3 Application notes**

4051 Transfers from subcontractors to the developer or between different development sites are not  
4052 considered here, but in the family [Developer environment security \(ALC\\_DVS\)](#).

4053 The end of the delivery phase is marked by the acceptance of the transfer of the TOE into the  
4054 responsibility of the downstream user.

4055 NOTE: This does not necessarily coincide with the arrival of the TOE at the downstream user's  
4056 location.

4057 The delivery procedures should consider, if applicable, issues such as:

- 4058 a) ensuring that the TOE received by the consumer corresponds precisely to the evaluated version of  
4059 the TOE;
- 4060 b) avoiding or detecting any tampering with the actual version of the TOE;
- 4061 c) preventing submission of a counterfeit version of the TOE;
- 4062 d) avoiding unwanted knowledge of distribution of the TOE to the consumer: there might be cases  
4063 where potential attackers should not know when and how it is delivered;
- 4064 e) avoiding or detecting the TOE being intercepted during delivery; and
- 4065 f) avoiding the TOE being delayed or stopped during distribution.

4066 The delivery procedures should include the recipient's actions implied by these issues. The consistent  
4067 description of these implied actions is examined in the Preparative procedures (AGD\_PRE) family, if  
4068 present.

#### 4069 **12.4.4 ALC\_DEL.1 Delivery procedures**

4070 Dependencies: No dependencies.

##### 4071 **12.4.4.1 Developer action elements**

###### 4072 **12.4.4.1.1 ALC\_DEL.1.1D**

4073 **The developer shall document and provide procedures for delivery of the TOE or parts of it to**  
4074 **the consumer.**

###### 4075 **12.4.4.1.2 ALC\_DEL.1.2D**

4076 **The developer shall use the delivery procedures.**

##### 4077 **12.4.4.2 Content and presentation elements**

###### 4078 **12.4.4.2.1 ALC\_DEL.1.1C**

4079 **The delivery documentation shall describe all procedures that are necessary to maintain**  
4080 **security when distributing versions of the TOE to the consumer.**

##### 4081 **12.4.4.3 Evaluator action elements**

###### 4082 **12.4.4.3.1 ALC\_DEL.1.1E**

4083 **The evaluator shall confirm that the information provided meets all requirements for content**  
4084 **and presentation of evidence.**

4085 **12.5 Developer environment security (ALC\_DVS)**

4086 **12.5.1 Objectives**

4087 Development security is concerned with the determination and specification of security controls  
4088 relating to the developer provided environment.

4089 NOTE: Such controls include coverage of security relevant aspects of asset management, human  
4090 resources security, physical and environmental security, communications and operations  
4091 management, access control, information systems acquisition, development and maintenance,  
4092 information security incident management, and business continuity management.

4093 **12.5.2 Component levelling**

4094 The components in this family are levelled on the basis of whether justification of the sufficiency of the  
4095 security controls is required.

4096 **12.5.3 Application notes**

4097 This family deals with controls to remove or reduce threads and security risks existing at the  
4098 developer's site.

4099 The evaluator should visit the site(s) in order to assess evidence for development security. This may  
4100 include sites of subcontractors involved in the TOE development and production. Any decision not to  
4101 visit shall be agreed with the evaluation authority.

4102 Although development security deals with the maintenance of the TOE and hence with aspects  
4103 becoming relevant after the completion of the evaluation, the [Developer environment security](#)  
4104 [\(ALC\\_DVS\)](#) requirements specify only that the development security controls be in place at the time of  
4105 evaluation. Furthermore, [Developer environment security \(ALC\\_DVS\)](#) does not contain any  
4106 requirements related to the sponsor's intention to apply the development security controls in the  
4107 future, after completion of the evaluation.

4108 It is recognised that confidentiality may not always be an issue for the protection of the TOE in its  
4109 development environment. The use of the word "necessary" allows for the selection of appropriate  
4110 safeguards.

4111 **12.5.4 ALC\_DVS.1 Identification of security controls**

4112 Dependencies: No dependencies.

4113 **12.5.4.1 Developer action elements**

4114 **12.5.4.1.1 ALC\_DVS.1.1D**

4115 **The developer shall produce and provide development security documentation.**

4116 **12.5.4.2 Content and presentation elements**

4117 **12.5.4.2.1 ALC\_DVS.1.1C**

4118 **The development security documentation shall describe all the physical, logical, procedural,**  
4119 **personnel, and other security controls that are necessary to protect the confidentiality and**  
4120 **integrity of the TOE design and implementation in its development environment.**

4121	<b>12.5.4.3 Evaluator action elements</b>
4122	<b>12.5.4.3.1 ALC_DVS.1.1E</b>
4123	<b>The evaluator shall confirm that the information provided meets all requirements for content</b>
4124	<b>and presentation of evidence.</b>
4125	<b>12.5.4.3.2 ALC_DVS.1.2E</b>
4126	<b>The evaluator shall confirm that the security controls are being applied.</b>
4127	<b>12.5.5 ALC_DVS.2 Sufficiency of security controls</b>
4128	Dependencies: No dependencies.
4129	<b>12.5.5.1 Developer action elements</b>
4130	<b>12.5.5.1.1 ALC_DVS.2.1D</b>
4131	The developer shall produce and provide development security documentation.
4132	<b>12.5.5.2 Content and presentation elements</b>
4133	<b>12.5.5.2.1 ALC_DVS.2.1C</b>
4134	The development security documentation shall describe all the physical, procedural, personnel, and
4135	other security controls that are necessary to protect the confidentiality and integrity of the TOE design
4136	and implementation in its development environment.
4137	<b>12.5.5.2.2 ALC_DVS.2.2C</b>
4138	<b>The development security documentation shall justify that the security controls provide the</b>
4139	<b>necessary level of protection to maintain the confidentiality and integrity of the TOE.</b>
4140	<b>12.5.5.3 Evaluator action elements</b>
4141	<b>12.5.5.3.1 ALC_DVS.2.1E</b>
4142	The evaluator shall confirm that the information provided meets all requirements for content and
4143	presentation of evidence.
4144	<b>12.5.5.3.2 ALC_DVS.2.2E</b>
4145	The evaluator shall confirm that the security controls are being applied.
4146	<b>12.6 Flaw remediation (ALC_FLR)</b>
4147	<b>12.6.1 Objectives</b>
4148	Flaw remediation requires that discovered security flaws be tracked and corrected by the developer.
4149	Although future compliance with flaw remediation procedures cannot be determined at the time of the
4150	TOE evaluation, it is possible to evaluate the policies and procedures that a developer has in place to
4151	track and correct flaws, and to distribute the flaw information and corrections.



**12.6.2 Component levelling**

The components in this family are levelled on the basis of the increasing extent in scope of the flaw remediation procedures and the rigour of the flaw remediation policies.

**12.6.3 Application notes**

This family provides assurance that the TOE will be maintained and supported in the future, requiring the TOE developer to track and correct flaws in the TOE. Additionally, requirements are included for the distribution of flaw corrections. However, this family does not impose evaluation requirements beyond the current evaluation.

The TOE user is considered to be the focal point in the user organisation that is responsible for receiving and implementing fixes to security flaws. This is not necessarily an individual user, but may be an organisational representative who is responsible for the handling of security flaws. The use of the term TOE user recognises that different organisations have different procedures for handling flaw reporting, which may be done either by an individual user, or by a central administrative body.

The flaw remediation procedures should describe the methods for dealing with all types of flaws encountered. These flaws may be reported by the developer, by users of the TOE, or by other parties with familiarity with the TOE. Some flaws may not be reparable immediately. There may be some occasions where a flaw cannot be fixed and other (e.g. procedural) controls must be taken. The documentation provided should cover the procedures for providing the operational sites with fixes, and providing information on flaws where fixes are delayed (and what to do in the interim) or when fixes are not possible.

Changes applied to a TOE after its release render it unevaluated; although some information from the original evaluation may still apply. The phrase "release of the TOE" used in this family therefore refers to a version of a product that is a release of a certified TOE, to which changes have been applied.

**12.6.4 ALC\_FLR.1 Basic flaw remediation**

Dependencies: No dependencies.

**12.6.4.1 Developer action elements****12.6.4.1.1 ALC\_FLR.1.1D**

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

**12.6.4.2 Content and presentation elements****12.6.4.2.1 ALC\_FLR.1.1C**

The flaw remediation procedures documentation shall describe the procedures used to track all reported security flaws in each release of the TOE.

**12.6.4.2.2 ALC\_FLR.1.2C**

The flaw remediation procedures shall require that a description of the nature and effect of each security flaw be provided, as well as the status of finding a correction to that flaw.

4188	<b>12.6.4.2.3 ALC_FLR.1.3C</b>
4189	<b>The flaw remediation procedures shall require that corrective actions be identified for each of</b>
4190	<b>the security flaws.</b>
4191	<b>12.6.4.2.4 ALC_FLR.1.4C</b>
4192	<b>The flaw remediation procedures documentation shall describe the methods used to provide</b>
4193	<b>flaw information, corrections and guidance on corrective actions to TOE users.</b>
4194	<b>12.6.4.3 Evaluator action elements</b>
4195	<b>12.6.4.3.1 ALC_FLR.1.1E</b>
4196	<b>The evaluator shall confirm that the information provided meets all requirements for content</b>
4197	<b>and presentation of evidence.</b>
4198	<b>12.6.5 ALC_FLR.2 Flaw reporting procedures</b>
4199	Dependencies: No dependencies.
4200	<b>12.6.5.1 Objectives</b>
4201	In order for the developer to be able to act appropriately upon security flaw reports from TOE users,
4202	and to know to whom to send corrective fixes, TOE users need to understand how to submit security
4203	flaw reports to the developer. Flaw remediation guidance from the developer to the TOE user ensures
4204	that TOE users are aware of this important information.
4205	<b>12.6.5.2 Developer action elements</b>
4206	<b>12.6.5.2.1 ALC_FLR.2.1D</b>
4207	The developer shall document and provide flaw remediation procedures addressed to TOE developers.
4208	<b>12.6.5.2.2 ALC_FLR.2.2D</b>
4209	<b>The developer shall establish a procedure for accepting and acting upon all reports of security</b>
4210	<b>flaws and requests for corrections to those flaws.</b>
4211	<b>12.6.5.2.3 ALC_FLR.2.3D</b>
4212	<b>The developer shall provide flaw remediation guidance addressed to TOE users.</b>
4213	<b>12.6.5.3 Content and presentation elements</b>
4214	<b>12.6.5.3.1 ALC_FLR.2.1C</b>
4215	The flaw remediation procedures documentation shall describe the procedures used to track all
4216	reported security flaws in each release of the TOE.
4217	<b>12.6.5.3.2 ALC_FLR.2.2C</b>
4218	The flaw remediation procedures shall require that a description of the nature and effect of each
4219	security flaw be provided, as well as the status of finding a correction to that flaw.

4220 **12.6.5.3.3 ALC\_FLR.2.3C**

4221 The flaw remediation procedures shall require that corrective actions be identified for each of the  
4222 security flaws.

4223 **12.6.5.3.4 ALC\_FLR.2.4C**

4224 The flaw remediation procedures documentation shall describe the methods used to provide flaw  
4225 information, corrections and guidance on corrective actions to TOE users.

4226 **12.6.5.3.5 ALC\_FLR.2.5C**

4227 **The flaw remediation procedures shall describe a means by which the developer receives from**  
4228 **TOE users reports and enquiries of suspected security flaws in the TOE.**

4229 **12.6.5.3.6 ALC\_FLR.2.6C**

4230 **The procedures for processing reported security flaws shall ensure that any reported flaws are**  
4231 **remediated and the remediation procedures issued to TOE users.**

4232 **12.6.5.3.7 ALC\_FLR.2.7C**

4233 **The procedures for processing reported security flaws shall provide safeguards that any**  
4234 **corrections to these security flaws do not introduce any new flaws.**

4235 **12.6.5.3.8 ALC\_FLR.2.8C**

4236 **The flaw remediation guidance shall describe a means by which TOE users report to the**  
4237 **developer any suspected security flaws in the TOE.**

4238 **12.6.5.4 Evaluator action elements**

4239 **12.6.5.4.1 ALC\_FLR.2.1E**

4240 The evaluator shall confirm that the information provided meets all requirements for content and  
4241 presentation of evidence.

4242 **12.6.6 ALC\_FLR.3 Systematic flaw remediation**

4243 Dependencies: No dependencies.

4244 **12.6.6.1 Objectives**

4245 In order for the developer to be able to act appropriately upon security flaw reports from TOE users,  
4246 and to know to whom to send corrective fixes, TOE users need to understand how to submit security  
4247 flaw reports to the developer, and how to register themselves with the developer so that they may  
4248 receive these corrective fixes. Flaw remediation guidance from the developer to the TOE user ensures  
4249 that TOE users are aware of this important information.

4250 **12.6.6.2 Developer action elements**

4251 **12.6.6.2.1 ALC\_FLR.3.1D**

4252 The developer shall document and provide flaw remediation procedures addressed to TOE developers.

4253 **12.6.6.2.2 ALC\_FLR.3.2D**

4254 The developer shall establish a procedure for accepting and acting upon all reports of security flaws  
4255 and requests for corrections to those flaws.

4256 **12.6.6.2.3 ALC\_FLR.3.3D**

4257 The developer shall provide flaw remediation guidance addressed to TOE users.

4258 **12.6.6.3 Content and presentation elements**

4259 **12.6.6.3.1 ALC\_FLR.3.1C**

4260 The flaw remediation procedures documentation shall describe the procedures used to track all  
4261 reported security flaws in each release of the TOE.

4262 **12.6.6.3.2 ALC\_FLR.3.2C**

4263 The flaw remediation procedures shall require that a description of the nature and effect of each  
4264 security flaw be provided, as well as the status of finding a correction to that flaw.

4265 **12.6.6.3.3 ALC\_FLR.3.3C**

4266 The flaw remediation procedures shall require that corrective actions be identified for each of the  
4267 security flaws.

4268 **12.6.6.3.4 ALC\_FLR.3.4C**

4269 The flaw remediation procedures documentation shall describe the methods used to provide flaw  
4270 information, corrections and guidance on corrective actions to TOE users.

4271 **12.6.6.3.5 ALC\_FLR.3.5C**

4272 The flaw remediation procedures shall describe a means by which the developer receives from TOE  
4273 users reports and enquiries of suspected security flaws in the TOE.

4274 **12.6.6.3.6 ALC\_FLR.3.6C**

4275 **The flaw remediation procedures shall include a procedure requiring timely response and the**  
4276 **automatic distribution of security flaw reports and the associated corrections to registered**  
4277 **users who might be affected by the security flaw.**

4278 **12.6.6.3.7 ALC\_FLR.3.7C**

4279 The procedures for processing reported security flaws shall ensure that any reported flaws are  
4280 remediated and the remediation procedures issued to TOE users.

4281 **12.6.6.3.8 ALC\_FLR.3.8C**

4282 The procedures for processing reported security flaws shall provide safeguards that any corrections to  
4283 these security flaws do not introduce any new flaws.

4284 **12.6.6.3.9 ALC\_FLR.3.9C**

4285 The flaw remediation guidance shall describe a means by which TOE users report to the developer any  
4286 suspected security flaws in the TOE.

4287 **12.6.6.3.10 ALC\_FLR.3.10C**

4288 **The flaw remediation guidance shall describe a means by which TOE users may register with**  
4289 **the developer, to be eligible to receive security flaw reports and corrections.**

4290 **12.6.6.3.11 ALC\_FLR.3.11C**

4291 **The flaw remediation guidance shall identify the specific points of contact for all reports and**  
4292 **enquiries about security issues involving the TOE.**

4293 **12.6.6.4 Evaluator action elements**

4294 **12.6.6.4.1 ALC\_FLR.3.1E**

4295 The evaluator shall confirm that the information provided meets all requirements for content and  
4296 presentation of evidence.

4297 **12.7 Development Life-cycle definition (ALC\_LCD)**

4298 **12.7.1 Objectives**

4299 Poorly defined or uncontrolled processes applied during the development, production and  
4300 maintenance of the TOE can result in a TOE that does not meet all of its security objectives. Therefore,  
4301 it is important that well defined and controlled processes be established as early as possible in the  
4302 TOE's life-cycle.

4303 Defining and implementing such processes does not guarantee that the TOE meets all of its SFRs. It is  
4304 possible that the processes will be insufficient or inadequate.

4305 Adopting a life-cycle model, or models that meets the needs of the developer's organization will  
4306 improve the likelihood that the development, production and maintenance processes applied to TOE  
4307 support the correct design and implementation of a TOE that meets the specified SFRs.

4308 The determination of appropriate process controls in order to support process improvement is a long  
4309 established best practice.

4310 **12.7.2 Component levelling**

4311 The components in this family are levelled on the basis of increasing requirements for measurability of  
4312 the life-cycle model, and for compliance with that model.

4313 **12.7.3 Application notes**

4314 A life-cycle model encompasses the procedures, tools and techniques used to develop and maintain the  
4315 TOE. Aspects of the process that may be covered by such a model include design methods, review  
4316 procedures, project management controls, change control procedures, test methods and acceptance  
4317 procedures. An effective life-cycle model will address these aspects of the development and  
4318 maintenance process within an overall management structure that assigns responsibilities and  
4319 monitors progress.

4320 There are different types of acceptance situations that are dealt with at different locations in the  
4321 criteria:

- 4322 • acceptance of parts delivered by subcontractors (“integration”) should be treated in this family
- 4323 • Development Life-cycle definition (ALC\_LCD)~~Life-cycle definition (ALC\_LCD)~~,
- 4324 • acceptance subsequent to internal transportations in Developer environment security  
4325 (ALC\_DVS)~~Development security (ALC\_DVS)~~,
- 4326 • acceptance of parts into the CM system in CM capabilities (ALC\_CMC), and
- 4327 • acceptance of the delivered TOE by the consumer in Delivery (ALC\_DEL).

4328 The first three types may overlap.

4329 Although life-cycle definition deals with the maintenance of the TOE and hence with aspects becoming  
4330 relevant after the completion of the evaluation, its evaluation adds assurance through an analysis of  
4331 the life-cycle information for the TOE provided at the time of the evaluation.

4332 A life-cycle model provides for the necessary control over the development and maintenance of the  
4333 TOE, if the model enables sufficient minimisation of the danger that the TOE will not meet its security  
4334 requirement.

4335 A measurable life-cycle model is a model using some quantitative valuation (arithmetic parameters  
4336 and/or metrics) of the managed product in order to measure development properties of the product.  
4337 Typical metrics are source code complexity metrics, defect density (errors per size of code) or mean  
4338 time to failure. For the security evaluation all those metrics are of relevance, which are used to  
4339 increase quality by decreasing the probability of faults and thereby in turn increasing assurance in the  
4340 security of the TOE.

4341 One should take into account that there exist standardised life-cycle models on the one hand (like the  
4342 waterfall model) and standardised metrics on the other hand (like error density), which may be  
4343 combined. The ISO/IEC 15408 series does not require the life-cycle to follow exactly one standard  
4344 defining both aspects.

#### 4345 **12.7.4 ALC\_LCD.1 Developer defined life-cycle processes**

4346 Dependencies: No dependencies.

##### 4347 **12.7.4.1 Developer action elements**

###### 4348 **12.7.4.1.1 ALC\_LCD.1.1D**

4349 **The developer shall establish a life-cycle model to be used in the development and**  
4350 **maintenance of the TOE.**

###### 4351 **12.7.4.1.2 ALC\_LCD.1.2D**

4352 **The developer shall provide life-cycle definition documentation.**

- 4353 **12.7.4.2 Content and presentation elements**
- 4354 **12.7.4.2.1 ALC\_LCD.1.1C**
- 4355 **The life-cycle definition documentation shall describe the processes used to develop and**  
4356 **maintain the TOE.**
- 4357 **12.7.4.2.2 ALC\_LCD.1.2C**
- 4358 **The life-cycle model shall provide for the necessary control over the development and**  
4359 **maintenance of the TOE.**
- 4360 **12.7.4.3 Evaluator action elements**
- 4361 **12.7.4.3.1 ALC\_LCD.1.1E**
- 4362 **The evaluator shall confirm that the information provided meets all requirements for content**  
4363 **and presentation of evidence.**
- 4364 **12.7.5 ALC\_LCD.2 Measurable life-cycle model**
- 4365 Dependencies: No dependencies.
- 4366 **12.7.5.1 Developer action elements**
- 4367 **12.7.5.1.1 ALC\_LCD.2.1D**
- 4368 **The developer shall establish a life-cycle model to be used in the development and maintenance of the**  
4369 **TOE that is based on a measurable life-cycle model.**
- 4370 **12.7.5.1.2 ALC\_LCD.2.2D**
- 4371 **The developer shall provide life-cycle definition documentation.**
- 4372 **12.7.5.1.3 ALC\_LCD.2.3D**
- 4373 **The developer shall measure the TOE development using the measurable life-cycle model..**
- 4374 **12.7.5.1.4 ALC\_LCD.2.4D**
- 4375 **The developer shall provide life-cycle output documentation.**
- 4376 **12.7.5.2 Content and presentation elements**
- 4377 **12.7.5.2.1 ALC\_LCD.2.1C**
- 4378 **The life-cycle definition documentation shall describe the model used to develop and maintain the TOE**  
4379 **including the details of its arithmetic parameters and/or metrics used to measure the quality of**  
4380 **the TOE and/or its development.**
- 4381 **12.7.5.2.2 ALC\_LCD.2.2C**
- 4382 **The life-cycle model shall provide for the necessary control over the development and maintenance of**  
4383 **the TOE.**

4384 **12.7.5.2.3 ALC\_LCD.2.3C**

4385 **The life-cycle output documentation shall provide the results of the measurements of the TOE**  
 4386 **development using the measurable life-cycle model.**

4387 **12.7.5.3 Evaluator action elements**4388 **12.7.5.3.1 ALC\_LCD.2.1E**

4389 The evaluator shall confirm that the information provided meets all requirements for content and  
 4390 presentation of evidence.

4391 **12.7.5.3.2 ALC\_LCD.2.2E**

4392 **The evaluator shall confirm that the measurements of the TOE development processes and**  
 4393 **security relevant properties of the TOE support improvements in the development processes**  
 4394 **and/or the TOE itself.**

4395 **12.8 TOE Development Artifacts (ALC\_TDA)**4396 **12.8.1 Objectives**

4397 This family aims to add trust to the development process or a development. It focuses on the  
 4398 generation of certain artifacts in the development process. These artifacts are used at a later point in  
 4399 time to assess the degree to which the development process is trustable. This trust is realized through  
 4400 the validation of the generated artifacts for confirming them as sufficient evidence for trustable  
 4401 development.

4402 This family introduces developer practices within the development process to generate the required  
 4403 artifacts for realizing trustable development. If a requirement in this family does not explicitly specify  
 4404 the use of automation to generate the required artifacts, the developer is free to undertake the  
 4405 corresponding practice manually, or to use some integrated automation in the development process,  
 4406 or to use a hybrid method of both. It is expected that the degree of trust in the development process is  
 4407 proportional to the degree of automation adoption to implement the corresponding practice in the  
 4408 development process.

4409 This family also has a relationship with the ALC\_TAT family. As ALC\_TAT focuses on the tools and  
 4410 techniques aspect for developing, analysing, and implementing the TOE, it provides the necessary  
 4411 context when describing the practices of this family being introduced into the development process.

4412 **12.8.2 Component levelling**

4413 The components in this family are levelled on the basis of increasing cross-checking for consistency  
 4414 with relevant evidence from components of other families of other security assurance classes.

4415 **12.8.3 Application notes**

4416 The requirements in ALC\_TDA.1 provide a degree of trust in the developer's ability to identify the set  
 4417 of implementation representation which actually has been used during the TOE generation time. This  
 4418 degree of trust helps to positively answer the question "is that really the source code for this software"  
 4419 or "is that really the register-transfer level (RTL) design or description for this integrated circuit  
 4420 hardware"" or "is that really the set of implementation representation for this TOE", which is  
 4421 potentially relevant in an evaluation. Such degree of trust is built on



- 4422 a) the timing of when the set of implementation representation identifiers is recorded or logged,
- 4423 b) the integrity and authenticity of the record of implementation representation identifiers, and
- 4424 c) the traceability of implementation representation identifiers from the TOE.

4425 In the case where some implementation representation elements are also covered in the configuration  
 4426 list due to ALC\_CMS.3, the requirements in ALC\_TDA.2 make sure that these implementation  
 4427 representation elements actually are identifiable through the use of the implementation  
 4428 representation identifiers of ALC\_TDA.1.

4429 With the accurate recording or logging of the actual implementation representation being used by the  
 4430 development tools under the scope of ALC\_TAT, it provides an additional evidence to convince a third  
 4431 party that a regeneration of the TOE is functionally equivalent to the original TOE.

4432 The requirements in ALC\_TDA.3 provide the developer an opportunity to testify the absence of  
 4433 functional differences between the two possibly visibly different TOEs which have been independently  
 4434 generated from the identical set of implementation representation.

#### 4435 **12.8.4 ALC\_TDA.1 Uniquely identifying implementation representation**

4436 Dependencies: No dependencies.

##### 4437 **12.8.4.1 Developer action elements**

###### 4438 **12.8.4.1.1 ALC\_TDA.1.1D**

4439 **The developer shall identify individual elements of the TOE implementation representation to**  
 4440 **record the list of unique TOE implementation representation identifiers, as the development**  
 4441 **tool generates the TOE.**

###### 4442 **12.8.4.1.2 ALC\_TDA.1.2D**

4443 **The developer shall use the current date and time to timestamp the list of unique TOE**  
 4444 **implementation representation identifiers as recorded during the TOE generation time.**

###### 4445 **12.8.4.1.3 ALC\_TDA.1.3D**

4446 **The developer shall maintain the integrity of the list of unique TOE implementation**  
 4447 **representation identifiers as recorded during the TOE generation time.**

###### 4448 **12.8.4.1.4 ALC\_TDA.1.4D**

4449 **The developer shall ensure the authenticity of the list of unique TOE implementation**  
 4450 **representation identifiers as recorded during the TOE generation time, with the maintenance**  
 4451 **of the (author) origination information.**

###### 4452 **12.8.4.1.5 ALC\_TDA.1.5D**

4453 **The developer shall be able to trace from the TOE to the list of unique TOE implementation**  
 4454 **representation identifiers as recorded during the TOE generation time.**

###### 4455 **12.8.4.1.6 ALC\_TDA.1.6D**

4456 **The developer shall produce and provide documentation describing**

- 4457 a) the developer's creation of the list of unique TOE implementation representation  
4458 identifiers as recorded during the TOE generation time;
- 4459 b) the developer's timestamp being applied to the list of unique TOE implementation  
4460 representation identifiers as recorded during the TOE generation time;
- 4461 c) the maintenance of the (author) origination information of the list of unique TOE  
4462 implementation representation identifiers as recorded during the TOE generation time;
- 4463 d) the maintenance of the integrity of the list of unique TOE implementation  
4464 representation identifiers as recorded during the TOE generation time and its  
4465 associated timestamp and (author) origination information;
- 4466 e) the developer's mechanism to trace from the TOE to the list of unique TOE  
4467 implementation representation identifiers as recorded during the TOE generation time.

#### 4468 12.8.4.2 Content and presentation elements

##### 4469 12.8.4.2.1 ALC\_TDA.1.1C

4470 The list of unique TOE implementation representation identifiers as recorded during the TOE  
4471 generation time shall demonstrate the correspondence between the TOE implementation  
4472 representation element identifiers and the TOE implementation representation element  
4473 names.

##### 4474 12.8.4.2.2 ALC\_TDA.1.2C

4475 The TOE implementation representation element names shall be in the same form as used or  
4476 referenced by the development tool to generate the TOE.

##### 4477 12.8.4.2.3 ALC\_TDA.1.3C

4478 The timestamp of the list of unique TOE implementation representation identifiers as recorded  
4479 during the TOE generation time shall be consistent with the creation time of the TOE.

##### 4480 12.8.4.2.4 ALC\_TDA.1.4C

4481 The (author) origination information of the list of unique TOE implementation representation  
4482 identifiers as recorded during the TOE generation time shall be consistent with the (author)  
4483 origination information of the TOE.

#### 4484 12.8.4.3 Evaluator action elements

##### 4485 12.8.4.3.1 ALC\_TDA.1.1E

4486 The evaluator shall confirm that the information provided meets all requirements for content  
4487 and presentation of evidence.

##### 4488 12.8.4.3.2 ALC\_TDA.1.2E

4489 The evaluator shall confirm that the development tool for generating the TOE is capable to use  
4490 or reference the implementation representation element names.

4491 **12.8.4.3.3 ALC\_TDA.1.3E**

4492 The evaluator shall confirm that the list of unique TOE implementation representation  
4493 identifiers as recorded during the TOE generation time is consistent with the creation time of  
4494 the TOE.

4495 **12.8.4.3.4 ALC\_TDA.1.4E**

4496 The evaluator shall confirm that the (author) origination information of the list of unique TOE  
4497 implementation representation identifiers as recorded during the TOE generation time is  
4498 consistent with the (author) origination information of the TOE.

4499 **12.8.4.3.5 ALC\_TDA.1.5E**

4500 The evaluator shall check the integrity of the list of unique TOE implementation representation  
4501 identifiers as recorded during the TOE generation time and its associated timestamp and  
4502 (author) origination information.

4503 **12.8.4.3.6 ALC\_TDA.1.6E**

4504 The evaluator shall confirm the developer's ability to trace from the TOE to the list of unique  
4505 TOE implementation representation identifiers as recorded during the TOE generation time.

4506 **12.8.5 ALC\_TDA.2 Matching CMS scope of implementation representation**

4507 Dependencies: ALC\_CMS.3 Implementation representation CM coverage

4508 **12.8.5.1 Developer action elements**

4509 **12.8.5.1.1 ALC\_TDA.2.1D**

4510 The developer shall identify individual elements of the TOE implementation representation to record  
4511 the list of unique TOE implementation representation identifiers, as the development tool generates  
4512 the TOE.

4513 **12.8.5.1.2 ALC\_TDA.2.2D**

4514 The developer shall use the current date and time to timestamp the list of unique TOE implementation  
4515 representation identifiers as recorded during the TOE generation time.

4516 **12.8.5.1.3 ALC\_TDA.2.3D**

4517 The developer shall maintain the integrity of the list of unique TOE implementation representation  
4518 identifiers as recorded during the TOE generation time.

4519 **12.8.5.1.4 ALC\_TDA.2.4D**

4520 The developer shall ensure the authenticity of the list of unique TOE implementation representation  
4521 identifiers as recorded during the TOE generation time, with the maintenance of the (author)  
4522 origination information.

4523 **12.8.5.1.5 ALC\_TDA.2.5D**

4524 The developer shall be able to trace from the TOE to the list of unique TOE implementation  
4525 representation identifiers as recorded during the TOE generation time.

4526 **12.8.5.1.6 ALC\_TDA.2.6D**

4527 The developer shall produce and provide documentation describing

- 4528 a) the developer's creation of the list of unique TOE implementation representation identifiers as  
 4529 recorded during the TOE generation time;
- 4530 b) the developer's timestamp being applied to the list of unique TOE implementation  
 4531 representation identifiers as recorded during the TOE generation time;
- 4532 c) the maintenance of the (author) origination information of the list of unique TOE  
 4533 implementation representation identifiers as recorded during the TOE generation time;
- 4534 d) the maintenance of the integrity of the list of unique TOE implementation representation  
 4535 identifiers as recorded during the TOE generation time and its associated timestamp and  
 4536 (author) origination information;
- 4537 e) the developer's mechanism to trace from the TOE to the list of unique TOE implementation  
 4538 representation identifiers as recorded during the TOE generation time.

4539 **12.8.5.1.7 ALC\_TDA.2.7D**

4540 **The developer shall provide evidence that the elements of implementation representation**  
 4541 **under the configuration scope of ALC\_CMS.3 are identified by the list of unique TOE**  
 4542 **implementation representation identifiers as recorded during the TOE generation time.**

4543 **12.8.5.2 Content and presentation elements**4544 **12.8.5.2.1 ALC\_TDA.2.1C**

4545 The list of unique TOE implementation representation identifiers as recorded during the TOE  
 4546 generation time shall demonstrate the correspondence between the TOE implementation  
 4547 representation element identifiers and the TOE implementation representation element names.

4548 **12.8.5.2.2 ALC\_TDA.2.2C**

4549 The TOE implementation representation element names shall be in the same form as used or  
 4550 referenced by the development tool to generate the TOE.

4551 **12.8.5.2.3 ALC\_TDA.2.3C**

4552 The timestamp of the list of unique TOE implementation representation identifiers as recorded during  
 4553 the TOE generation time shall be consistent with the creation time of the TOE.

4554 **12.8.5.2.4 ALC\_TDA.2.4C**

4555 The (author) origination information of the list of unique TOE implementation representation  
 4556 identifiers as recorded during the TOE generation time shall be consistent with the (author)  
 4557 origination information of the TOE.

4558 **12.8.5.2.5 ALC\_TDA.2.5C**

4559 **The list of identifiers of the elements of implementation representation under the**  
 4560 **configuration scope of ALC\_CMS.3 shall match with the list of unique TOE implementation**  
 4561 **representation identifiers as recorded during the TOE generation time.**

4562 **12.8.5.3 Evaluator action elements**

4563 **12.8.5.3.1 ALC\_TDA.2.1E**

4564 The evaluator shall confirm that the information provided meets all requirements for content and  
4565 presentation of evidence.

4566 **12.8.5.3.2 ALC\_TDA.2.2E**

4567 The evaluator shall confirm that the development tool for generating the TOE is capable to use or  
4568 reference the implementation representation element names.

4569 **12.8.5.3.3 ALC\_TDA.2.3E**

4570 The evaluator shall confirm that the list of unique TOE implementation representation identifiers as  
4571 recorded during the TOE generation time is consistent with the creation time of the TOE.

4572 **12.8.5.3.4 ALC\_TDA.2.4E**

4573 The evaluator shall confirm that the (author) origination information of the list of unique TOE  
4574 implementation representation identifiers as recorded during the TOE generation time is consistent  
4575 with the (author) origination information of the TOE.

4576 **12.8.5.3.5 ALC\_TDA.2.5E**

4577 The evaluator shall check the integrity of the list of unique TOE implementation representation  
4578 identifiers as recorded during the TOE generation time and its associated timestamp and (author)  
4579 origination information.

4580 **12.8.5.3.6 ALC\_TDA.2.6E**

4581 The evaluator shall confirm the developer's ability to trace from the TOE to the list of unique TOE  
4582 implementation representation identifiers as recorded during the TOE generation time.

4583 **12.8.5.3.7 ALC\_TDA.2.7E**

4584 **The evaluator shall confirm that the list of identifiers of the elements of implementation**  
4585 **representation under the configuration scope of ALC\_CMS.3 matches with the list of unique TOE**  
4586 **implementation representation identifiers as recorded during the TOE generation time.**

4587 **12.8.6 ALC\_TDA.3 Regenerate TOE with well-defined development tools**

4588 Dependencies: ALC\_CMS.3 Implementation representation CM coverage

4589 ALC\_TAT.1 Well-defined development tools and

4590 ADV\_IMP.1 Implementation representation of the TSF

4591 **12.8.6.1 Developer action elements**

4592 **12.8.6.1.1 ALC\_TDA.3.1D**

4593 The developer shall identify individual elements of the TOE implementation representation to record  
4594 the list of unique TOE implementation representation identifiers, as the development tool generates  
4595 the TOE.

4596 **12.8.6.1.2 ALC\_TDA.3.2D**

4597 The developer shall use the current date and time to timestamp the list of unique TOE implementation  
4598 representation identifiers as recorded during the TOE generation time.

4599 **12.8.6.1.3 ALC\_TDA.3.3D**

4600 The developer shall maintain the integrity of the list of unique TOE implementation representation  
4601 identifiers as recorded during the TOE generation time.

4602 **12.8.6.1.4 ALC\_TDA.3.4D**

4603 The developer shall ensure the authenticity of the list of unique TOE implementation representation  
4604 identifiers as recorded during the TOE generation time, with the maintenance of the (author)  
4605 origination information.

4606 **12.8.6.1.5 ALC\_TDA.3.5D**

4607 The developer shall be able to trace from the TOE to the list of unique TOE implementation  
4608 representation identifiers as recorded during the TOE generation time.

4609 **12.8.6.1.6 ALC\_TDA.3.6D**

4610 The developer shall produce and provide documentation describing

- 4611 a) the developer's creation of the list of unique TOE implementation representation identifiers as  
4612 recorded during the TOE generation time;
- 4613 b) the developer's timestamp being applied to the list of unique TOE implementation  
4614 representation identifiers as recorded during the TOE generation time;
- 4615 c) the maintenance of the (author) origination information of the list of unique TOE  
4616 implementation representation identifiers as recorded during the TOE generation time;
- 4617 d) the maintenance of the integrity of the list of unique TOE implementation representation  
4618 identifiers as recorded during the TOE generation time and its associated timestamp and  
4619 (author) origination information;
- 4620 e) the developer's mechanism to trace from the TOE to the list of unique TOE implementation  
4621 representation identifiers as recorded during the TOE generation time.

4622 **12.8.6.1.7 ALC\_TDA.3.7D**

4623 The developer shall provide evidence that the elements of implementation representation under the  
4624 configuration scope of ALC\_CMS.3 are identified by the list of unique TOE implementation  
4625 representation identifiers as recorded during the TOE generation time.

4626 **12.8.6.1.8 ALC\_TDA.3.8D**

4627 **After applying the development tools to another copy of the TOE implementation**  
4628 **representation according to the list of unique TOE implementation representation identifiers**  
4629 **to regenerate a TOE copy, the developer shall explain the functional differences, if any, between**  
4630 **the TOE copy and the original TOE.**

4631 **12.8.6.1.9 ALC\_TDA.3.9D**

4632 **The developer shall produce and provide documentation explaining the functional differences,**  
4633 **if any, between the regenerated TOE copy and the original TOE.**

4634 **12.8.6.2 Content and presentation elements**

4635 **12.8.6.2.1 ALC\_TDA.3.1C**

4636 The list of unique TOE implementation representation identifiers as recorded during the TOE  
4637 generation time shall demonstrate the correspondence between the TOE implementation  
4638 representation element identifiers and the TOE implementation representation element names.

4639 **12.8.6.2.2 ALC\_TDA.3.2C**

4640 The TOE implementation representation element names shall be in the same form as used or  
4641 referenced by the development tool to generate the TOE.

4642 **12.8.6.2.3 ALC\_TDA.3.3C**

4643 The timestamp of the list of unique TOE implementation representation identifiers as recorded during  
4644 the TOE generation time shall be consistent with the creation time of the TOE.

4645 **12.8.6.2.4 ALC\_TDA.3.4C**

4646 The (author) origination information of the list of unique TOE implementation representation  
4647 identifiers as recorded during the TOE generation time shall be consistent with the (author)  
4648 origination information of the TOE.

4649 **12.8.6.2.5 ALC\_TDA.3.5C**

4650 The list of identifiers of the elements of implementation representation under the configuration scope  
4651 of ALC\_CMS.3 shall match with the list of unique TOE implementation representation identifiers as  
4652 recorded during the TOE generation time.

4653 **12.8.6.2.6 ALC\_TDA.3.6C**

4654 **The developer's explanation of the functional differences, if any, between the regenerated TOE**  
4655 **copy and the original TOE shall take into account all visible differences, if any, between the**  
4656 **regenerated TOE copy and the original TOE.**

4657 **12.8.6.3 Evaluator action elements**

4658 **12.8.6.3.1 ALC\_TDA.3.1E**

4659 The evaluator shall confirm that the information provided meets all requirements for content and  
4660 presentation of evidence.

4661 **12.8.6.3.2 ALC\_TDA.3.2E**

4662 The evaluator shall confirm that the development tool for generating the TOE is capable to use or  
4663 reference the implementation representation element names.

4664 **12.8.6.3.3 ALC\_TDA.3.3E**

4665 The evaluator shall confirm that the list of unique TOE implementation representation identifiers as  
4666 recorded during the TOE generation time is consistent with the creation time of the TOE.

4667 **12.8.6.3.4 ALC\_TDA.3.4E**

4668 The evaluator shall confirm that the (author) origination information of the list of unique TOE  
 4669 implementation representation identifiers as recorded during the TOE generation time is consistent  
 4670 with the (author) origination information of the TOE.

4671 **12.8.6.3.5 ALC\_TDA.3.5E**

4672 The evaluator shall check the integrity of the list of unique TOE implementation representation  
 4673 identifiers as recorded during the TOE generation time and its associated timestamp and (author)  
 4674 origination information.

4675 **12.8.6.3.6 ALC\_TDA.3.6E**

4676 The evaluator shall confirm the developer's ability to trace from the TOE to the list of unique TOE  
 4677 implementation representation identifiers as recorded during the TOE generation time.

4678 **12.8.6.3.7 ALC\_TDA.3.7E**

4679 The evaluator shall confirm that the list of identifiers of the elements of implementation  
 4680 representation under the configuration scope of ALC\_CMS.3 matches with the list of unique TOE  
 4681 implementation representation identifiers as recorded during the TOE generation time.

4682 **12.8.6.3.8 ALC\_TDA.3.8E**

4683 **The evaluator shall check that the developer's explanation of the functional differences, if any,**  
 4684 **between the regenerated TOE copy and the original TOE takes into account all visible**  
 4685 **differences, if any, between the regenerated TOE copy and the original TOE.**

4686 **12.9 Tools and techniques (ALC\_TAT)**4687 **12.9.1 Objectives**

4688 Tools and techniques is an aspect of selecting tools that are used to develop, analyse and implement  
 4689 the TOE. It includes requirements to prevent ill-defined, inconsistent or incorrect development tools  
 4690 from being used to develop the TOE. This includes, but is not limited to, programming languages,  
 4691 documentation, implementation standards, and other parts of the TOE such as supporting runtime  
 4692 libraries.

4693 **12.9.2 Component levelling**

4694 The components in this family are levelled on the basis of increasing requirements on the description  
 4695 and scope of the implementation standards and the documentation of implementation-dependent  
 4696 options.

4697 **12.9.3 Application notes**

4698 There is a requirement for well-defined development tools. These are tools that are clearly and  
 4699 completely described. For example, programming languages and computer aided design (CAD)  
 4700 systems that are based on a standard published by standards bodies are considered to be well-defined.  
 4701 Self-made tools would need further investigation to clarify whether they are well-defined.

4702 The requirement in ALC\_TAT.1.2C is especially applicable to programming languages so as to ensure  
 4703 that all statements in the source code have an unambiguous meaning.



4704 In ALC\_TAT.2 and ALC\_TAT.3, implementation guidelines may be accepted as an implementation  
4705 standard if they have been approved by some group of experts (e.g. academic experts, standards  
4706 bodies). Implementation standards are normally public, well accepted and common practise in a  
4707 specific industry, but developer-specific implementation guidelines may also be accepted as a  
4708 standard; the emphasis is on the expertise.

4709 Tools and techniques distinguishes between the implementation standards applied by the developer  
4710 (ALC\_TAT.2.3D) and the implementation standards for "all parts of the TOE" (ALC\_TAT.3.3D) which  
4711 include third party software, hardware, or firmware. The configuration list introduced in CM scope  
4712 (ALC\_CMS) requires that for each TSF relevant configuration item to indicate if it has been generated  
4713 by the TOE developer or by third party developers

#### 4714 **12.9.4 ALC\_TAT.1 Well-defined development tools**

4715 Dependencies: ADV\_IMP.1 Implementation representation of the TSF

##### 4716 **12.9.4.1 Developer action elements**

###### 4717 **12.9.4.1.1 ALC\_TAT.1.1D**

4718 **The developer shall provide the documentation identifying each development tool being used**  
4719 **for the TOE.**

###### 4720 **12.9.4.1.2 ALC\_TAT.1.2D**

4721 **The developer shall document and provide the selected implementation-dependent options of**  
4722 **each development tool.**

##### 4723 **12.9.4.2 Content and presentation elements**

###### 4724 **12.9.4.2.1 ALC\_TAT.1.1C**

4725 **Each development tool used for implementation shall be well-defined.**

###### 4726 **12.9.4.2.2 ALC\_TAT.1.2C**

4727 **The documentation of each development tool shall unambiguously define the meaning of all**  
4728 **statements as well as all conventions and directives used in the implementation.**

###### 4729 **12.9.4.2.3 ALC\_TAT.1.3C**

4730 **The documentation of each development tool shall unambiguously define the meaning of all**  
4731 **implementation-dependent options.**

##### 4732 **12.9.4.3 Evaluator action elements**

###### 4733 **12.9.4.3.1 ALC\_TAT.1.1E**

4734 **The evaluator shall confirm that the information provided meets all requirements for content**  
4735 **and presentation of evidence.**

#### 4736 **12.9.5 ALC\_TAT.2 Compliance with implementation standards**

4737 Dependencies: ADV\_IMP.1 Implementation representation of the TSF

4738 **12.9.5.1 Developer action elements**

4739 **12.9.5.1.1 ALC\_TAT.2.1D**

4740 The developer shall provide the documentation identifying each development tool being used for the  
4741 TOE.

4742 **12.9.5.1.2 ALC\_TAT.2.2D**

4743 The developer shall document and provide the selected implementation-dependent options of each  
4744 development tool.

4745 **12.9.5.1.3 ALC\_TAT.2.3D**

4746 **The developer shall describe and provide the implementation standards that are being applied**  
4747 **by the developer.**

4748 **12.9.5.2 Content and presentation elements**

4749 **12.9.5.2.1 ALC\_TAT.2.1C**

4750 Each development tool used for implementation shall be well-defined.

4751 **12.9.5.2.2 ALC\_TAT.2.2C**

4752 The documentation of each development tool shall unambiguously define the meaning of all  
4753 statements as well as all conventions and directives used in the implementation.

4754 **12.9.5.2.3 ALC\_TAT.2.3C**

4755 The documentation of each development tool shall unambiguously define the meaning of all  
4756 implementation-dependent options.

4757 **12.9.5.3 Evaluator action elements**

4758 **12.9.5.3.1 ALC\_TAT.2.1E**

4759 The evaluator shall confirm that the information provided meets all requirements for content and  
4760 presentation of evidence.

4761 **12.9.5.3.2 ALC\_TAT.2.2E**

4762 **The evaluator shall confirm that the implementation standards have been applied.**

4763 **12.9.6 ALC\_TAT.3 Compliance with implementation standards - all parts**

4764 Dependencies: ADV\_IMP.1 Implementation representation of the TSF

4765 **12.9.6.1 Developer action elements**

4766 **12.9.6.1.1 ALC\_TAT.3.1D**

4767 The developer shall provide the documentation identifying each development tool being used for the  
4768 TOE.

4769 **12.9.6.1.2 ALC\_TAT.3.2D**

4770 The developer shall document and provide the selected implementation-dependent options of each  
4771 development tool.

4772 **12.9.6.1.3 ALC\_TAT.3.3D**

4773 The developer shall describe and provide the implementation standards that are being applied by the  
4774 developer **and by any third-party providers for all parts of the TOE.**

4775 **12.9.6.2 Content and presentation elements**

4776 **12.9.6.2.1 ALC\_TAT.3.1C**

4777 Each development tool used for implementation shall be well-defined.

4778 **12.9.6.2.2 ALC\_TAT.3.2C**

4779 The documentation of each development tool shall unambiguously define the meaning of all  
4780 statements as well as all conventions and directives used in the implementation.

4781 **12.9.6.2.3 ALC\_TAT.3.3C**

4782 The documentation of each development tool shall unambiguously define the meaning of all  
4783 implementation-dependent options.

4784 **12.9.6.3 Evaluator action elements**

4785 **12.9.6.3.1 ALC\_TAT.3.1E**

4786 The evaluator shall confirm that the information provided meets all requirements for content and  
4787 presentation of evidence.

4788 **12.9.6.3.2 ALC\_TAT.3.2E**

4789 **The evaluator shall confirm that the implementation standards have been applied.**

4790 **12.10 Integration of composition parts and consistency check of delivery procedures**  
4791 **(ALC\_COMP)**

4792 **12.10.1 Objectives**

4793 The aims of this activity are to determine whether

- 4794 - the correct version of the application is installed onto/into the correct version of the  
4795 underlying platform, and
- 4796 - the preparative guidance procedures of Platform and Application Developers are compatible  
4797 with the acceptance procedure of the Composite Product Integrator.

4798 **12.10.2 Component levelling**

4799 This family contains only one component.

4800 **12.10.3 ALC\_COMP.1 Integration of the application into the underlying platform and**  
 4801 **Consistency check for delivery and acceptance procedures**

4802 Dependencies: No dependencies

4803 **12.10.3.1 Developer action elements**

4804 **12.10.3.1.1 ALC\_COMP.1.1D**

4805 The developer shall provide components configuration evidence; cf. item #7, item #8 and item #3 in  
 4806 Table D1, section Fehler! Verweisquelle konnte nicht gefunden werden.D.1-7.

4807 **12.10.3.2 Content and presentation elements**

4808 **12.10.3.2.1 ALC\_COMP.1.1C**

4809 The components configuration evidence shall show that the evaluated version of the application has  
 4810 been installed onto / embedded into the certified version of the underlying platform.

4811 **12.10.3.2.2 ALC\_COMP.1.2C**

4812 The components configuration evidence shall show that:

4813 i. The evidence for delivery and acceptance compatibility shall show that the delivery procedures of  
 4814 the Platform and Application Developers are compatible with the acceptance procedure of the  
 4815 Composite Product Integrator.

4816 ii. the evidence shall show that preparative guidance procedures prescribed by the Platform and  
 4817 Application Developers are either actually being used by the Composite Product Integrator or  
 4818 compatible with the Composite Product Integrator guidance and do not contradict each other.

4819 **12.10.3.3 Evaluator action elements**

4820 **12.10.3.3.1 ALC\_COMP.1.1E**

4821 The evaluator shall confirm that the information provided meets all requirements for content and  
 4822 presentation of evidence.

4823 **12.10.3.3.2 ALC\_COMP.1.2E**

4824 The evaluator shall confirm that the evidence for delivery compatibility is complete, coherent, and  
 4825 internally consistent.

4826 **13 Class ATE: Tests**

4827 **13.1 Introduction**

4828 The class “Tests” encompasses five families: Coverage (ATE\_COV), Depth (ATE\_DPT), Independent  
 4829 testing (ATE\_IND) (i.e. functional testing performed by evaluators), Functional tests (ATE\_FUN) and  
 4830 Composite functional testing (ATE\_COMP). Testing provides assurance that the TSF behaves as  
 4831 described (in the functional specification, TOE design, implementation representation, and allows  
 4832 straightforward traceability of SFR in test scenario).

4833 The emphasis in this class is on confirmation that the TSF operates according to its design  
 4834 descriptions. This class does not address penetration testing, which is based upon an analysis of the

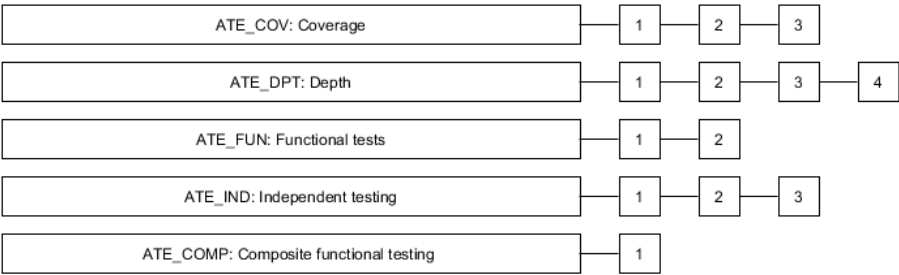
4835 TSF that specifically seeks to identify vulnerabilities in the design and implementation of the TSF.  
4836 Penetration testing is addressed separately as an aspect of vulnerability assessment in the AVA:  
4837 Vulnerability assessment class.

4838 The ATE: Tests class separates testing into developer testing and evaluator testing. The Coverage  
4839 (ATE\_COV), and Depth (ATE\_DPT) families address the completeness of developer testing. Coverage  
4840 (ATE\_COV) addresses the rigour with which the functional specification is tested; Depth (ATE\_DPT)  
4841 addresses whether testing against other design descriptions (security architecture, TOE design, and  
4842 implementation representation) is required.

4843 Functional tests (ATE\_FUN) addresses the performing of the tests by the developer and how this  
4844 testing should be documented. Finally, Independent testing (ATE\_IND) then addresses evaluator  
4845 testing: whether the evaluator should repeat part or all of the developer testing and how much  
4846 independent testing the evaluator should do.

4847 Composite functional testing (ATE\_COMP) determines whether the composite product as a whole  
4848 exhibits the properties necessary to satisfy the functional requirements of its Security Target.

4849 Figure 12 shows the families within this class, and the hierarchy of components within the families.



4850

4851 **Figure 12 — ATE: Tests class decomposition**

4852 **13.2 Coverage (ATE\_COV)**

4853 **13.2.1 Objectives**

4854 This family establishes that the TSF has been tested against its functional specification. This is  
4855 achieved through an examination of developer evidence of correspondence.

4856 **13.2.2 Component levelling**

4857 The components in this family are levelled on the basis of specification.

4858 **13.2.3 Application notes**

4859 **13.2.4 ATE\_COV.1 Evidence of coverage**

4860 Dependencies: ADV\_FSP.2 Security-enforcing functional specification

4861 ATE\_FUN.1 Functional testing

4862 **13.2.4.1 Objectives**

4863 The objective of this component is to establish that some of the TSFIs have been tested.

4864 **13.2.4.2 Application notes**

4865 In this component the developer shows how tests in the test documentation correspond to TSFIs in the  
4866 functional specification. This can be achieved by a statement of correspondence, perhaps using a table.

4867 **13.2.4.3 Developer action elements**

4868 **13.2.4.3.1 ATE\_COV.1.1D**

4869 The developer shall provide evidence of the test coverage.

4870 **13.2.4.4 Content and presentation elements**

4871 **13.2.4.4.1 ATE\_COV.1.1C**

4872 The evidence of the test coverage shall show the correspondence between the tests in the test  
4873 documentation and the TSFIs in the functional specification.

4874 **13.2.4.5 Evaluator action elements**

4875 **13.2.4.5.1 ATE\_COV.1.1E**

4876 The evaluator shall confirm that the information provided meets all requirements for content  
4877 and presentation of evidence.

4878 **13.2.5 ATE\_COV.2 Analysis of coverage**

4879 Dependencies: ADV\_FSP.2 Security-enforcing functional specification

4880 ATE\_FUN.1 Functional testing

4881 **13.2.5.1 Objectives**

4882 The objective of this component is to confirm that all of the TSFIs have been tested.

4883 **13.2.5.2 Application notes**

4884 In this component the developer confirms that tests in the test documentation correspond to all of the  
4885 TSFIs in the functional specification. This can be achieved by a statement of correspondence, perhaps  
4886 using a table, but the developer also provides an analysis of the test coverage.

4887 **13.2.5.3 Developer action elements**

4888 **13.2.5.3.1 ATE\_COV.2.1D**

4889 The developer shall provide an analysis of the test coverage.

4890 **13.2.5.4 Content and presentation elements**

4891 **13.2.5.4.1 ATE\_COV.2.1C**

4892 The **analysis** of the test coverage shall **demonstrate** the correspondence between the tests in the test  
4893 documentation and the TSFIs in the functional specification.

4894 **13.2.5.4.2 ATE\_COV.2.2C**

4895 **The analysis of the test coverage shall demonstrate that all TSFIs in the functional specification**  
4896 **have been tested.**

4897 **13.2.5.5 Evaluator action elements**

4898 **13.2.5.5.1 ATE\_COV.2.1E**

4899 The evaluator shall confirm that the information provided meets all requirements for content and  
4900 presentation of evidence.

4901 **13.2.6 ATE\_COV.3 Rigorous analysis of coverage**

4902 Dependencies: ADV\_FSP.2 Security-enforcing functional specification

4903 ATE\_FUN.1 Functional testing

4904 **13.2.6.1 Objectives**

4905 In this component, the objective is to confirm that the developer performed exhaustive tests of all  
4906 interfaces in the functional specification.

4907 The objective of this component is to confirm that all parameters of all of the TSFIs have been tested.

4908 **13.2.6.2 Application notes**

4909 In this component the developer is required to show how tests in the test documentation correspond  
4910 to all of the TSFIs in the functional specification. This can be achieved by a statement of  
4911 correspondence, perhaps using a table, but in addition the developer is required to demonstrate that  
4912 the tests exercise all of the parameters of all TSFIs. This additional requirement includes bounds  
4913 testing (i.e. verifying that errors are generated when stated limits are exceeded) and negative testing  
4914 (e.g. when access is given to User A, verifying not only that User A now has access, but also that User B  
4915 did not suddenly gain access). This kind of testing is not, strictly speaking, *exhaustive* because not  
4916 every possible value of the parameters is expected to be checked.

4917 **13.2.6.3 Developer action elements**

4918 **13.2.6.3.1 ATE\_COV.3.1D**

4919 The developer shall provide an analysis of the test coverage.

4920 **13.2.6.4 Content and presentation elements**

4921 **13.2.6.4.1 ATE\_COV.3.1C**

4922 The analysis of the test coverage shall demonstrate the correspondence between the tests in the test  
4923 documentation and the TSFIs in the functional specification.

4924 **13.2.6.4.2 ATE\_COV.3.2C**

4925 The analysis of the test coverage shall demonstrate that all TSFIs in the functional specification have  
 4926 been **completely** tested.

4927 **13.2.6.5 Evaluator action elements**4928 **13.2.6.5.1 ATE\_COV.3.1E**

4929 The evaluator shall confirm that the information provided meets all requirements for content and  
 4930 presentation of evidence.

4931 **13.3 Depth (ATE\_DPT)**4932 **13.3.1 Objectives**

4933 The components in this family deal with the level of detail to which the TSF is tested by the developer.  
 4934 Testing of the TSF is based upon increasing depth of information derived from additional design  
 4935 representations and descriptions (TOE design, implementation representation, and security  
 4936 architecture description).

4937 The objective is to counter the risk of missing an error in the development of the TOE. Testing that  
 4938 exercises specific internal interfaces can provide assurance not only that the TSF exhibits the desired  
 4939 external security behaviour, but also that this behaviour stems from correctly operating internal  
 4940 functionality.

4941 **13.3.2 Component levelling**

4942 The components in this family are levelled on the basis of increasing detail provided in the TSF  
 4943 representations, from the TOE design to the implementation representation. This levelling reflects the  
 4944 TSF representations presented in the ADV class.

4945 **13.3.3 Application notes**

4946 The TOE design describes the internal components (e.g. subsystems) and, perhaps, modules of the TSF,  
 4947 together with a description of the interfaces among these components and modules. Evidence of  
 4948 testing of this TOE design must show that the internal interfaces have been exercised and seen to  
 4949 behave as described. This may be achieved through testing via the external interfaces of the TSF, or by  
 4950 testing of the TOE subsystem or module interfaces in isolation, perhaps employing a test harness. In  
 4951 cases where some aspects of an internal interface cannot be tested via the external interfaces, there  
 4952 should either be justification that these aspects need not be tested, or the internal interface needs to  
 4953 be tested directly. In the latter case the TOE design needs to be sufficiently detailed in order to  
 4954 facilitate direct testing.

4955 In cases where the description of the TSF's architectural soundness (in Security Architecture  
 4956 (ADV\_ARC)) cites specific mechanisms, the tests performed by the developer must show that the  
 4957 mechanisms have been exercised and seen to behave as described.

4958 At the highest component of this family, the testing is performed not only against the TOE design, but  
 4959 also against the implementation representation.

4960 **13.3.4 ATE\_DPT.1 Testing: basic design**

4961 Dependencies: ADV\_ARC.1 Security architecture description



4962	ADV_TDS.2 Architectural design
4963	ATE_FUN.1 Functional testing
4964	<b>13.3.4.1 Objectives</b>
4965	The subsystem descriptions of the TSF provide a high-level description of the internal workings of the
4966	TSF. Testing at the level of the TOE subsystems provides assurance that the TSF subsystems behave
4967	and interact as described in the TOE design and the security architecture description.
4968	<b>13.3.4.2 Developer action elements</b>
4969	<b>13.3.4.2.1 ATE_DPT.1.1D</b>
4970	<b>The developer shall provide the analysis of the depth of testing.</b>
4971	<b>13.3.4.3 Content and presentation elements</b>
4972	<b>13.3.4.3.1 ATE_DPT.1.1C</b>
4973	<b>The analysis of the depth of testing shall demonstrate the correspondence between the tests in</b>
4974	<b>the test documentation and the TSF subsystems in the TOE design.</b>
4975	<b>13.3.4.3.2 ATE_DPT.1.2C</b>
4976	<b>The analysis of the depth of testing shall demonstrate that all TSF subsystems in the TOE design</b>
4977	<b>have been tested.</b>
4978	<b>13.3.4.4 Evaluator action elements</b>
4979	<b>13.3.4.4.1 ATE_DPT.1.1E</b>
4980	<b>The evaluator shall confirm that the information provided meets all requirements for content</b>
4981	<b>and presentation of evidence.</b>
4982	<b>13.3.5 ATE_DPT.2 Testing: security enforcing modules</b>
4983	Dependencies: ADV_ARC.1 Security architecture description
4984	ADV_TDS.3 Basic modular design
4985	ATE_FUN.1 Functional testing
4986	<b>13.3.5.1 Objectives</b>
4987	The subsystem and module descriptions of the TSF provide a high-level description of the internal
4988	workings, and a description of the interfaces of the SFR-enforcing modules, of the TSF. Testing at this
4989	level of TOE description provides assurance that the TSF subsystems and SFR-enforcing modules
4990	behave and interact as described in the TOE design and the security architecture description.
4991	<b>13.3.5.2 Developer action elements</b>
4992	<b>13.3.5.2.1 ATE_DPT.2.1D</b>
4993	<b>The developer shall provide the analysis of the depth of testing.</b>

4994 **13.3.5.3 Content and presentation elements**

4995 **13.3.5.3.1 ATE\_DPT.2.1C**

4996 The analysis of the depth of testing shall demonstrate the correspondence between the tests in the test  
4997 documentation and the TSF subsystems **and SFR-enforcing modules** in the TOE design.

4998 **13.3.5.3.2 ATE\_DPT.2.2C**

4999 The analysis of the depth of testing shall demonstrate that all TSF subsystems in the TOE design have  
5000 been tested.

5001 **13.3.5.3.3 ATE\_DPT.2.3C**

5002 **The analysis of the depth of testing shall demonstrate that the SFR-enforcing modules in the**  
5003 **TOE design have been tested.**

5004 **13.3.5.4 Evaluator action elements**

5005 **13.3.5.4.1 ATE\_DPT.2.1E**

5006 The evaluator shall confirm that the information provided meets all requirements for content and  
5007 presentation of evidence.

5008 **13.3.6 ATE\_DPT.3 Testing: modular design**

5009 Dependencies: ADV\_ARC.1 Security architecture description

5010 ADV\_TDS.4 Semiformal modular design

5011 ATE\_FUN.1 Functional testing

5012 **13.3.6.1 Objectives**

5013 The subsystem and module descriptions of the TSF provide a high-level description of the internal  
5014 workings, and a description of the interfaces of the modules, of the TSF. Testing at this level of TOE  
5015 description provides assurance that the TSF subsystems and modules behave and interact as  
5016 described in the TOE design and the security architecture description.

5017 **13.3.6.2 Developer action elements**

5018 **13.3.6.2.1 ATE\_DPT.3.1D**

5019 The developer shall provide the analysis of the depth of testing.

5020 **13.3.6.3 Content and presentation elements**

5021 **13.3.6.3.1 ATE\_DPT.3.1C**

5022 The analysis of the depth of testing shall demonstrate the correspondence between the tests in the test  
5023 documentation and the TSF subsystems and modules in the TOE design.

5024 **13.3.6.3.2 ATE\_DPT.3.2C**

5025 The analysis of the depth of testing shall demonstrate that all TSF subsystems in the TOE design have  
5026 been tested.

5027 **13.3.6.3.3 ATE\_DPT.3.3C**

5028 The analysis of the depth of testing shall demonstrate that **all TSF** modules in the TOE design have  
5029 been tested.

5030 **13.3.6.4 Evaluator action elements**

5031 **13.3.6.4.1 ATE\_DPT.3.1E**

5032 The evaluator shall confirm that the information provided meets all requirements for content and  
5033 presentation of evidence.

5034 **13.3.7 ATE\_DPT.4 Testing: implementation representation**

5035 Dependencies: ADV\_ARC.1 Security architecture description

5036 ADV\_TDS.4 Semiformal modular design

5037 ADV\_IMP.1 Implementation representation of the TSF

5038 ATE\_FUN.1 Functional testing

5039 **13.3.7.1 Objectives**

5040 The subsystem and module descriptions of the TSF provide a high-level description of the internal  
5041 workings, and a description of the interfaces of the modules, of the TSF. Testing at this level of TOE  
5042 description provides assurance that the TSF subsystems and modules behave and interact as  
5043 described in the TOE design and the security architecture description, and in accordance with the  
5044 implementation representation.

5045 **13.3.7.2 Developer action elements**

5046 **13.3.7.2.1 ATE\_DPT.4.1D**

5047 The developer shall provide the analysis of the depth of testing.

5048 **13.3.7.3 Content and presentation elements**

5049 **13.3.7.3.1 ATE\_DPT.4.1C**

5050 The analysis of the depth of testing shall demonstrate the correspondence between the tests in the test  
5051 documentation and the TSF subsystems and modules in the TOE design.

5052 **13.3.7.3.2 ATE\_DPT.4.2C**

5053 The analysis of the depth of testing shall demonstrate that all TSF subsystems in the TOE design have  
5054 been tested.

5055 **13.3.7.3.3 ATE\_DPT.4.3C**

5056 The analysis of the depth of testing shall demonstrate that all modules in the TOE design have been  
5057 tested.

5058 **13.3.7.3.4 ATE\_DPT.4.4C**

5059 **The analysis of the depth of testing shall demonstrate that the TSF operates in accordance with**  
5060 **its implementation representation.**

5061 **13.3.7.4 Evaluator action elements**

5062 **13.3.7.4.1 ATE\_DPT.4.1E**

5063 The evaluator shall confirm that the information provided meets all requirements for content and  
5064 presentation of evidence.

5065 **13.4 Functional tests (ATE\_FUN)**

5066 **13.4.1 Objectives**

5067 Functional testing performed by the developer provides assurance that the tests in the test  
5068 documentation are performed and documented correctly. The correspondence of these tests to the  
5069 design descriptions of the TSF is achieved through the Coverage (ATE\_COV) and Depth (ATE\_DPT)  
5070 families.

5071 This family contributes to providing assurance that the likelihood of undiscovered flaws is relatively  
5072 small.

5073 The families Coverage (ATE\_COV), Depth (ATE\_DPT) and Functional tests (ATE\_FUN) are used in  
5074 combination to define the evidence of testing to be supplied by a developer. Independent functional  
5075 testing by the evaluator is specified by Independent testing (ATE\_IND).

5076 **13.4.2 Component levelling**

5077 This family contains two components, the higher requiring that ordering dependencies are analysed.

5078 **13.4.3 Application notes**

5079 Procedures for performing tests are expected to provide instructions for using test programs and test  
5080 suites, including the test environment, test conditions, test data parameters and values. The test  
5081 procedures should also show how the test results are derived from the test inputs.

5082 Ordering dependencies are relevant when the successful execution of a particular test depends upon  
5083 the existence of a particular state. For example, this might require that test A be executed immediately  
5084 before test B, since the state resulting from the successful execution of test A is a prerequisite for the  
5085 successful execution of test B. Thus, failure of test B could be related to a problem with the ordering  
5086 dependencies. In the above example, test B could fail because test C (rather than test A) was executed  
5087 immediately before it, or the failure of test B could be related to a failure of test A.

5088 **13.4.4 ATE\_FUN.1 Functional testing**

5089 Dependencies: ATE\_COV.1 Evidence of coverage

5090 **13.4.4.1 Objectives**

5091 The objective is for the developer to demonstrate that the tests in the test documentation are  
5092 performed and documented correctly.

- 5093 **13.4.4.2 Developer action elements**
- 5094 **13.4.4.2.1 ATE\_FUN.1.1D**
- 5095 **The developer shall test the TSF and document the results.**
- 5096 **13.4.4.2.2 ATE\_FUN.1.2D**
- 5097 **The developer shall provide test documentation.**
- 5098 **13.4.4.3 Content and presentation elements**
- 5099 **13.4.4.3.1 ATE\_FUN.1.1C**
- 5100 **The test documentation shall consist of test plans, expected test results and actual test results.**
- 5101 **13.4.4.3.2 ATE\_FUN.1.2C**
- 5102 **The test plans shall identify the tests to be performed and describe the scenarios for**  
 5103 **performing each test. These scenarios shall include any ordering dependencies on the results**  
 5104 **of other tests.**
- 5105 **13.4.4.3.3 ATE\_FUN.1.3C**
- 5106 **The expected test results shall show the anticipated outputs from a successful execution of the**  
 5107 **tests.**
- 5108 **13.4.4.3.4 ATE\_FUN.1.4C**
- 5109 **The actual test results shall be consistent with the expected test results.**
- 5110 **13.4.4.4 Evaluator action elements**
- 5111 **13.4.4.4.1 ATE\_FUN.1.1E**
- 5112 **The evaluator shall confirm that the information provided meets all requirements for content**  
 5113 **and presentation of evidence.**
- 5114 **13.4.5 ATE\_FUN.2 Ordered functional testing**
- 5115 Dependencies: ATE\_COV.1 Evidence of coverage
- 5116 **13.4.5.1 Objectives**
- 5117 **The objectives are for the developer to demonstrate that the tests in the test documentation are**  
 5118 **performed and documented correctly, and to ensure that testing is structured such as to avoid circular**  
 5119 **arguments about the correctness of the interfaces being tested.**
- 5120 **13.4.5.2 Application notes**
- 5121 **Although the test procedures may state pre-requisite initial test conditions in terms of ordering of**  
 5122 **tests, they may not provide a rationale for the ordering. An analysis of test ordering is an important**  
 5123 **factor in determining the adequacy of testing, as there is a possibility of faults being concealed by the**  
 5124 **ordering of tests.**

5125 **13.4.5.3 Developer action elements**

5126 **13.4.5.3.1 ATE\_FUN.2.1D**

5127 The developer shall test the TSF and document the results.

5128 **13.4.5.3.2 ATE\_FUN.2.2D**

5129 The developer shall provide test documentation.

5130 **13.4.5.4 Content and presentation elements**

5131 **13.4.5.4.1 ATE\_FUN.2.1C**

5132 The test documentation shall consist of test plans, expected test results and actual test results.

5133 **13.4.5.4.2 ATE\_FUN.2.2C**

5134 The test plans shall identify the tests to be performed and describe the scenarios for performing each  
5135 test. These scenarios shall include any ordering dependencies on the results of other tests.

5136 **13.4.5.4.3 ATE\_FUN.2.3C**

5137 The expected test results shall show the anticipated outputs from a successful execution of the tests.

5138 **13.4.5.4.4 ATE\_FUN.2.4C**

5139 The actual test results shall be consistent with the expected test results.

5140 **13.4.5.4.5 ATE\_FUN.2.5C**

5141 **The test documentation shall include an analysis of the test procedure ordering dependencies.**

5142 **13.4.5.5 Evaluator action elements**

5143 **13.4.5.5.1 ATE\_FUN.2.1E**

5144 The evaluator shall confirm that the information provided meets all requirements for content and  
5145 presentation of evidence.

5146 **13.5 Independent testing (ATE\_IND)**

5147 **13.5.1 Objectives**

5148 The objectives of this family are built upon the assurances achieved in the ATE\_FUN, ATE\_COV, and  
5149 ATE\_DPT families by verifying the developer testing and performing additional tests by the evaluator.

5150 **13.5.2 Component levelling**

5151 Levelling is based upon the amount of developer test documentation and test support and the amount  
5152 of evaluator testing.

5153 **13.5.3 Application notes**

5154 This family deals with the degree to which there is independent functional testing of the TSF.  
5155 Independent functional testing may take the form of repeating the developer's functional tests (in

5156 whole or in part) or of extending the scope or the depth of the developer's tests. These activities are  
 5157 complementary, and an appropriate mix must be planned for each TOE, which takes into account the  
 5158 availability and coverage of test results, and the functional complexity of the TSF.

5159 Sampling of developer tests is intended to provide confirmation that the developer has carried out his  
 5160 planned test programme on the TSF, and has correctly recorded the results. The size of sample  
 5161 selected will be influenced by the detail and quality of the developer's functional test results. The  
 5162 evaluator will also need to consider the scope for devising additional tests, and the relative benefit that  
 5163 may be gained from effort in these two areas. It is recognised that repetition of all developer tests may  
 5164 be feasible and desirable in some cases, but may be very arduous and less productive in others. The  
 5165 highest component in this family should therefore be used with caution. Sampling will address the  
 5166 whole range of test results available, including those supplied to meet the requirements of both  
 5167 Coverage (ATE\_COV) and Depth (ATE\_DPT).

5168 There is also a need to consider the different configurations of the TOE that are included within the  
 5169 evaluation. The evaluator will need to assess the applicability of the results provided, and to plan his  
 5170 own testing accordingly.

5171 The suitability of the TOE for testing is based on the access to the TOE, and the supporting  
 5172 documentation and information required (including any test software or tools) to run tests. The need  
 5173 for such support is addressed by the dependencies to other assurance families.

5174 Additionally, suitability of the TOE for testing may be based on other considerations. For example, the  
 5175 version of the TOE submitted by the developer may not be the final version.

5176 The term *interfaces* refers to interfaces described in the functional specification and TOE design, and  
 5177 parameters passed through invocations identified in the implementation representation. The exact set  
 5178 of interfaces to be used is selected through Coverage (ATE\_COV) and the Depth (ATE\_DPT)  
 5179 components.

5180 References to a subset of the interfaces are intended to allow the evaluator to design an appropriate  
 5181 set of tests which is consistent with the objectives of the evaluation being conducted.

#### 5182 **13.5.4 ATE\_IND.1 Independent testing - conformance**

5183 Dependencies: ADV\_FSP.1 Basic functional specification

5184 AGD\_OPE.1 Operational user guidance

5185 AGD\_PRE.1 Preparative procedures

##### 5186 **13.5.4.1 Objectives**

5187 In this component, the objective is to demonstrate that the TOE operates in accordance with its design  
 5188 representations and guidance documents.

##### 5189 **13.5.4.2 Application notes**

5190 This component does not address the use of developer test results. It is applicable where such results  
 5191 are not available, and also in cases where the developer's testing is accepted without validation. The  
 5192 evaluator is required to devise and conduct tests with the objective of confirming that the TOE  
 5193 operates in accordance with its design representations, including but not limited to the functional  
 5194 specification. The approach is to gain confidence in correct operation through representative testing,  
 5195 rather than to conduct every possible test. The extent of testing to be planned for this purpose is a

5196 methodology issue, and needs to be considered in the context of a particular TOE and the balance of  
5197 other evaluation activities.

5198 **13.5.4.3 Developer action elements**

5199 **13.5.4.3.1 ATE\_IND.1.1D**

5200 **The developer shall provide the TOE for testing.**

5201 **13.5.4.4 Content and presentation elements**

5202 **13.5.4.4.1 ATE\_IND.1.1C**

5203 **The TOE shall be suitable for testing.**

5204 **13.5.4.5 Evaluator action elements**

5205 **13.5.4.5.1 ATE\_IND.1.1E**

5206 **The evaluator shall confirm that the information provided meets all requirements for content**  
5207 **and presentation of evidence.**

5208 **13.5.4.5.2 ATE\_IND.1.2E**

5209 **The evaluator shall test a subset of the TSF to confirm that the TSF operates as specified.**

5210 **13.5.5 ATE\_IND.2 Independent testing - sample**

5211 Dependencies: ADV\_FSP.2 Security-enforcing functional specification

5212 AGD\_OPE.1 Operational user guidance

5213 AGD\_PRE.1 Preparative procedures

5214 ATE\_COV.1 Evidence of coverage

5215 ATE\_FUN.1 Functional testing

5216 **13.5.5.1 Objectives**

5217 In this component, the objective is to demonstrate that the TOE operates in accordance with its design  
5218 representations and guidance documents. Evaluator testing confirms that the developer performed  
5219 some tests of some interfaces in the functional specification.

5220 **13.5.5.2 Application notes**

5221 The intent is that the developer should provide the evaluator with materials necessary for the efficient  
5222 reproduction of developer tests. This may include such things as machine-readable test  
5223 documentation, test programs, etc.

5224 This component contains a requirement that the evaluator has available test results from the  
5225 developer to supplement the programme of testing. The evaluator will repeat a sample of the  
5226 developer's tests to gain confidence in the results obtained. Having established such confidence the  
5227 evaluator will build upon the developer's testing by conducting additional tests that exercise the TOE  
5228 in a different manner. By using a platform of validated developer test results the evaluator is able to



5229 gain confidence that the TOE operates correctly in a wider range of conditions than would be possible  
 5230 purely using the developer's own efforts, given a fixed level of resource. Having gained confidence that  
 5231 the developer has tested the TOE, the evaluator will also have more freedom, where appropriate, to  
 5232 concentrate testing in areas where examination of documentation or specialist knowledge has raised  
 5233 particular concerns.

5234 **13.5.5.3 Developer action elements**

5235 **13.5.5.3.1 ATE\_IND.2.1D**

5236 The developer shall provide the TOE for testing.

5237 **13.5.5.4 Content and presentation elements**

5238 **13.5.5.4.1 ATE\_IND.2.1C**

5239 The TOE shall be suitable for testing.

5240 **13.5.5.4.2 ATE\_IND.2.2C**

5241 **The developer shall provide an equivalent set of resources to those that were used in the**  
 5242 **developer's functional testing of the TSF.**

5243 **13.5.5.5 Evaluator action elements**

5244 **13.5.5.5.1 ATE\_IND.2.1E**

5245 The evaluator shall confirm that the information provided meets all requirements for content and  
 5246 presentation of evidence.

5247 **13.5.5.5.2 ATE\_IND.2.2E**

5248 **The evaluator shall execute a sample of tests in the test documentation to verify the developer**  
 5249 **test results.**

5250 **13.5.5.5.3 ATE\_IND.2.3E**

5251 The evaluator shall test a subset of the TSF to confirm that the TSF operates as specified.

5252 **13.5.6 ATE\_IND.3 Independent testing - complete**

5253 Dependencies: ADV\_FSP.4 Complete functional specification

5254 AGD\_OPE.1 Operational user guidance

5255 AGD\_PRE.1 Preparative procedures

5256 ATE\_COV.1 Evidence of coverage

5257 ATE\_FUN.1 Functional testing

5258 **13.5.6.1 Objectives**

5259 In this component, the objective is to demonstrate that the TOE operates in accordance with its design  
 5260 representations and guidance documents. Evaluator testing includes repeating all of the developer  
 5261 tests.

5262 **13.5.6.2 Application notes**

5263 The intent is that the developer should provide the evaluator with materials necessary for the efficient  
 5264 reproduction of developer tests. This may include such things as machine-readable test  
 5265 documentation, test programs, etc.

5266 In this component the evaluator must repeat all of the developer's tests as part of the programme of  
 5267 testing. As in the previous component the evaluator will also conduct tests that aim to exercise the TSF  
 5268 in a different manner from that achieved by the developer. In cases where developer testing has been  
 5269 exhaustive, there may remain little scope for this.

5270 **13.5.6.3 Developer action elements**5271 **13.5.6.3.1 ATE\_IND.3.1D**

5272 The developer shall provide the TOE for testing.

5273 **13.5.6.4 Content and presentation elements**5274 **13.5.6.4.1 ATE\_IND.3.1C**

5275 The TOE shall be suitable for testing.

5276 **13.5.6.4.2 ATE\_IND.3.2C**

5277 The developer shall provide an equivalent set of resources to those that were used in the developer's  
 5278 functional testing of the TSF.

5279 **13.5.6.5 Evaluator action elements**5280 **13.5.6.5.1 ATE\_IND.3.1E**

5281 The evaluator shall confirm that the information provided meets all requirements for content and  
 5282 presentation of evidence.

5283 **13.5.6.5.2 ATE\_IND.3.2E**

5284 The evaluator shall execute **all** tests in the test documentation to verify the developer test results.

5285 **13.5.6.5.3 ATE\_IND.3.3E**

5286 The evaluator shall test the TSF to confirm that the **entire** TSF operates as specified.

5287 **13.6 Composite functional testing (ATE\_COMP)**5288 **13.6.1 Objectives**

5289 The objective of this family is to determine whether composite product as a whole exhibits the  
 5290 properties necessary to satisfy the functional requirements of its Security Target.

5291 **13.6.2 Application notes**

5292 A composite product can be tested by testing the components of it separately and by testing the  
5293 integrated product. Separate testing means that the platform and the application are being tested  
5294 independent of each other. A lot of tests of the platform may have been performed within the scope of  
5295 its accomplished evaluation. The application may be tested on a simulator or an emulator, which  
5296 represent a virtual machine.

5297 Integration testing means that the composite product is being tested as it is: the application is running  
5298 on the platform.

5299 Behaviour of implementation of some SFRs can depend on properties of the underlying platform as  
5300 well as of the application (e.g. correctness of the measures of the composite product to withstand a  
5301 side channel attack or correctness of the implementation of tamper resistance against physical  
5302 attacks). In such a case the SFR implementation shall be tested on the final composite product, but not  
5303 on a simulator or an emulator.

5304 This activity focuses exclusively on testing of the composite product as a whole and represents merely  
5305 partial efforts within the general test approach being covered by the assurance ATE. These integration  
5306 tests shall be specified and performed, whereby the approach of the standard assurance families of  
5307 the class ATE shall be applied.

5308 **13.6.3 ATE\_COMP.1 Composite product functional testing**

5309 Dependencies: No dependencies

5310 **13.6.3.1 Developer action elements**

5311 **13.6.3.1.1 ATE\_COMP.1.1D**

5312 **The developer shall provide a set of tests as required by the assurance package chosen.**

5313 **13.6.3.1.2 ATE\_COMP.1.2D**

5314 **The developer shall provide the composite TOE for testing.**

5315 **13.6.3.2 Content and presentation elements**

5316 **13.6.3.2.1 ATE\_COMP.1.1C**

5317 **Content and presentation of the specification and documentation of the *integration* tests shall**  
5318 **correspond to the standard<sup>12</sup> requirements of the assurance families ATE\_FUN and ATE\_COV.**

5319 **13.6.3.2.2 ATE\_COMP.1.2C**

5320 **The composite TOE provided shall be suitable for testing.**

---

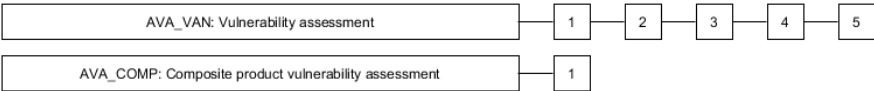
<sup>12</sup> i.e. as defined by ISO/IEC 18045

5321 **13.6.3.3 Evaluator action elements**  
5322 **13.6.3.3.1 ATE\_COMP.1.1E**  
5323 **The evaluator shall confirm that the information provided meets all requirements for content**  
5324 **and presentation of evidence.**

5325 **14 Class AVA: Vulnerability assessment**

5326 **14.1 Introduction**

5327 The AVA: Vulnerability assessment class addresses the possibility of exploitable vulnerabilities  
5328 introduced in the development or the operation of the TOE.  
5329 Figure 13 shows the families within this class, and the hierarchy of components within the families.



5330  
5331 **Figure 13 — AVA: Vulnerability assessment class decomposition**

5332 **14.2 Application notes**

5333 Generally, the vulnerability assessment activity covers various vulnerabilities in the development and  
5334 operation of the TOE. Development vulnerabilities take advantage of some property of the TOE, or the  
5335 product where the TOE resides, which was introduced during its development, e.g. defeating the TSF  
5336 self-protection through tampering, direct attack or monitoring of the TSF, defeating the TSF domain  
5337 separation through monitoring or direct attack the TSF, or defeating non-bypassability through  
5338 circumventing (bypassing) the TSF. Explicit dependencies of the TOE on IT systems in the  
5339 environment must also be considered. Operational vulnerabilities take advantage of weaknesses in  
5340 non-technical countermeasures to violate the TOE SFRs, e.g. misuse or incorrect configuration. Misuse  
5341 investigates whether the TOE can be configured or used in a manner that is insecure, but that an  
5342 administrator or user of the TOE would reasonably believe to be secure.

5343 Assessment of development vulnerabilities is covered by the assurance family AVA\_VAN. Basically, all  
5344 development vulnerabilities can be considered in the context of AVA\_VAN due to the fact, that this  
5345 family allows application of a wide range of assessment methodologies being unspecific to the kind of  
5346 an attack scenario. These unspecific assessment methodologies comprise, among other, also the  
5347 specific methodologies for those TSF where covert channels are to be considered (a channel capacity  
5348 estimation can be done using informal engineering measurements, as well as actual test  
5349 measurements) or can be overcome by the use of sufficient resources in the form of a direct attack  
5350 (underlying technical concept of those TSF is based on probabilistic or permutational mechanisms; a  
5351 qualification of their security behaviour and the effort required to overcome them can be made using a  
5352 quantitative or statistical analysis).

5353 If there are security objectives specified in the ST to either to prevent one user of the TOE from  
5354 observing activity associated with another user of the TOE, or to ensure that information flows cannot  
5355 be used to achieve enforced illicit data signals, covert channel analysis should be considered during  
5356 the conduct of the vulnerability analysis. This is often reflected by the inclusion of Unobservability

5357 (FPR\_UNO) and multilevel access control policies specified through Access control policy (FDP\_ACC)  
5358 and/or Information flow control policy (FDP\_IFC) requirements in the ST.

### 5359 **14.3 Vulnerability analysis (AVA\_VAN)**

#### 5360 **14.3.1 Objectives**

5361 Vulnerability analysis is an assessment to determine whether potential vulnerabilities identified,  
5362 during the evaluation of the development and anticipated operation of the TOE or by other methods  
5363 (e.g. by flaw hypotheses or quantitative or statistical analysis of the security behaviour of the  
5364 underlying security mechanisms), could allow attackers to violate the SFRs.

5365 Vulnerability analysis deals with the threats that an attacker will be able to discover flaws that will  
5366 allow unauthorised access to data and functionality, allow the ability to interfere with or alter the TSF,  
5367 or interfere with the authorised capabilities of other users.

5368 In case of a **multi-assurance evaluation** the vulnerability analysis shall assess the defined **sub-TSF** as  
5369 well as the TOE as a whole.

#### 5370 **14.3.2 Component levelling**

5371 Levelling is based on an increasing rigour of vulnerability analysis by the evaluator and increased  
5372 levels of attack potential required by an attacker to identify and exploit the potential vulnerabilities.

#### 5373 **14.3.3 AVA\_VAN.1 Vulnerability survey**

5374 Dependencies: ADV\_FSP.1 Basic functional specification

5375 AGD\_OPE.1 Operational user guidance

5376 AGD\_PRE.1 Preparative procedures

##### 5377 **14.3.3.1 Objectives**

5378 A vulnerability survey of information available in the public domain is performed by the evaluator to  
5379 ascertain potential vulnerabilities that may be easily found by an attacker.

5380 The evaluator performs penetration testing, to confirm that the potential vulnerabilities cannot be  
5381 exploited in the operational environment for the TOE. Penetration testing is performed by the  
5382 evaluator assuming an attack potential of Basic.

##### 5383 **14.3.3.2 Developer action elements**

###### 5384 **14.3.3.2.1 AVA\_VAN.1.1D**

5385 **The developer shall provide the TOE for testing.**

##### 5386 **14.3.3.3 Content and presentation elements**

###### 5387 **14.3.3.3.1 AVA\_VAN.1.1C**

5388 **The TOE shall be suitable for testing.**

5389 **14.3.3.4 Evaluator action elements**

5390 **14.3.3.4.1 AVA\_VAN.1.1E**

5391 **The evaluator shall confirm that the information provided meets all requirements for content**  
5392 **and presentation of evidence.**

5393 **14.3.3.4.2 AVA\_VAN.1.2E**

5394 **The evaluator shall perform a search of public domain sources to identify potential**  
5395 **vulnerabilities in the TOE.**

5396 **14.3.3.4.3 AVA\_VAN.1.3E**

5397 **The evaluator shall conduct penetration testing, based on the identified potential**  
5398 **vulnerabilities, to determine that the TOE is resistant to attacks performed by an attacker**  
5399 **possessing Basic attack potential.**

5400 **14.3.4 AVA\_VAN.2 Vulnerability analysis**

5401 Dependencies: ADV\_ARC.1 Security architecture description

5402 ADV\_FSP.2 Security-enforcing functional specification

5403 ADV\_TDS.1 Basic design

5404 AGD\_OPE.1 Operational user guidance

5405 AGD\_PRE.1 Preparative procedures

5406 **14.3.4.1 Objectives**

5407 A vulnerability analysis is performed by the evaluator to ascertain the presence of potential  
5408 vulnerabilities.

5409 The evaluator performs penetration testing, to confirm that the potential vulnerabilities cannot be  
5410 exploited in the operational environment for the TOE. Penetration testing is performed by the  
5411 evaluator assuming an attack potential of Basic.

5412 **14.3.4.2 Developer action elements**

5413 **14.3.4.2.1 AVA\_VAN.2.1D**

5414 The developer shall provide the TOE for testing.

5415 **14.3.4.2.2 AVA\_VAN.2.2D**

5416 **The developer shall provide a list of third party components included in the TOE and the TOE**  
5417 **delivery.**

5418 **14.3.4.3 Content and presentation elements**

5419 **14.3.4.3.1 AVA\_VAN.2.1C**

5420 The TOE shall be suitable for testing.

5421 **14.3.4.3.2 AVA\_VAN.2.2C**

5422 **The list of third party components shall include components provided by third parties, and that**  
5423 **are part of the TOE or otherwise part of the TOE delivery.**

5424 **14.3.4.4 Evaluator action elements**

5425 **14.3.4.4.1 AVA\_VAN.2.1E**

5426 The evaluator shall confirm that the information provided meets all requirements for content and  
5427 presentation of evidence.

5428 **14.3.4.4.2 AVA\_VAN.2.2E**

5429 The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in  
5430 the TOE **the components in the list of third party components, and specific IT products in the**  
5431 **environment that the TOE depends on.**

5432 **14.3.4.4.3 AVA\_VAN.2.3E**

5433 **The evaluator shall perform an independent vulnerability analysis of the TOE using the**  
5434 **guidance documentation, functional specification, TOE design and security architecture**  
5435 **description to identify potential vulnerabilities in the TOE.**

5436 **14.3.4.4.4 AVA\_VAN.2.4E**

5437 The evaluator shall conduct penetration testing, based on the identified potential vulnerabilities, to  
5438 determine that the TOE is resistant to attacks performed by an attacker possessing Basic attack  
5439 potential.

5440 **14.3.5 AVA\_VAN.3 Focused vulnerability analysis**

5441 Dependencies: ADV\_ARC.1 Security architecture description

5442 ADV\_FSP.4 Complete functional specification

5443 ADV\_TDS.3 Basic modular design

5444 ADV\_IMP.1 Implementation representation of the TSF

5445 AGD\_OPE.1 Operational user guidance

5446 AGD\_PRE.1 Preparative procedures

5447 ATE\_DPT.1 Testing: basic design

5448 **14.3.5.1 Objectives**

5449 A vulnerability analysis is performed by the evaluator to ascertain the presence of potential  
5450 vulnerabilities.

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 Enhanced-Basic.

5454 **14.3.5.2 Developer action elements**

5455 **14.3.5.2.1 AVA\_VAN.3.1D**

5456 The developer shall provide the TOE for testing.

5457 **14.3.5.2.2 AVA\_VAN.3.2D**

5458 The developer shall provide a list of third party components included in the TOE and the TOE delivery.

5459 **14.3.5.3 Content and presentation elements**

5460 **14.3.5.3.1 AVA\_VAN.3.1C**

5461 The TOE shall be suitable for testing.

5462 **14.3.5.3.2 AVA\_VAN.3.2C**

5463 The list of third party components shall include components provided by third parties, and that are  
5464 part of the TOE or otherwise part of the TOE delivery.

5465 **14.3.5.4 Evaluator action elements**

5466 **14.3.5.4.1 AVA\_VAN.3.1E**

5467 The evaluator shall confirm that the information provided meets all requirements for content and  
5468 presentation of evidence.

5469 **14.3.5.4.2 AVA\_VAN.3.2E**

5470 The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in  
5471 the TOE the components in the list of third party components, and specific IT products in the  
5472 environment that the TOE depends on.

5473 **14.3.5.4.3 AVA\_VAN.3.3E**

5474 The evaluator shall perform an independent, **focused** vulnerability analysis of the TOE using the  
5475 guidance documentation, functional specification, TOE design, security architecture description **and**  
5476 **implementation representation** to identify potential vulnerabilities in the TOE.

5477 **14.3.5.4.4 AVA\_VAN.3.4E**

5478 The evaluator shall conduct penetration testing, based on the identified potential vulnerabilities, to  
5479 determine that the TOE is resistant to attacks performed by an attacker possessing **Enhanced-Basic**  
5480 attack potential.

5481 **14.3.6 AVA\_VAN.4 Methodical vulnerability analysis**

5482 Dependencies: ADV\_ARC.1 Security architecture description

5483 ADV\_FSP.4 Complete functional specification

5484 ADV\_TDS.3 Basic modular design

5485 ADV\_IMP.1 Implementation representation of the TSF



5486	AGD_OPE.1 Operational user guidance
5487	AGD_PRE.1 Preparative procedures
5488	ATE_DPT.1 Testing: basic design
5489	<b>14.3.6.1 Objectives</b>
5490	A methodical vulnerability analysis is performed by the evaluator to ascertain the presence of
5491	potential vulnerabilities.
5492	The evaluator performs penetration testing, to confirm that the potential vulnerabilities cannot be
5493	exploited in the operational environment for the TOE. Penetration testing is performed by the
5494	evaluator assuming an attack potential of Moderate.
5495	<b>14.3.6.2 Developer action elements</b>
5496	<b>14.3.6.2.1 AVA_VAN.4.1D</b>
5497	The developer shall provide the TOE for testing.
5498	<b>14.3.6.2.2 AVA_VAN.4.2D</b>
5499	The developer shall provide a list of third party components included in the TOE and the TOE delivery.
5500	<b>14.3.6.3 Content and presentation elements</b>
5501	<b>14.3.6.3.1 AVA_VAN.4.1C</b>
5502	The TOE shall be suitable for testing.
5503	<b>14.3.6.3.2 AVA_VAN.4.2C</b>
5504	The list of third party components shall include components provided by third parties, and that are
5505	part of the TOE or otherwise part of the TOE delivery.
5506	<b>14.3.6.4 Evaluator action elements</b>
5507	<b>14.3.6.4.1 AVA_VAN.4.1E</b>
5508	The evaluator shall confirm that the information provided meets all requirements for content and
5509	presentation of evidence.
5510	<b>14.3.6.4.2 AVA_VAN.4.2E</b>
5511	The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in
5512	the TOE the components in the list of third party components, and specific IT products in the
5513	environment that the TOE depends on.
5514	<b>14.3.6.4.3 AVA_VAN.4.3E</b>
5515	The evaluator shall perform an independent, <b>methodical</b> vulnerability analysis of the TOE using the
5516	guidance documentation, functional specification, TOE design, security architecture description and
5517	implementation representation to identify potential vulnerabilities in the TOE.

5518 **14.3.6.4.4 AVA\_VAN.4.4E**

5519 The evaluator shall conduct penetration testing based on the identified potential vulnerabilities to  
5520 determine that the TOE is resistant to attacks performed by an attacker possessing **Moderate** attack  
5521 potential.

5522 **14.3.7 AVA\_VAN.5 Advanced methodical vulnerability analysis**

5523 Dependencies: ADV\_ARC.1 Security architecture description

5524 ADV\_FSP.4 Complete functional specification

5525 ADV\_TDS.3 Basic modular design

5526 ADV\_IMP.1 Implementation representation of the TSF

5527 AGD\_OPE.1 Operational user guidance

5528 AGD\_PRE.1 Preparative procedures

5529 ATE\_DPT.1 Testing: basic design

5530 **14.3.7.1 Objectives**

5531 A methodical vulnerability analysis is performed by the evaluator to ascertain the presence of  
5532 potential vulnerabilities.

5533 The evaluator performs penetration testing, to confirm that the potential vulnerabilities cannot be  
5534 exploited in the operational environment for the TOE. Penetration testing is performed by the  
5535 evaluator assuming an attack potential of High.

5536 **14.3.7.2 Developer action elements**

5537 **14.3.7.2.1 AVA\_VAN.5.1D**

5538 The developer shall provide the TOE for testing.

5539 **14.3.7.2.2 AVA\_VAN.5.2D**

5540 The developer shall provide a list of third party components included in the TOE and the TOE delivery.

5541 **14.3.7.3 Content and presentation elements**

5542 **14.3.7.3.1 AVA\_VAN.5.1C**

5543 The TOE shall be suitable for testing.

5544 **14.3.7.3.2 AVA\_VAN.2.2C**

5545 The list of third party components shall include components provided by third parties, and that are  
5546 part of the TOE or otherwise part of the TOE delivery.

5547 **14.3.7.4 Evaluator action elements**

5548 **14.3.7.4.1 AVA\_VAN.5.1E**

5549 The evaluator shall confirm that the information provided meets all requirements for content and  
5550 presentation of evidence.

5551 **14.3.7.4.2 AVA\_VAN.5.2E**

5552 The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in  
5553 the TOE the components in the list of third party components, and specific IT products in the  
5554 environment that the TOE depends on.

5555 **14.3.7.4.3 AVA\_VAN.5.3E**

5556 The evaluator shall perform an independent, methodical vulnerability analysis of the TOE using the  
5557 guidance documentation, functional specification, TOE design, security architecture description and  
5558 implementation representation to identify potential vulnerabilities in the TOE.

5559 **14.3.7.4.4 AVA\_VAN.5.4E**

5560 The evaluator shall conduct penetration testing based on the identified potential vulnerabilities to  
5561 determine that the TOE is resistant to attacks performed by an attacker possessing **High** attack  
5562 potential.

5563 **14.4 Composite vulnerability assessment (AVA\_COMP)**

5564 **14.4.1 Objectives**

5565 The aim of this activity is to determine the exploitability of flaws or weaknesses in the composite TOE  
5566 as a whole in the intended environment.

5567 **14.4.2 AVA\_COMP.1 Composite product vulnerability assessment**

5568 Dependencies: No dependencies

5569 **14.4.2.1 Application notes**

5570 This activity focuses exclusively on vulnerability assessment of the composite product *as a whole* and  
5571 represents merely *partial efforts* within the general approach being covered by the standard<sup>13</sup>  
5572 assurance family of the class AVA: AVA\_VAN.

5573 The results of the vulnerability assessment for the underlying platform represented in the ETR\_COMP  
5574 can be reused under the following conditions: they are up to date and all composite activities for  
5575 correctness – ASE\_COMP.1, ALC\_COMP.1, ADV\_COMP.1 and ATE\_COMP.1 – are finalised with the  
5576 verdict PASS.

5577 The yellow marked references need to be aligned with CEM. This will be done later on if the necessary  
5578 content has been included in CEM.

<sup>13</sup> i.e. as defined by ISO/IEC 18045

5579 Due to composing of the platform and the application a new quality arises, which can cause additional  
 5580 vulnerabilities of the platform which might be not mentioned in the ETR\_COMP. In these  
 5581 circumstances [R44] in chapter **Fehler! Verweisquelle konnte nicht gefunden werden.D.3** applies.

Kommentiert [NC3]: Überarbeiten, da der Annex nach 18045 verschoben wird

#### 5582 14.4.2.2 Developer action elements

##### 5583 14.4.2.2.1 AVA\_COMP.1.1D

5584 The developer shall provide the composite TOE for penetrating testing.

#### 5585 14.4.2.3 Content and presentation elements

##### 5586 14.4.2.3.1 AVA\_COMP.1.1C

5587 The composite TOE provided shall be suitable for testing as a whole.

#### 5588 14.4.2.4 Evaluator action elements

##### 5589 14.4.2.4.1 AVA\_COMP.1.1E

5590 The evaluator shall conduct penetration testing of the composite product *as a whole* building on  
 5591 evaluator's own vulnerability analysis, to ensure that the vulnerabilities being relevant for the  
 5592 Composite-ST are not exploitable.

### 5593 15 Class ACO: Composition

#### 5594 15.1 Introduction

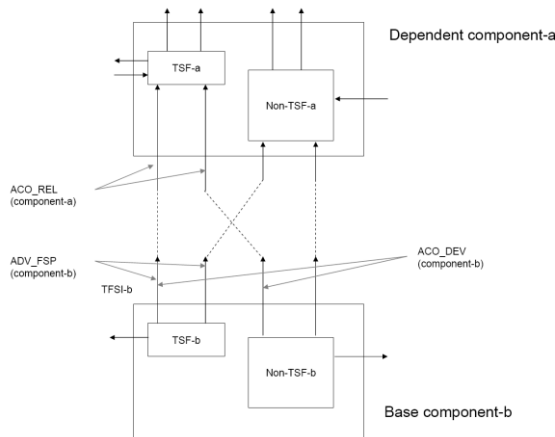
5595 The class ACO: Composition encompasses five families. These families specify assurance requirements  
 5596 that are designed to provide confidence that a composed TOE will operate securely when relying upon  
 5597 security functionality provided by previously evaluated software, firmware or hardware components.

5598 Composition involves taking two or more IT entities successfully evaluated against the ISO/IEC 15408  
 5599 series security assurance requirements packages (base components and dependent components, see  
 5600 Annex B) and combining them for use, with no further development of either IT entity. The  
 5601 development of additional IT entities is not included (entities that have not previously been the  
 5602 subject of a component evaluation). The composed TOE forms a new product that can be installed and  
 5603 integrated into any specific environment instance that meets the objectives for the environment.

5604 This approach does not provide an alternative approach for the evaluation of components.  
 5605 Composition under ACO provides a composed TOE integrator a method, which can be used as an  
 5606 alternative to other assurance levels specified in ISO/IEC 15408, to gain confidence in a TOE that is the  
 5607 combination of two or more successfully evaluated components without having to re-evaluate the  
 5608 composite TSF. (The composed TOE integrator is referred to as "developer" throughout the ACO class,  
 5609 with any references to the developer of the base or dependent components clarified as such.)

5610 Composed Assurance Packages, as defined in part 5 provide an assurance scale for composed TOEs.  
 5611 This assurance scale is required in addition to other assurance packages, for example the EALs,  
 5612 because to combine components evaluated against another assurance package and gain equivalent  
 5613 assurance in the resulting composed TOE, all SARs have to be applied to the composed TOE.  
 5614 Although reuse can be made of the component TOE evaluation results, there are often additional  
 5615 aspects of the components that have to be considered in the composed TOE, as described in Annex B.3.  
 5616 Due to the different parties involved in a composed TOE evaluation activity it is generally not possible  
 5617 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 14 — Relationship between ACO families and interactions between components**

In a composed TOE it is generally the case that one component relies on the services provided by another component. The component requiring services is termed the dependent component and the component providing the services is termed the base component. This interaction and distinct is discussed further in Annex B. It is assumed to be the case that the developer of the dependent component is supporting the composed TOE evaluation in some manner (as developer, sponsor, or just cooperating and providing the necessary evaluation evidence from the dependent component evaluation) The ACO components included in the CAP assurance packages should not be used as augmentations for component TOE evaluations, as this would provide no meaningful assurance for the component.

The families within the ACO class interact in a similar manner to the ADV, ATE and AVA classes in a component TOE evaluation and hence leverage from the specification of requirements from those classes where applicable. There are however a few items specific to composed TOE evaluations. To determine how the components interact and identify any deviations from the evaluations of the components, the dependencies that the dependent component has upon the underlying base component are identified (ACO\_REL). This reliance on the base component is specified in terms of the interfaces through which the dependent component makes calls for services in support of the dependent component SFRs. The interfaces, and at higher levels the supporting behaviour, provided by the base component in response to those service requests are analysed in ACO\_DEV. The ACO\_DEV family is based on the ADV\_TDS family, as at the simplest level the TSF of each component can be viewed as a subsystem of the composed TOE, with additional portions of each component seen as additional subsystems. Therefore, the interfaces between the components are seen as interactions between subsystems in a component TOE evaluation.

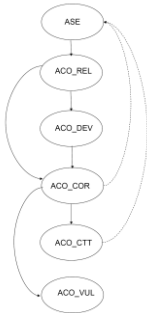
It is possible that the interfaces and supporting behaviour descriptions provided for ACO\_DEV are incomplete. This is determined during the conduct of ACO\_COR. The ACO\_COR family takes the outputs of ACO\_REL and ACO\_DEV and determines whether the components are being used in their evaluated

5647 configuration and identifies where any specifications are incomplete, which are then identified as  
5648 inputs into testing (ACO\_CTT) and vulnerability analysis (ACO\_VUL) activities of the composed TOE.

5649 Testing of the composed TOE is performed to determine that the composed TOE exhibits the expected  
5650 behaviour as determined by the composed TOE SFRs, and at higher levels demonstrates the  
5651 compatibility of the interfaces between the components of the composed TOE.

5652 The vulnerability analysis of the composed TOE leverages from the outputs of the vulnerability  
5653 analysis of the component evaluations. The composed TOE vulnerability analysis considers any  
5654 residual vulnerabilities from the component evaluations to determine that the residual vulnerabilities  
5655 are not applicable to the composed TOE. A search of publicly available information relating to the  
5656 components is also performed to identify any issues reported in the components since the completion  
5657 of the respective evaluations.

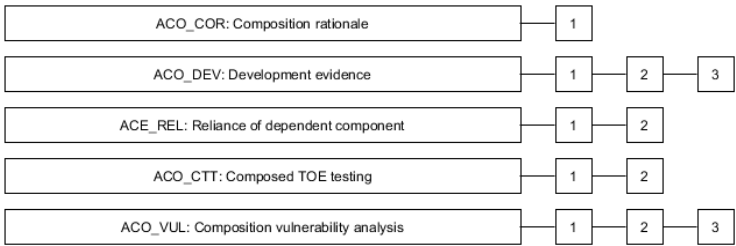
5658 The interaction between the ACO families is depicted in Figure 15 below. This shows by solid arrowed  
5659 lines where the evidence and understanding gained in one family feeds into the next activity and the  
5660 dashed arrows identify where an activity explicitly traces back to the composed TOE SFRs, as  
5661 described above.



5662  
5663

Figure 15 — Relationship between ACO families

5664 Further discussion of the definition and interactions within composed TOEs is provided in Annex B.  
5665 Figure 16 shows the families within this class, and the hierarchy of components within the families.



5666  
5667

Figure 16 — ACO: Composition class decomposition

5668 **15.2 Composition rationale (ACO\_COR)**

5669 **15.2.1 Objectives**

5670 This family addresses the requirement to demonstrate that the base component can provide an  
5671 appropriate level of assurance for use in composition.

5672 **15.2.2 Component levelling**

5673 There is only a single component in this family.

5674 **15.2.3 ACO\_COR.1 Composition rationale**

5675 Dependencies: ACO\_DEV.1 Functional Description

5676 ALC\_CMC.1 Labelling of the TOE

5677 ACO\_REL.1 Basic reliance information

5678 **15.2.3.1 Developer action elements**

5679 **15.2.3.1.1 ACO\_COR.1.1D**

5680 **The developer shall provide composition rationale for the base component.**

5681 **15.2.3.2 Content and presentation elements**

5682 **15.2.3.2.1 ACO\_COR.1.1C**

5683 **The composition rationale shall demonstrate that a level of assurance at least as high as that of**  
5684 **the dependent component has been obtained for the support functionality of the base**  
5685 **component, when the base component is configured as required to support the TSF of the**  
5686 **dependent component.**

5687 **15.2.3.3 Evaluator action elements**

5688 **15.2.3.3.1 ACO\_COR.1.1E**

5689 **The evaluator shall confirm that the information meets all requirements for content and**  
5690 **presentation of evidence.**

5691 **15.3 Development evidence (ACO\_DEV)**

5692 **15.3.1 Objectives**

5693 This family sets out requirements for a specification of the base component in increasing levels of  
5694 detail. Such information is required to gain confidence that the appropriate security functionality is  
5695 provided to support the requirements of the dependent component (as identified in the reliance  
5696 information).

5697 **15.3.2 Component levelling**

5698 The components are levelled on the basis of increasing amounts of detail about the interfaces  
5699 provided, and how they are implemented.

### 15.3.3 Application notes

The TSF of the base component is often defined without knowledge of the dependencies of the possible applications with which it may be composed. The TSF of this base component is defined to include all parts of the base component that have to be relied upon for enforcement of the base component SFRs. This will include all parts of the base component required to implement the base component SFRs.

The functional specification of the base component will describe the TSFI in terms of the interfaces the base component provides to allow an external entity to invoke operations of the TSF. This includes interfaces to the human user to permit interaction with the operation of the TSF invoking SFRs and also interfaces allowing an external IT entity to make calls into the TSF.

The functional specification only provides a description of what the TSF provides at its interface and the means by which that TSF functionality are invoked. Therefore, the functional specification does not necessarily provide a complete interface specification of all possible interfaces available between an external entity and the base component. It does not include what the TSF expects/requires from the operational environment. The description of what a dependent component TSF relies upon of a base component is considered in Reliance of dependent component (ACO\_REL) and the development information evidence provides a response to the interfaces specified.

The development information evidence includes a specification of the base component. This may be the evidence used during evaluation of the base component to satisfy the ADV requirements, or may be another form of evidence produced by either the base component developer or the composed TOE developer. This specification of the base component is used during Development evidence (ACO\_DEV) to gain confidence that the appropriate security functionality is provided to support the requirements of the dependent component. The level of detail required of this evidence increases to reflect the level of required assurance in the composed TOE. This is expected to broadly reflect the increasing confidence gained from the application of the assurance packages to the components. The evaluator determines that this description of the base component is consistent with the reliance information provided for the dependent component.

### 15.3.4 ACO\_DEV.1 Functional Description

Dependencies: ACO\_REL.1 Basic reliance information

#### 15.3.4.1 Objectives

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

#### 15.3.4.2 Developer action elements

##### 15.3.4.2.1 ACO\_DEV.1.1D

**The developer shall provide development information for the base component.**

#### 15.3.4.3 Content and presentation elements

##### 15.3.4.3.1 ACO\_DEV.1.1C

**The development information shall describe the purpose of each interface of the base component used in the composed TOE.**



5740 **15.3.4.3.2 ACO\_DEV.1.2C**

5741 **The development information shall show correspondence between the interfaces, used in the**  
5742 **composed TOE, of the base component and the dependent component to support the TSF of the**  
5743 **dependent component.**

5744 **15.3.4.4 Evaluator action elements**

5745 **15.3.4.4.1 ACO\_DEV.1.1E**

5746 **The evaluator shall confirm that the information meets all requirements for content and**  
5747 **presentation of evidence.**

5748 **15.3.4.4.2 ACO\_DEV.1.2E**

5749 **The evaluator shall determine that the interface description provided is consistent with the**  
5750 **reliance information provided for the dependent component.**

5751 **15.3.5 ACO\_DEV.2 Basic evidence of design**

5752 Dependencies: ACO\_REL.1 Basic reliance information

5753 **15.3.5.1 Objectives**

5754 A description of the interfaces in the base component, on which the dependent component relies, is  
5755 required. This is examined to determine whether or not it is consistent with the description of  
5756 interfaces on which the dependent component relies, as provided in the reliance information.

5757 In addition, the security behaviour of the base component that supports the dependent component  
5758 TSF is described.

5759 **15.3.5.2 Developer action elements**

5760 **15.3.5.2.1 ACO\_DEV.2.1D**

5761 The developer shall provide development information for the base component.

5762 **15.3.5.3 Content and presentation elements**

5763 **15.3.5.3.1 ACO\_DEV.2.1C**

5764 The development information shall describe the purpose **and method of use of** each interface of the  
5765 base component used in the composed TOE.

5766 **15.3.5.3.2 ACO\_DEV.2.2C**

5767 **The development information shall provide a high-level description of the behaviour of the**  
5768 **base component, which supports the enforcement of the dependent component SFRs.**

5769 **15.3.5.3.3 ACO\_DEV.2.3C**

5770 The development information shall show correspondence between the interfaces, used in the  
5771 composed TOE, of the base component and the dependent component to support the TSF of the  
5772 dependent component.

5773 **15.3.5.4 Evaluator action elements**

5774 **15.3.5.4.1 ACO\_DEV.2.1E**

5775 The evaluator shall confirm that the information meets all requirements for content and presentation  
5776 of evidence.

5777 **15.3.5.4.2 ACO\_DEV.2.2E**

5778 The evaluator shall determine that the interface description provided is consistent with the reliance  
5779 information provided for the dependent component.

5780 **15.3.6 ACO\_DEV.3 Detailed evidence of design**

5781 Dependencies: ACO\_REL.2 Reliance information

5782 **15.3.6.1 Objectives**

5783 A description of the interfaces in the base component, on which the dependent component relies, is  
5784 required. This is examined to determine whether or not it is consistent with the description of  
5785 interfaces on which the dependent component relies, as provided in the reliance information.

5786 The interface description of the architecture of the base component is provided to enable the evaluator  
5787 to determine whether or not that interface formed part of the TSF of the base component.

5788 **15.3.6.2 Developer action elements**

5789 **15.3.6.2.1 ACO\_DEV.3.1D**

5790 The developer shall provide development information for the base component.

5791 **15.3.6.3 Content and presentation elements**

5792 **15.3.6.3.1 ACO\_DEV.3.1C**

5793 The development information shall describe the purpose and method of use of each interface of the  
5794 base component used in the composed TOE.

5795 **15.3.6.3.2 ACO\_DEV.3.2C**

5796 **The development information shall identify the subsystems of the base component that**  
5797 **provide interfaces of the base component used in the composed TOE.**

5798 **15.3.6.3.3 ACO\_DEV.3.3C**

5799 The development information shall provide a high-level description of the behaviour of the base  
5800 component **subsystems**, which **support** the enforcement of the dependent component SFRs.

5801 **15.3.6.3.4 ACO\_DEV.3.4C**

5802 **The development information shall provide a mapping from the interfaces to the subsystems of**  
5803 **the base component.**

5804 **15.3.6.3.5 ACO\_DEV.3.5C**

5805 The development information shall show correspondence between the interfaces, used in the  
 5806 composed TOE, of the base component and the dependent component to support the TSF of the  
 5807 dependent component.

5808 **15.3.6.4 Evaluator action elements**5809 **15.3.6.4.1 ACO\_DEV.3.1E**

5810 The evaluator shall confirm that the information meets all requirements for content and presentation  
 5811 of evidence.

5812 **15.3.6.4.2 ACO\_DEV.3.2E**

5813 The evaluator shall determine that the interface description provided is consistent with the reliance  
 5814 information provided for the dependent component.

5815 **15.4 Reliance of dependent component (ACO\_REL)**5816 **15.4.1 Objectives**

5817 The purpose of this family is to provide evidence that describes the reliance that a dependent  
 5818 component has upon the base component. This information is useful to persons responsible for  
 5819 integrating the component with other evaluated IT components to form the composed TOE, and for  
 5820 providing insight into the security properties of the resulting composition.

5821 This provides a description of the interface between the dependent and base components of the  
 5822 composed TOE that may not have been analysed during evaluation of the individual components, as  
 5823 the interfaces were not TSFIs of the individual component TOEs.

5824 **15.4.2 Component levelling**

5825 The components in this family are levelled according to the amount of detail provided in the  
 5826 description of the reliance by the dependent component upon the base component.

5827 **15.4.3 Application notes**

5828 The Reliance of dependent component (ACO\_REL) family considers the interactions between the  
 5829 components where the dependent component relies upon a service from the base component to  
 5830 support the operation of security functionality of the dependent component. The interfaces into these  
 5831 services of the base component may not have been considered during evaluation of the base  
 5832 component because the service in the base component was not considered security-relevant in the  
 5833 component evaluation, either because of the inherent purpose of the service (e.g., adjust type font) or  
 5834 because associated ISO/IEC 15408-2 SFRs are not being claimed in the base component's ST (e.g. the  
 5835 login interface when no FIA: Identification and authentication SFRs are claimed). These interfaces into  
 5836 the base component are often viewed as functional interfaces in the evaluation of the base component,  
 5837 and are in addition to the security interfaces (TSFI) considered in the functional specification.

5838 In summary, the TSFIs described in the functional specification only include the calls made into a TSF  
 5839 by external entities and responses to those calls. Calls made by a TSF, which were not explicitly  
 5840 considered during evaluation of the components, are described by the reliance information provided  
 5841 to satisfy Reliance of dependent component (ACO\_REL).

5842 **15.4.4 ACO\_REL.1 Basic reliance information**

5843 Dependencies: No dependencies.

5844 **15.4.4.1 Developer action elements**

5845 **15.4.4.1.1 ACO\_REL.1.1D**

5846 The developer shall provide reliance information of the dependent component.

5847 **15.4.4.2 Content and presentation elements**

5848 **15.4.4.2.1 ACO\_REL.1.1C**

5849 The reliance information shall describe the functionality of the base component hardware,  
5850 firmware and/or software that is relied upon by the dependent component TSF.

5851 **15.4.4.2.2 ACO\_REL.1.2C**

5852 The reliance information shall describe all interactions through which the dependent  
5853 component TSF requests services from the base component.

5854 **15.4.4.2.3 ACO\_REL.1.3C**

5855 The reliance information shall describe how the dependent TSF protects itself from  
5856 interference and tampering by the base component.

5857 **15.4.4.3 Evaluator action elements**

5858 **15.4.4.3.1 ACO\_REL.1.1E**

5859 The evaluator shall confirm that the information provided meets all requirements for content  
5860 and presentation of evidence.

5861 **15.4.5 ACO\_REL.2 Reliance information**

5862 Dependencies: No dependencies.

5863 **15.4.5.1 Developer action elements**

5864 **15.4.5.1.1 ACO\_REL.2.1D**

5865 The developer shall provide reliance information of the dependent component.

5866 **15.4.5.2 Content and presentation elements**

5867 **15.4.5.2.1 ACO\_REL.2.1C**

5868 The reliance information shall describe the functionality of the base component hardware, firmware  
5869 and/or software that is relied upon by the dependent component TSF.

5870 **15.4.5.2.2 ACO\_REL.2.2C**

5871 The reliance information shall describe all interactions through which the dependent component TSF  
5872 requests services from the base component.

5873 **15.4.5.2.3 ACO\_REL.2.3C**

5874 **The reliance information shall describe each interaction in terms of the interface used and the**  
5875 **return values from those interfaces.**

5876 **15.4.5.2.4 ACO\_REL.2.4C**

5877 The reliance information shall describe how the dependent TSF protects itself from interference and  
5878 tampering by the base component.

5879 **15.4.5.3 Evaluator action elements**

5880 **15.4.5.3.1 ACO\_REL.2.1E**

5881 The evaluator shall confirm that the information provided meets all requirements for content and  
5882 presentation of evidence.

5883 **15.5 Composed TOE testing (ACO\_CTT)**

5884 **15.5.1 Objectives**

5885 This family requires that testing of composed TOE and testing of the base component, as used in the  
5886 composed TOE, is performed.

5887 **15.5.2 Component levelling**

5888 The components in this family are levelled on the basis of increasing rigour of interface testing and  
5889 increasing rigour of the analysis of the sufficiency of the tests to demonstrate that the composed TSF  
5890 operates in accordance with the reliance information and the composed TOE SFRs.

5891 **15.5.3 Application notes**

5892 There are two distinct aspects of testing associated with this family:

- 5893 a) testing of the interfaces between the base component and the dependent component, which the  
5894 dependent component rely upon for enforcement of security functionality, to demonstrate their  
5895 compatibility;
- 5896 b) testing of the composed TOE to demonstrate that the TOE behaves in accordance with the SFRs for  
5897 the composed TOE.

5898 If the test configurations used during evaluation of the dependent component included use of the base  
5899 component as a “platform” and the test analysis sufficiently demonstrates that the TSF behaves in  
5900 accordance with the SFRs, the developer need perform no further testing of the composed TOE  
5901 functionality. However, if the base component was not used in the testing of the dependent  
5902 component, or the configuration of either component varied, then the developer is to perform testing  
5903 of the composed TOE. This may take the form of repeating the dependent component developer  
5904 testing of the dependent component, provided this adequately demonstrates the composed TOE TSF  
5905 behaves in accordance with the SFRs.

5906 The developer is to provide evidence of testing the base component interfaces used in the  
5907 composition. The operation of base component TSFIs would have been tested as part of the ATE: Tests  
5908 activities during evaluation of the base component. Therefore, provided the appropriate interfaces  
5909 were included within the test sample of the base component evaluation and it was determined in  
5910 Composition rationale (ACO\_COR) that the base component is operating in accordance with the base

5911 component evaluated configuration, with all security functionality required by the dependent  
5912 component included in the TSF, the evaluator action ACO\_CTT.1.1E may be met through reuse of the  
5913 base component ATE: Tests verdicts.

5914 If this is not the case, the base component interfaces used relevant to the composition that are affected  
5915 by any variations to the evaluated configuration and any additional security functionally will be tested  
5916 to ensure they demonstrate the expected behaviour. The expected behaviour to be tested is that  
5917 described in the reliance information (Reliance of dependent component (ACO\_REL) evidence).

#### 5918 **15.5.4 ACO\_CTT.1 Interface testing**

5919 Dependencies: ACO\_REL.1 Basic reliance information

5920 ACO\_DEV.1 Functional Description

##### 5921 **15.5.4.1 Objectives**

5922 The objective of this component is to ensure that each interface of the base component, on which the  
5923 dependent component relies, is tested.

##### 5924 **15.5.4.2 Developer action elements**

###### 5925 **15.5.4.2.1 ACO\_CTT.1.1D**

5926 **The developer shall provide composed TOE test documentation.**

###### 5927 **15.5.4.2.2 ACO\_CTT.1.2D**

5928 **The developer shall provide base component interface test documentation.**

###### 5929 **15.5.4.2.3 ACO\_CTT.1.3D**

5930 **The developer shall provide the composed TOE for testing.**

###### 5931 **15.5.4.2.4 ACO\_CTT.1.4D**

5932 **The developer shall provide an equivalent set of resources to those that were used in the base**  
5933 **component developer's functional testing of the base component.**

##### 5934 **15.5.4.3 Content and presentation elements**

###### 5935 **15.5.4.3.1 ACO\_CTT.1.1C**

5936 **The composed TOE and base component interface test documentation shall consist of test**  
5937 **plans, expected test results and actual test results.**

###### 5938 **15.5.4.3.2 ACO\_CTT.1.2C**

5939 **The test documentation from the developer execution of the composed TOE tests shall**  
5940 **demonstrate that the TSF behaves as specified.**

- 5941 **15.5.4.3.3 ACO\_CTT.1.3C**
- 5942 **The test documentation from the developer execution of the base component interface tests**  
 5943 **shall demonstrate that the base component interface relied upon by the dependent component**  
 5944 **behaves as specified.**
- 5945 **15.5.4.3.4 ACO\_CTT.1.4C**
- 5946 **The base component shall be suitable for testing.**
- 5947 **15.5.4.4 Evaluator action elements**
- 5948 **15.5.4.4.1 ACO\_CTT.1.1E**
- 5949 **The evaluator shall confirm that the information provided meets all requirements for content**  
 5950 **and presentation of evidence.**
- 5951 **15.5.4.4.2 ACO\_CTT.1.2E**
- 5952 **The evaluator shall execute a sample of test in the test documentation to verify the developer**  
 5953 **test results.**
- 5954 **15.5.4.4.3 ACO\_CTT.1.3E**
- 5955 **The evaluator shall test a subset of the TSF interfaces of the composed TOE to confirm that the**  
 5956 **composed TSF operates as specified.**
- 5957 **15.5.5 ACO\_CTT.2 Rigorous interface testing**
- 5958 Dependencies: ACO\_REL.2 Reliance information
- 5959 ACO\_DEV.2 Basic evidence of design
- 5960 **15.5.5.1 Objectives**
- 5961 The objective of this component is to ensure that each interface of the base component, on which the  
 5962 dependent component relies, is tested.
- 5963 **15.5.5.2 Developer action elements**
- 5964 **15.5.5.2.1 ACO\_CTT.2.1D**
- 5965 The developer shall provide composed TOE test documentation.
- 5966 **15.5.5.2.2 ACO\_CTT.2.2D**
- 5967 The developer shall provide base component interface test documentation.
- 5968 **15.5.5.2.3 ACO\_CTT.2.3D**
- 5969 The developer shall provide the composed TOE for testing.
- 5970 **15.5.5.2.4 ACO\_CTT.2.4D**
- 5971 The developer shall provide an equivalent set of resources to those that were used in the base  
 5972 component developer's functional testing of the base component.

5973 **15.5.5.3 Content and presentation elements**

5974 **15.5.5.3.1 ACO\_CTT.2.1C**

5975 The composed TOE and base component interface test documentation shall consist of test plans,  
5976 expected test results and actual test results.

5977 **15.5.5.3.2 ACO\_CTT.2.2C**

5978 The test documentation from the developer execution of the composed TOE tests shall demonstrate  
5979 that the TSF behaves as specified **and is complete**.

5980 **15.5.5.3.3 ACO\_CTT.2.3C**

5981 The test documentation from the developer execution of the base component interface tests shall  
5982 demonstrate that the base component interface relied upon by the dependent component behaves as  
5983 specified **and is complete**.

5984 **15.5.5.3.4 ACO\_CTT.2.4C**

5985 The base component shall be suitable for testing.

5986 **15.5.5.4 Evaluator action elements**

5987 **15.5.5.4.1 ACO\_CTT.2.1E**

5988 The evaluator shall confirm that the information provided meets all requirements for content and  
5989 presentation of evidence.

5990 **15.5.5.4.2 ACO\_CTT.2.2E**

5991 The evaluator shall execute a sample of test in the test documentation to verify the developer test  
5992 results.

5993 **15.5.5.4.3 ACO\_CTT.2.3E**

5994 The evaluator shall test a subset of the TSF interfaces of the composed TOE to confirm that the  
5995 composed TSF operates as specified.

5996 **15.6 Composition vulnerability analysis (ACO\_VUL)**

5997 **15.6.1 Objectives**

5998 This family calls for an analysis of vulnerability information available in the public domain and of  
5999 vulnerabilities that may be introduced as a result of the composition.

6000 **15.6.2 Component levelling**

6001 The components in this family are levelled on the basis of increasing scrutiny of vulnerability  
6002 information from the public domain and independent vulnerability analysis.

6003 **15.6.3 Application notes**

6004 The developer will provide details of any residual vulnerabilities reported during evaluation of the  
6005 components. These may be gained from the component developers or evaluation reports for the



6006 components. These will be used as inputs into the evaluator's vulnerability analysis of the composed  
6007 TOE in the operational environment.

6008 The operational environment of the composed TOE is examined to ensure that the assumptions and  
6009 objectives for the component operational environment (specified in each component ST) are satisfied  
6010 in the composed TOE. An initial analysis of the consistency of assumptions and objectives between the  
6011 components and the composed TOE STs will have been performed during the conduct of the ASE  
6012 activities for the composed TOE. However, this analysis is revisited with the knowledge acquired  
6013 during the ACO\_REL, ACO\_DEV and the ACO\_COR activities to ensure that, for example, assumptions of  
6014 the dependent component that were addressed by the environment in the dependent component ST  
6015 are not reintroduced as a result of composition (i.e. that the base component adequately addresses the  
6016 assumptions of the dependent component ST in the composed TOE).

6017 A search by the evaluator for issues in each component will identify potential vulnerabilities reported  
6018 in the public domain since completion of the evaluation of the components. Any potential  
6019 vulnerabilities will then be subject to testing.

6020 If the base component used in the composed TOE has been the subject of assurance continuity  
6021 activities since certification, the evaluator will consider during the composed TOE vulnerability  
6022 analysis activities the changes made in base component.

#### 6023 **15.6.4 ACO\_VUL.1 Composition vulnerability review**

6024 Dependencies: ACO\_DEV.1 Functional Description

##### 6025 **15.6.4.1 Developer action elements**

###### 6026 **15.6.4.1.1 ACO\_VUL.1.1D**

6027 **The developer shall provide the composed TOE for testing.**

##### 6028 **15.6.4.2 Content and presentation elements**

###### 6029 **15.6.4.2.1 ACO\_VUL.1.1C**

6030 **The composed TOE shall be suitable for testing.**

##### 6031 **15.6.4.3 Evaluator action elements**

###### 6032 **15.6.4.3.1 ACO\_VUL.1.1E**

6033 **The evaluator shall confirm that the information provided meets all requirements for content**  
6034 **and presentation of evidence.**

###### 6035 **15.6.4.3.2 ACO\_VUL.1.2E**

6036 **The evaluator shall perform an analysis to determine that any residual vulnerabilities**  
6037 **identified for the base and dependent components are not exploitable in the composed TOE in**  
6038 **its operational environment.**

###### 6039 **15.6.4.3.3 ACO\_VUL.1.3E**

6040 **The evaluator shall perform a search of public domain sources to identify possible**  
6041 **vulnerabilities arising from use of the base and dependent components in the composed TOE**  
6042 **operational environment.**

6043 **15.6.4.3.4 ACO\_VUL.1.4E**

6044 **The evaluator shall conduct penetration testing, based on the identified vulnerabilities, to**  
6045 **demonstrate that the composed TOE is resistant to attacks by an attacker with basic attack**  
6046 **potential.**

6047 **15.6.5 ACO\_VUL.2 Composition vulnerability analysis**

6048 Dependencies: ACO\_DEV.2 Basic evidence of design

6049 **15.6.5.1 Developer action elements**

6050 **15.6.5.1.1 ACO\_VUL.2.1D**

6051 The developer shall provide the composed TOE for testing.

6052 **15.6.5.2 Content and presentation elements**

6053 **15.6.5.2.1 ACO\_VUL.2.1C**

6054 The composed TOE shall be suitable for testing.

6055 **15.6.5.3 Evaluator action elements**

6056 **15.6.5.3.1 ACO\_VUL.2.1E**

6057 The evaluator shall confirm that the information provided meets all requirements for content and  
6058 presentation of evidence.

6059 **15.6.5.3.2 ACO\_VUL.2.2E**

6060 The evaluator shall perform an analysis to determine that any residual vulnerabilities identified for  
6061 the base and dependent components are not exploitable in the composed TOE in its operational  
6062 environment.

6063 **15.6.5.3.3 ACO\_VUL.2.3E**

6064 The evaluator shall perform a search of public domain sources to identify possible vulnerabilities  
6065 arising from use of the base and dependent components in the composed TOE operational  
6066 environment.

6067 **15.6.5.3.4 ACO\_VUL.2.4E**

6068 **The evaluator shall perform an independent vulnerability analysis of the composed TOE, using**  
6069 **the guidance documentation, reliance information and composition rationale to identify**  
6070 **potential vulnerabilities in the composed TOE.**

6071 **15.6.5.3.5 ACO\_VUL.2.5E**

6072 The evaluator shall conduct penetration testing, based on the identified vulnerabilities, to demonstrate  
6073 that the composed TOE is resistant to attacks by an attacker with basic attack potential.

6074 **15.6.6 ACO\_VUL.3 Enhanced-Basic Composition vulnerability analysis**

6075 Dependencies: ACO\_DEV.3 Detailed evidence of design

6076 **15.6.6.1 Developer action elements**

6077 **15.6.6.1.1 ACO\_VUL.3.1D**

6078 The developer shall provide the composed TOE for testing.

6079 **15.6.6.2 Content and presentation elements**

6080 **15.6.6.2.1 ACO\_VUL.3.1C**

6081 The composed TOE shall be suitable for testing.

6082 **15.6.6.3 Evaluator action elements**

6083 **15.6.6.3.1 ACO\_VUL.3.1E**

6084 The evaluator shall confirm that the information provided meets all requirements for content and  
6085 presentation of evidence.

6086 **15.6.6.3.2 ACO\_VUL.3.2E**

6087 The evaluator shall perform an analysis to determine that any residual vulnerabilities identified for  
6088 the base and dependent components are not exploitable in the composed TOE in its operational  
6089 environment.

6090 **15.6.6.3.3 ACO\_VUL.3.3E**

6091 The evaluator shall perform a search of public domain sources to identify possible vulnerabilities  
6092 arising from use of the base and dependent components in the composed TOE operational  
6093 environment.

6094 **15.6.6.3.4 ACO\_VUL.3.4E**

6095 The evaluator shall perform an independent vulnerability analysis of the composed TOE, using the  
6096 guidance documentation, reliance information and composition rationale to identify potential  
6097 vulnerabilities in the composed TOE.

6098 **15.6.6.3.5 ACO\_VUL.3.5E**

6099 The evaluator shall conduct penetration testing, based on the identified vulnerabilities, to demonstrate  
6100 that the composed TOE is resistant to attacks by an attacker with **Enhanced-Basic** attack potential.

## Annex A (informative)

### Development (ADV)

This annex contains ancillary material to further explain and provide additional examples for the topics brought up in families of the ADV: Development class.

#### A.1 ADV\_ARC: Supplementary material on security architectures

A security architecture is a set of properties that the TSF exhibits; these properties include self-protection, domain separation, and non-bypassability. Having these properties provides a basis of confidence that the TSF is providing its security services. This annex provides additional material on these properties, as well as discussion on contents of a security architecture description.

The remainder of this subclause first explains these properties, then discusses the kinds of information that are needed to describe how the TSF exhibits those properties.

##### A.1.1 Security architecture properties

*Self-protection* refers to the ability of the TSF to protect itself from manipulation from external entities that may result in changes to the TSF. Without these properties, the TSF might be disabled from performing its security services.

It is oftentimes the case that a TOE uses services or resources supplied by other IT entities in order to perform its functions (e.g. an application that relies upon its underlying operating system). In these cases, the TSF does not protect itself entirely on its own, because it depends on the other IT entities to protect the services it uses.

*Domain separation* is a property whereby the TSF creates separate *security domains* for each untrusted active entity to operate on its resources, and then keeps those domains separated from one another so that no entity can run in the domain of any other. For example, an operating system TOE supplies a domain (address space, per-process environment variables) for each process associated with untrusted entities.

For some TOEs such domains do not exist because all of the actions of the untrusted entities are brokered by the TSF. A packet-filter firewall is an example of such a TOE, where there are no untrusted entity domains; there are only data structures maintained by the TSF. The existence of domains, then, is dependant upon 1) the type of TOE and 2) the SFRs levied on the TOE. In the cases where the TOE does provide domains for untrusted entities, this family requires that those domains are isolated from one another such that untrusted entities in one domain are prevented from tampering (affecting without brokering by the TSF) from another untrusted entity's domain.

*Non-bypassability* is a property that the security functionality of the TSF (as specified by the SFRs) is always invoked and cannot be circumvented when appropriate for that specific mechanism. For example, if access control to files is specified as a capability of the TSF via an SFR, there must be no interfaces through which files can be accessed without invoking the TSF's access control mechanism (an interface through which a raw disk access takes place might be an example of such an interface).

As is the case with self-protection, the very nature of some TOEs might depend upon their environments to play a role in non-bypassability of the TSF. For example, a security application TOE requires that it be invoked by the underlying operating system. Similarly, a firewall depends upon the fact that there are no direct connections between the internal and external networks and that all traffic between them must go through the firewall.

#### **A.1.2 Security architecture descriptions**

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

The security architecture description presents the TSF's properties of self-protection, domain separation, and non-bypassability in terms of the decomposition descriptions. The level of this description is commensurate with the TSF description required by the ADV\_FSP, ADV\_TDS and ADV\_IMP requirements that are being claimed. For example, if ADV\_FSP is the only TSF description available, it would be difficult to provide any meaningful security architecture description because none of the details of any internal workings of the TSF would be available.

However, if the TOE design were also available, even at the most basic level (ADV\_TDS.1), there would be some information available concerning the subsystems that make up the TSF, and there would be a description of how they work to implement self-protection, domain separation, and non-bypassability. For example, perhaps all user interaction with the TOE is constrained through a process that acts on that user's behalf, adopting all of the user's security attributes; the security architecture description would describe how such a process comes into being, how the process's behaviour is constrained by the TSF (so it cannot corrupt the TSF), how all actions of that process are mediated by the TSF (thereby explaining why the TSF cannot be bypassed), etc.

If the available TOE design is more detailed (e.g. at the modular level), or the implementation representation is also available, then the security architecture description would be correspondingly more detailed, explaining how the user's process communicate with the TSF processes, how different requests are processed by the TSF, what parameters are passed, what programmatic protections (buffer overflow prevention, parameter bounds checking, time of check/time of use checking, etc.) are in place. Similarly, a TOE whose ST claimed the ADV\_IMP component would go into implementation-specific detail.

The explanations provided in the security architecture description are expected to be of sufficient detail that one would be able to test their accuracy. That is, simple assertions (e.g. "The TSF keeps domains separate") provide no useful information to convince the reader that the TSF does indeed create and separate domains.

##### **A.1.2.1 Domain Separation**

In cases where the TOE exhibits domain separation entirely on its own, there would be a straightforward description of how this is attained. The security architecture description would explain the different kinds of domains that are defined by the TSF, how they are defined (i.e. what resources are allocated to each domain), how no resources are left unprotected, and how the domains are kept separated so that active entities in one domain cannot tamper with resources in another domain.

6183 For cases where the TOE depends upon other IT entities to play a role in domain separation, that  
 6184 sharing of roles must be made clear. For example, a TOE that is solely application software relies upon  
 6185 the underlying operating system to correctly instantiate the domains that the TOE defines; if the TOE  
 6186 defines separate processing space, memory space, etc, for each domain, it depends upon the  
 6187 underlying operating system to operate correctly and benignly (e.g. allow the process to execute only  
 6188 in the execution space that is requested by the TOE software).

6189 For example, mechanisms that implement domain separation (e.g., memory management, protected  
 6190 processing modes provided by the hardware, etc.) would be identified and described. Or, the TSF  
 6191 might implement software protection constructs or coding conventions that contribute to  
 6192 implementing separation of software domains, perhaps by delineating user address space from system  
 6193 address space.

6194 The vulnerability analysis and testing (see AVA\_VAN) activities will likely include attempts to defeat  
 6195 the described TSF domain separation through the use of monitoring or direct attack the TSF.

#### 6196 **A.1.2.2 TSF Self-protection**

6197 In cases where the TOE exhibits self-protection entirely on its own, there would be a straightforward  
 6198 description of how this self-protection is attained. Mechanisms that provide domain separation to  
 6199 define a TSF domain that is protected from other (user) domains would be identified and described.

6200 For cases where the TOE depends upon other IT entities to play a role in protecting itself, that sharing  
 6201 of roles must be made clear. For example, a TOE that is solely application software relies upon the  
 6202 underlying operating system to operate correctly and benignly; the application cannot protect itself  
 6203 against a malicious operating system that subverts it (for example, by overwriting its executable code  
 6204 or TSF data).

6205 The security architecture description also covers how user input is handled by the TSF in such a way  
 6206 that the TSF does not subject itself to being corrupted by that user input. For example, the TSF might  
 6207 implement the notion of privilege and protect itself by using privileged-mode routines to handle user  
 6208 data. The TSF might make use of processor-based separation mechanisms (e.g. privilege levels or  
 6209 rings) to separate TSF code and data from user code and data. The TSF might implement software  
 6210 protection constructs or coding conventions that contribute to implementing separation of software,  
 6211 perhaps by delineating user address space from system address space.

6212 For TOEs that start up in a low-function mode (for example, a single-user mode accessible only to  
 6213 installers or administrators) and then transition to the evaluated secure configuration (a mode  
 6214 whereby untrusted users are able to login and use the services and resources of the TOE), the security  
 6215 architecture description also includes an explanation of how the TSF is protected against this  
 6216 initialisation code that does not run in the evaluated configuration. For such TOEs, the security  
 6217 architecture description would explain what prevents those services that should be available only  
 6218 during initialisation (e.g. direct access to resources) from being accessible in the evaluated  
 6219 configuration. It would also explain what prevents initialisation code from running while the TOE is in  
 6220 the evaluated configuration.

6221 There must also be an explanation of how the trusted initialisation code will maintain the integrity of  
 6222 the TSF (and of its initialisation process) such that the initialisation process is able to detect any  
 6223 modification that would result in the TSF being spoofed into believe it was in an initial secure state.

6224 The vulnerability analysis and testing (see AVA\_VAN) activities will likely include attempts to defeat  
 6225 the described TSF self protection through the use of tampering, direct attack, or monitoring of the TSF.

### 6226 **A.1.2.3 TSF Non-Bypassability**

6227 The property of non-bypassability is concerned with interfaces that permit the bypass of the  
 6228 enforcement mechanisms. In most cases this is a consequence of the implementation, where if a  
 6229 programmer is writing an interface that accesses or manipulates an object, it is that programmer's  
 6230 responsibility to use interfaces that are part of the SFR enforcement mechanism for the object and not  
 6231 to try to circumvent those interfaces. For the description pertaining to non-bypassability, then, there  
 6232 are two broad areas that have to be covered.

6233 The first consists of those interfaces to the SFR-enforcement. The property for these interfaces is that  
 6234 they contain no operations or modes that allow them to be used to bypass the TSF. It is likely that the  
 6235 evidence for ADV\_FSP and ADV\_TDS can be used in large part to make this determination. Because  
 6236 non-bypassability is the concern, if only certain operations available through these TSFIs are  
 6237 documented (because they are SFR-enforcing) and others are not, the developer should consider  
 6238 whether additional information (to that presented in ADV\_FSP and ADV\_TDS) is necessary to make a  
 6239 determination that the SFR-supporting and SFR-non-interfering operations of the TSFI do not afford  
 6240 an untrusted entity the ability to bypass the policy being enforced. If such information is necessary, it  
 6241 is included in the security architecture description.

6242 The second area of non-bypassability is concerned with those interfaces whose interactions are not  
 6243 associated with SFR-enforcement. Depending on the ADV\_FSP and ADV\_TDS components claimed,  
 6244 some information about these interfaces may or may not exist in the functional specification and TOE  
 6245 design documentation. The information presented for such interfaces (or groups of interfaces) should  
 6246 be sufficient so that a reader can make a determination (at the level of detail commensurate with the  
 6247 rest of the evidence supplied in the ADV: Development class) that the enforcement mechanisms cannot  
 6248 be bypassed.

6249 The property that the security functionality cannot be bypassed applies to all security functionality  
 6250 equally. That is, the design description should cover objects that are protected under the SFRs (e.g.  
 6251 FDP\_\* components) and functionality (e.g., audit) that is provided by the TSF. The description should  
 6252 also identify the interfaces that are associated with security functionality; this might make use of the  
 6253 information in the functional specification. This description should also describe any design  
 6254 constructs, such as object managers, and their method of use. For instance, if routines are to use a  
 6255 standard macro to produce an audit record, this convention is a part of the design that contributes to  
 6256 the non-bypassability of the audit mechanism. It is important to note that *non-bypassability* in this  
 6257 context is not an attempt to answer the question "could a part of the TSF implementation, if malicious,  
 6258 bypass the security functionality", but rather to document how the implementation does not bypass  
 6259 the security functionality.

6260 The vulnerability analysis and testing (see AVA\_VAN) activities will likely include attempts to defeat  
 6261 the described non-bypassability by circumventing the TSF.

### 6262 **A.2ADV\_FSP: Supplementary material on functional specification**

6263 The purpose in specifying the TSFIs is to provide the necessary information to conduct testing;  
 6264 without knowing the possible means interact with the TSF, one cannot adequately test the behaviour  
 6265 of the TSF.

6266 There are two parts to specifying the TSFIs: identifying them and describing them. Because of the  
 6267 diversity of possible TOEs, and of different TSFs therein, there is no standard set of interfaces that  
 6268 constitute "TSFIs". This annex provides guidance on the factors that determine which interfaces are  
 6269 TSFIs.

6270 **A.2.1 Non-TSF part of the TOE**

6271 The TSF comprises all parts of the TOE the user has to rely on in order to trust the security  
6272 functionality.

6273 To say it in other words: Those parts of the TOE that do not belong to the TSF can be modified by an  
6274 attacker without any impact on the TOE security functionality. If this isn't the case, these parts of the  
6275 TOE have to be included in the TSF.

6276 If the TSF and the TSF implementation are defined then it is clear whether there exist further parts of  
6277 the TOE which can be classified as non-TSF parts of the TOE. Such parts do not have to be part of the  
6278 TSF but they are still part of the TOE.

6279 The relationship between TSF and non-TSF parts of TOE is given by their definitions and the ARC  
6280 properties as follows:

- 6281 • non-TSF parts do not bypass the TSF and
- 6282 • parts of the TSF protects themselves against tampering.

6283 A subsystem of the TOE which is not part of the TSF has to fulfil the following condition (described as a  
6284 rule of thumb<sup>14</sup>): The subsystem must not have any security impact of the TOE even if it were  
6285 substituted by an attacker.

6286 Therefore between the Non-TSF parts and the TSF parts it seems that some kind of "separation  
6287 mechanism" is advisable<sup>15</sup> because such "separation mechanism" may build the basis for the  
6288 assessment that there is no impact on the TSF parts from the Non-TSF parts possible.

6289 Such "separation mechanism" could be implemented by the security architecture or by an explicitly  
6290 realised part of the implementation (e.g. a firewall between TSF and Non-TSF parts of the TOE).

6291 The analysis of the "separation mechanism" is then subject of the vulnerability assessment because it  
6292 must withstand attacks by an attacker of the respective strength according to the VAN level of the  
6293 evaluation.

6294 The developer shall provide evidence for non-bypassability and self-protection in its security  
6295 architecture description and the evaluator shall analyse this evidence in subactivity for ADV\_ARC.1  
6296 and assess the effectiveness in the vulnerability assessment.

6297 The goal of TOE design documentation is to provide sufficient information to determine the TSF  
6298 boundary, and to describe how the TSF implements the SFR. Further attention is needed by the fact  
6299 that the family ADV\_TDS requires only identification of the non-TSF subsystems of the TOE. No  
6300 interface description is provided for these subsystems in ADV\_FSP or ADV\_TDS. SFR non-interference  
6301 of these subsystems is assumed but not demonstrated by the developer and not examined in details by  
6302 the evaluator. However from the TOE design point of view this is not that important as long as the  
6303 above mentioned separation mechanism is in place and the vulnerability assessment confirms that it is

---

<sup>14</sup> 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.

<sup>15</sup> 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.



strong enough. Therefore this "separation mechanism" implements the TSF or enforces ARC properties as security feature. But non-bypassability may be enforced by "pure architecture properties" as well.

Parts of the TOE classified as non-TSF must not provide means to bypass the TSF (no matter whether a valid user or even an attacker makes uses of those parts) and must not contribute to the TSF. It is important that the developer provides clear evidence and demonstrate how this requirement is fulfilled.

Therefore the developer shall demonstrate and the evaluator shall examine that the TOE identification of subsystems as non-TSF (cf. ADV\_TDS.x.1) is correct and consequently no detailed description of these subsystems is necessary. The evaluator examination shall include the ARC properties non-bypassability and self-protection being described in the ADV\_ARC documentation provided by the developer (see the paragraphs above).

#### **A.2.2 Determining the TSFI**

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

Once the TSF has been defined, the TSFI are identified. The TSFI consists of all means by which external entities (or subjects in the TOE but outside of the TSF) supply data to the TSF, receive data from the TSF and invoke services from the TSF. These service invocations and responses are the means of crossing the TSF boundary. While many of these are readily apparent, others might not be as obvious. The question that should be asked when determining the TSFIs is: "How can a potential attacker interact with the TSF in an attempt to subvert the SFRs?"

Therefore from the evaluation point of view it is also important whether the interface can be misused by an attacker to get access to the security functionality in order to compromise the assets protected by TSF.

Any interface of the TSF which can be potentially used by an attacker belongs to the TSFI (regardless of the further classification as SFR-enforcing, SFR-supporting or SFR-non-interfering).

It is not important whether the TSF will be accessed from outside or whether the TSF accesses the external resources (e.g. TSF calls platform or user). The only criteria is whether there is a potential interference with the TSF from outside.

The following discussions illustrate the application of the TSFI definition in different contexts.

##### **A.2.2.1 Electrical interfaces**

In TOEs such as smart cards, where the adversary has not only logical access to the TOE, but also complete physical access to the TOE, the TSF boundary is the physical boundary. Therefore, the exposed electrical interfaces are considered TSFI because their manipulation could affect the

6346 behaviour of the TSF. As such, all these interfaces (electrical contacts) need to be described: various  
6347 voltages that might be applied, etc.

#### 6348 A.2.2.2 Network protocol stack

6349 The TSFIs of a TOE that performs protocol processing would be those protocol layers to which a  
6350 potential attacker has direct access. This need not be the entire protocol stack, but it might be.

6351 For example, if the TOE were some sort of a network appliance that allowed potential attackers to  
6352 affect every level of the protocol stack (i.e. to send arbitrary signals, arbitrary voltages, arbitrary  
6353 packets, arbitrary datagrams, etc.), then the TSF boundary exists at each layer of the stack. Therefore,  
6354 the functional specification would have to address every protocol at every layer of the stack.

6355 If, however, the TOE were a firewall that protects an internal network from the Internet, a potential  
6356 attacker would have no means of directly manipulating the voltages that enter the TOE; any extreme  
6357 voltages would simply not be passed through the Internet. That is, the attacker would have access only  
6358 to those protocols at the Internet layer or above. The TSF boundary exists at each layer of the stack.  
6359 Therefore, the functional specification would have to address only those protocols at or above the  
6360 Internet layer: it would describe each of the different communication layers at which the firewall is  
6361 exposed in terms of what constitutes well-formed input for what might appear on the line, and the  
6362 result of both well-formed and malformed inputs. For example, the description of the Internet protocol  
6363 layer would describe what constitutes a well-formed IP packet and what happens when both  
6364 correctly-formed and malformed packets are received. Likewise, the description of the TCP layer  
6365 would describe a successful TCP connection and what happens both when successful connections are  
6366 established and when connections cannot be established or are inadvertently dropped. Presuming the  
6367 firewall's purpose is to filter application-level commands (like FTP or telnet), the description of the  
6368 application layer would describe the application-level commands that are recognised and filtered by  
6369 the firewall, as well as the results of encountering unknown commands.

6370 The descriptions of these layers would likely reference published communication standards (telnet,  
6371 FTP, TCP, etc.) that are used, noting which user-defined options are chosen.

### 6372 A.2.2.3 Wrappers

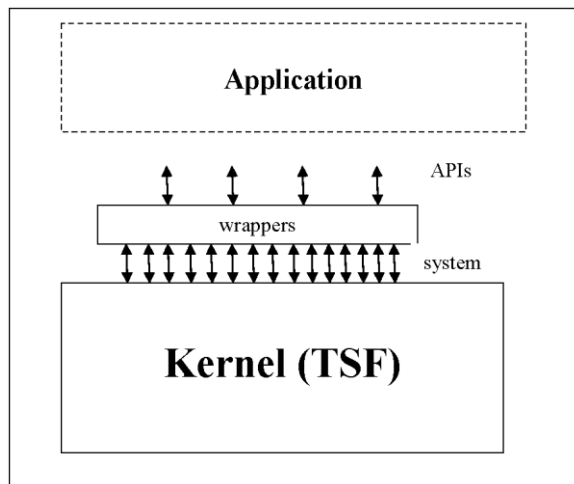


Figure A.1 — Wrappers

“Wrappers” translate complex series of interactions into simplified common services, such as when Operating Systems create APIs for use by applications (as shown in Figure A.1). Whether the TSFIs would be the system calls or the APIs depends upon what is available to the application: if the application can use the system calls directly, then the system calls are the TSFIs. If, however, there were something that prohibits their direct use and requires all communication through the APIs, then the APIs would be the TSFIs.

A Graphical User interface is similar: it translates between machine-understandable commands and user-friendly graphics. Similarly, the TSFIs would be the commands if users have access to them, or the graphics (pull-down menus, check-boxes, text fields) if the users are constrained to using them.

It is worth noting that, in both of these examples, if the user is prohibited from using the more primitive interfaces (i.e. the system calls or the commands), the description of this restriction and of its enforcement would be included in the Security Architecture Description (see A.1). Also, the wrapper would be part of the TSF.

### 6388 A.2.2.4 Inaccessible interfaces

For a given TOE, not all of the interfaces may be *accessible*. That is, the security objectives for the operational environment (in the Security Target) may prevent access to these interfaces or limit access in such a way that they are practically inaccessible. Such interfaces would not be considered TSFIs. Some examples:

- a) If the security objectives for the operational environment for the stand-alone firewall state that “the firewall will be operational in a server room environment to which only trusted and trained personnel will have access, and which will be equipped with an interruptible power supply (against power failure)”, physical and power interfaces will not be accessible, since trusted and trained personnel will not attempt to dismantle the firewall and/or disable its power supply.

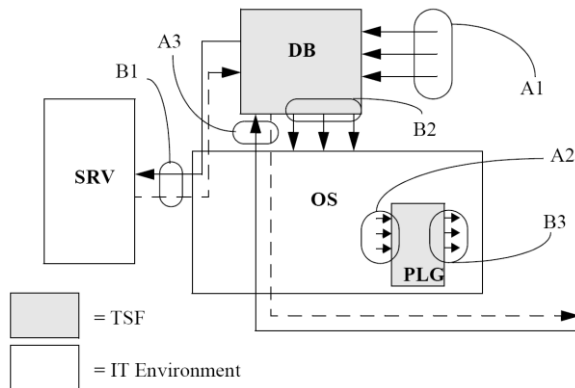
6398 b) If the security objectives for the operational environment for the software firewall (application)  
6399 state that “the OS and the hardware will provide a security domain for the application free from  
6400 tampering by other programs”, the interfaces through which the firewall can be accessed by other  
6401 applications on the OS (e.g. deleting or modifying the firewall executable, direct reading or writing  
6402 to the memory space of the firewall) will not be accessible, since the OS/hardware part of the  
6403 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.

6410 For all of these examples, these inaccessible interfaces would not be TSFIs.

### 6411 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<sub>x</sub>* for *TSFI*, and *B<sub>x</sub>* for other interfaces that would be documented in *ACO: Composition*. Each of these groups of interfaces is now discussed.

6428 Interface group A1 represents the most obvious set of TSFI. These are interfaces used by users to  
6429 directly access the database and its security functionality and resources.

6430 Interface group A2 represent the TSFI that the OS invokes to obtain the functionality provided by the  
6431 pluggable module. These are contrasted with interface group B3, which represent calls that the  
6432 pluggable module makes to obtain services from the IT environment.

6433 Interface group A3 represent TSFI that pass through the IT environment. In this case, the DBMS  
6434 communicates over the network using a proprietary application-level protocol. While the IT  
6435 environment is responsible for providing various supporting protocols (e.g., Ethernet, IP, TCP), the  
6436 application layer protocol that is used to obtain services from the DBMS is a TSFI and must be  
6437 documented as such. The dotted line indicates return values/services from the TSF over the network  
6438 connection.

6439 The interfaces labelled *Bx* represent interfaces to functionality in the IT Environment. These interfaces  
6440 are not TSFI and need only be discussed and analysed when the TOE is being used in a composite  
6441 evaluation as part of the activities associated with the ACO class.

#### 6442 **A.2.4 Example Functional Specification**

6443 The Example firewall is used between an internal network and an external network. It verifies the  
6444 source address of data received (to ensure that external data is not attempting to masquerade as  
6445 originating from the internal data); if it detects any such attempts, it saves the offending attempt to the  
6446 audit log. The administrator connects to the firewall by establishing a telnet connection to the firewall  
6447 from the internal network. Administrator actions consist of authenticating, changing passwords,  
6448 reviewing the audit log, and setting or changing the addresses of the internal and external networks.

6449 The Example firewall presents the following interfaces to the internal network:

6450 a) IP datagrams

6451 b) Administrator Commands

6452 and the following interfaces to the external network:

6453 a) IP datagrams

#### 6454 **Interfaces Descriptions: IP Datagrams**

6455 The datagrams are in the format specified by RFC 791.

6456 • Purpose - to transmit blocks of data ("datagrams") from source hosts to destination hosts  
6457 identified by fixed length addresses; also provides for fragmentation and reassembly of long  
6458 datagrams, if necessary, for transmission through small-packet networks.

6459 • Method of Use - they arrive from the lower-level (e.g. data link) protocol.

6460 • Parameters - the following fields of the IP datagram header: source address, destination address,  
6461 don't-fragment flag.

6462 • Parameter description - [As defined by RFC 791, subclause 3.1 ("Internet Header Format")]

6463 • Actions - Transmits datagrams that are not masquerading; fragments large datagrams if necessary;  
6464 reassembles fragments into datagrams.

6465 • Error messages - (none). No reliability guaranteed (reliability to be provided by upper-level  
6466 protocols) Undeliverable datagrams (e.g. must be fragmented for transmission, but don't-fragment  
6467 flag is set) dropped.

#### 6468 Interfaces Descriptions: Administrator Commands

6469 The administrator commands provide a means for the administrator to interact with the firewall.  
6470 These **commands and responses ride atop a telnet (RFC 854) connection established from any**  
6471 **host on the internal** network. Available commands are:

##### 6472 • **Passwd**

- 6473 • Purpose - sets administrator password
- 6474 • Method of Use - **Passwd** <password>
- 6475 • Parameters - password
- 6476 • Parameter description - value of new password
- 6477 • Actions - changes password to new value supplied. There are no restrictions.
- 6478 • Error messages - none.

##### 6479 • **Readaudit**

- 6480 • Purpose - presents the audit log to the administrator
- 6481 • Method of Use - **Readaudit**
- 6482 • Parameters - none
- 6483 • Parameter description - none
- 6484 • Actions - provides the text of the audit log
- 6485 • Error messages - none.

##### 6486 • **Setintaddr**

- 6487 • Purpose - sets the address of the internal address.
- 6488 • Method of Use - **Setintaddr** <address>
- 6489 • Parameters - address
- 6490 • Parameter description - first three fields of an IP address (as defined in RFC 791). For example:  
6491 123.123.123.
- 6492 • Actions - changes the internal value of the variable defining the internal network, the value of  
6493 which is used to judge attempted masquerades.

- 6494 • Error messages - "address in use": indicates the identified internal network is the same as the  
6495 external network.
- 6496 • **Setextaddr**
- 6497 • Purpose - sets the address of the external address
- 6498 • Method of Use - **Setextaddr** <address>
- 6499 • Parameters - address
- 6500 • Parameter description - first three fields of an IP address (as defined in RFC 791). For example:  
6501 123.123.123.
- 6502 • Actions - changes the internal value of the variable defining the external network.
- 6503 • Error messages - "address in use": indicates the identified external network is the same as the  
6504 internal network.

#### 6505 **A.3ADV\_INT: Supplementary material on TSF internals**

6506 The wide variety of TOEs makes it impossible to codify anything more specific than "well-structured"  
6507 or "minimum complexity". Judgements on structure and complexity are expected to be derived from  
6508 the specific technologies used in the TOE. For example, software is likely to be considered well-  
6509 structured if it exhibits the characteristics cited in the software engineering disciplines.

6510 This annex provides supplementary material on assessing the structure and complexity of procedure-  
6511 based software portions of the TSF. This material is based on information readily available in software  
6512 engineering literature. For other kinds of internals (e.g. hardware, non-procedural software such as  
6513 object-oriented code, etc.), corresponding literature on good practises should be consulted.

##### 6514 **A.3.1 Structure of procedural software**

6515 The structure of procedural software is traditionally assessed according to its *modularity*. Software  
6516 written with a modular design aids in achieving understandability by clarifying what dependencies a  
6517 module has on other modules (*coupling*) and by including in a module only tasks that are strongly  
6518 related to each other (*cohesion*). The use of modular design reduces the interdependence between  
6519 elements of the TSF and thus reduces the risk that a change or error in one module will have effects  
6520 throughout the TOE. Its use enhances clarity of design and provides for increased assurance that  
6521 unexpected effects do not occur. Additional desirable properties of modular decomposition are a  
6522 reduction in the amount of redundant or unneeded code.

6523 Minimising the amount of functionality in the TSF allows the evaluator as well as the developer to  
6524 focus only on that functionality which is necessary for SFR enforcement, contributing further to  
6525 understandability and further lowering the likelihood of design or implementation errors.

6526 The incorporation of modular decomposition, layering and minimisation into the design and  
6527 implementation process must be accompanied by sound software engineering considerations. A  
6528 practical, useful software system will usually entail some undesirable coupling among modules, some  
6529 modules that include loosely-related functions, and some subtlety or complexity in a module's design.  
6530 These deviations from the ideals of modular decomposition are often deemed necessary to achieve  
6531 some goal or constraint, be it related to performance, compatibility, future planned functionality, or  
6532 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.

a) *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.

### 6597 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.4ADV\_TDS: Subsystems and Modules

This subclause 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.

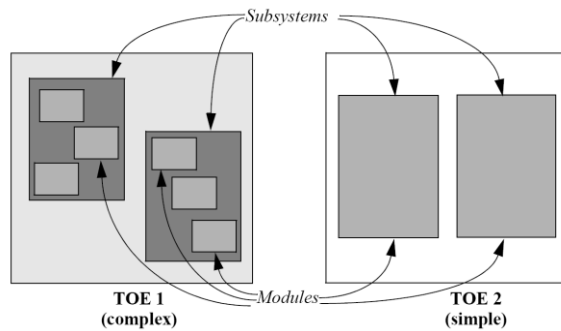


Figure A.3 — Subsystems and Modules

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.

6646 An *SFR-enforcing* subsystem is a subsystem that provides mechanisms for enforcing an element of any  
 6647 SFR, or directly supports a subsystem that is responsible for enforcing an SFR. If a subsystem provides  
 6648 (implements) an SFR-enforcing TSFI, then the subsystem is SFR-enforcing.

6649 Subsystems can also be identified as *SFR-supporting* and *SFR-non-interfering*. An SFR-supporting  
 6650 subsystem is one that is depended on by an SFR-enforcing subsystem in order to implement an SFR,  
 6651 but does not play as direct a role as an SFR-enforcing subsystem. An SFR-non-interfering subsystem is  
 6652 one that is not depended upon, in either a supporting or enforcing role, to implement an SFR.

#### 6653 A.4.2 Modules

6654 A module is generally a relatively small architectural unit that can be characterised in terms of the  
 6655 properties discussed in TSF internals (ADV\_INT). When both ADV\_TDS.3 Basic modular design (or  
 6656 above) requirements and TSF internals (ADV\_INT) requirements are present in a PP or ST, a “module”  
 6657 in terms of the TOE design (ADV\_TDS) requirements refers to the same entity as a “module” for the  
 6658 TSF internals (ADV\_INT) requirements. Unlike subsystems, modules describe the implementation in a  
 6659 level of detail that can serve as a guide to reviewing the implementation representation.

6660 It is important to note that, depending on the TOE, modules and subsystems may refer to the same  
 6661 abstraction. For ADV\_TDS.1 Basic design and ADV\_TDS.2 Architectural design (which do not require  
 6662 description at the module level) the subsystem description provides the lowest level detail available  
 6663 about the TSF. For ADV\_TDS.3 Basic modular design (which require module descriptions) these  
 6664 descriptions provide the lowest level of detail, while the subsystem descriptions (if they exist as  
 6665 separate entities) merely serve to put to the module descriptions in context. That is, it is not necessary  
 6666 to provide detailed subsystem descriptions if module descriptions exist. In TOEs that are sufficiently  
 6667 simple, a separate “subsystem description” is not necessary; the requirements can be met through  
 6668 documentation provided by modules. For complex TOEs, the purpose of the subsystem description  
 6669 (with respect to the TSF) is to provide the reader context so they can focus their analysis  
 6670 appropriately. This difference is illustrated in Figure A.3.

6671 An SFR-enforcing module is a module that completely or partially implements a security functional  
 6672 requirement (SFR) in the ST. Such modules may implement an SFR-enforcing TSFI, but some  
 6673 functionality expressed in an SFR (for example, audit and object re-use functionality) may not be  
 6674 directly tied to a single TSFI. As was the case with subsystems, SFR-supporting modules are those  
 6675 modules that are depended upon by an SFR-enforcing module, but are not responsible for directly  
 6676 implementing an SFR. SFR-non-interfering modules are those modules that do not deal, directly or  
 6677 indirectly, with the enforcement of SFRs.

6678 It is important to note that the determination of what “directly implements” means is somewhat  
 6679 subjective. In the narrowest sense of the term, it could be interpreted to mean the one or two lines of  
 6680 code that actually perform a comparison, zeroing operation, etc. that implements a requirement. A  
 6681 broader interpretation might be that it includes the module that is invoked in response to a SFR-  
 6682 enforcing TSFI, and all modules that may be invoked in turn by that module (and so on until the  
 6683 completion of the call). Neither of these interpretations is particularly satisfying, since the narrowness  
 6684 of the first interpretation may lead to important modules being incorrectly categorised as SFR  
 6685 supporting, while the second leads to modules that are actually not SFR-enforcing being classified as  
 6686 such.

6687 A description of a module should be such that one could create an implementation of the module from  
 6688 the description, and the resulting implementation would be 1) identical to the actual TSF  
 6689 implementation in terms of the interfaces presented, 2) identical in the use of interfaces that are  
 6690 mentioned in the design, and 3) functionally equivalent to the description of the purpose of the TSF  
 6691 module. For instance, RFC 793 provides a high-level description of the TCP protocol. It is necessarily

6692 implementation independent. While it provides a wealth of detail, it is **not** a suitable design  
 6693 description because it is not specific to an implementation. An actual implementation can add to the  
 6694 protocol specified in the RFC, and implementation choices (for example, the use of global data vs. local  
 6695 data in various parts of the implementation) may have an impact on the analysis that is performed.  
 6696 The design description of the TCP module would list the interfaces presented by the implementation  
 6697 (rather than just those defined in RFC 793), as well as an algorithm description of the processing  
 6698 associated with the modules implementing TCP (assuming they were part of the TSF).

6699 In the design, modules are described in detail in terms of the function they provide (the purpose); the  
 6700 interfaces they present (when required by the criteria); the return values from such interfaces; the  
 6701 interfaces (presented by other modules) they use (provided those interfaces are required to be also  
 6702 described); and a description of how they provide their functionality using a technique appropriate to  
 6703 the method used to implement the module.

6704 The purpose of a module should be described indicating what function the module is providing. It  
 6705 should be sufficient so that the reader could get a general idea of what the module's function is in the  
 6706 architecture.

6707 The interfaces presented by a module are those interfaces used by other modules to invoke the  
 6708 functionality provided. Interfaces include both *explicit* interfaces (e.g., a calling sequence invoked by  
 6709 other modules) as well as *implicit* interfaces (e.g., global data manipulated by the module). Interfaces  
 6710 are described in terms of how they are invoked, and any values that are returned. This description  
 6711 would include a list of parameters, and descriptions of these parameters. If a parameter were expected  
 6712 to take on a set of values (e.g., a "flag" parameter), the complete set of values the parameter could take  
 6713 on that would have an effect on module processing would be specified. Likewise, parameters  
 6714 representing data structures are described such that each field of the data structure is identified and  
 6715 described. Global data should be described to the extent required to understand their purpose. The  
 6716 level of description required for a global data structure needs to be identical to the one for module  
 6717 interfaces, where the input parameter and return values correspond to the individual fields and their  
 6718 possible values in the data structure. Global data structures may be described separate from the  
 6719 modules that manipulate or read them as long as the design of the modules contain sufficient  
 6720 information about the global data structures updated or the information extracted from global data  
 6721 structures.

6722 Note that different programming languages may have additional "interfaces" that would be non-  
 6723 obvious; an example would be operator/function overloading in C++. This "implicit interface" in the  
 6724 class description would also be described as part of the module design. Note that although a module  
 6725 could present only one interface, it is more common that a module presents a small set of related  
 6726 interfaces.

6727 When it is required to describe the interfaces used by a module, it must be clear from either the design  
 6728 description of the module or the purpose of the module called, what service is expected from the  
 6729 module called. For example if Module A is being described, and it uses Module B's bubble sort routine,  
 6730 the description of the interaction between modules must allow to identify why Module B's bubble sort  
 6731 routine is called and what this call contributes to the implementation of the SFRs. The interface and  
 6732 purpose of Module B's bubble sort routine must be described as part of the interfaces of Module B  
 6733 (provided the level of ADV\_TDS and the classification of Module B require a description its interfaces)  
 6734 and so Module A just needs to identify what data it needs to have sorted using this routine. An  
 6735 adequate description would be: "Module A invokes Module B's interface *double\_bubble()* to sort the  
 6736 usernames in alphabetical order".

6737 Note that if this sorting of the user names is not important for the enforcement of any SFR (e. g. it is  
 6738 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.

#### 6764 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.1A.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 <sup>16</sup>	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 <sup>17</sup>	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 <sup>18</sup>	purpose, all interfaces	purpose, all interfaces
ADV_TDS.6 Complete semiformal modular design with formal high-level design presentation	description, interactions	description, interactions	description, interactions	purpose, all interfaces	purpose, all interfaces	purpose, all interfaces

<sup>16</sup> *designation support* means that only documentation sufficient to support the classification of the subsystem / module is needed.

<sup>17</sup> *SFR interfaces* means that the module description contains, for each SFR-related interface, the returned values and the called interfaces to other modules.

<sup>18</sup> *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
(semiformal presentation; additional formal presentation)						

Table A.4.1 — 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.

6812 Therefore the analysis of non-interference during evaluation requires examination of the security  
 6813 architecture of the TOE (ADV\_ARC) and may need more information on non-TSF subsystems than only  
 6814 the TOE structure in terms of subsystems as provided for ADV\_TDS.x.1. The developers shall provide a  
 6815 rationale that TSF is correctly defined and the analysis of SFR-non-interfering module in terms of its  
 6816 purpose and interaction with other modules

- 6817 • **purpose:** how a module provides their functionality, no further design decisions are needed.
- 6818 • **interaction:** reason that subsystems or modules communicate, and characterizes the  
 6819 information that is passed (less details than for interfaces).

6820 During evaluation non-interference shall be analyzed as part of the examination of functional  
 6821 specification and TOE design, and the vulnerability analysis. The categorization of interfaces,  
 6822 subsystems and modules as SFR-enforcing, SFR-supporting and SFR-noninterfering implies specific  
 6823 examination of the functional specification, design and testing. An interpretation of TSFI as all  
 6824 accessible external interfaces of the TSF would help this analysis. The functional tests of all TSF  
 6825 subsystems (beginning with ATE\_DPT.1) and all TSF modules (ATE\_DPT.3 and higher) should provide  
 6826 evidence for the correctness of their security categorization.

#### 6827 A.5 Supplementary material on formal methods

6828 Formal methods provide a mathematical representation of the TSF and its behaviour and are required  
 6829 by the ADV\_FSP.6 Complete semi-formal functional specification with additional formal specification,  
 6830 ADV\_SPM.1 Formal TOE security policy model, and ADV\_TDS.6 Complete semiformal modular design  
 6831 with formal high-level design presentation components. There are two aspects of formal methods: the  
 6832 *specification language* that is used for formal expression, and the *theorem prover* that mathematically  
 6833 proves the completeness and correctness of the formal specification.

6834 A formal specification is expressed within a formal system based upon well-established mathematical  
 6835 concepts. These mathematical concepts are used to define well-defined semantics, syntax and rules of  
 6836 inference. A formal system is an abstract system of identities and relations that can be described by  
 6837 specifying a formal alphabet, a formal language over that alphabet which is based on a formal syntax,  
 6838 and a set of formal rules of inference for constructing derivations of sentences in the formal language.

6839 The evaluator should examine the identified formal systems to make sure that:

- 6840 • The semantics, syntax and inference rules of the formal system are defined or a definition is  
 6841 referenced.
- 6842 • Each formal system is accompanied by explanatory text that provides defined **semantics** so that:
  - 6843 1) the explanatory text provides defined meanings of terms, abbreviations and acronyms that  
 6844 are used in a context other than that accepted by normal usage;
  - 6845 2) the use of a formal system and semiformal notation use is accompanied by supporting  
 6846 explanatory text in informal style appropriate for unambiguous meaning;
  - 6847 3) the formal system is able to express rules and characteristics of applicable SFPs, security  
 6848 functionality and interfaces (providing details of effects, exceptions and error messages) of  
 6849 TSF, their subsystems or modules to be specified for the assurance family for which the  
 6850 notations are used;
  - 6851 4) the notation provides rules to determine the meaning of syntactical valid constructs.



- 6852 • Each formal system uses a formal syntax that provides rules to unambiguously recognise  
6853 constructs.
- 6854 • Each formal system provides proof rules which
- 6855 5) support logical reasoning of well-established mathematical concepts,
- 6856 6) help to prevent derivation of contradictions.
- 6857 If the developer uses a formal system which is already accepted by the evaluation authority the  
6858 evaluator can rely on the level of formality and strength of the system and focus on the instantiation of  
6859 the formal system to the TOE specifications and correspondence proofs.
- 6860 The formal style supports mathematical proofs of the security properties based on the security  
6861 features, the consistency of refinements and the correspondence of the representations. Formal tool  
6862 support seems adequate whenever manual derivations would otherwise become long winded and  
6863 incomprehensible. Formal tools are also apt to reduce the error probability inherent in manual  
6864 derivations.
- 6865 Examples of formal systems:
- 6866 • The **Z specification language** is highly expressive, and supports many different methods or styles  
6867 of formal specification. The use of Z has been predominantly for model-oriented specification,  
6868 using *schemas* to formally specify operations. See [https://en.wikipedia.org/wiki/Z\\_notation](https://en.wikipedia.org/wiki/Z_notation) for  
6869 more information.
- 6870 • **ACL2** is an open-source formal system comprising a LISP-based specification language and a  
6871 theorem prover. See <http://www.cs.utexas.edu/users/moore/acl2/> for further information.
- 6872 • **Isabelle** is a popular generic theorem proving environment that allows mathematical formulae to  
6873 be expressed in a formal language and provides tools for proving those formulae within a logical  
6874 calculus (see e.g. <http://www.cl.cam.ac.uk/Research/HVG/Isabelle/> for additional information).
- 6875 • The **B method** is a formal system based on the propositional calculus, the first order predicate  
6876 calculus with inference rules and set theory (see e.g. <https://en.wikipedia.org/wiki/B-Method> for  
6877 further information).
- 6878 • **NuSMV** (based on its predecessor SMV) is a symbolic model checker designed to be an open  
6879 architecture for model checking which can be reliably used for the verification of industrial  
6880 designs, as a core for custom verification tools, and as a testbed for formal verification techniques.  
6881 See <http://nusmv.fbk.eu/> for more information.
- 6882 • **Coq** is a formal proof management system that provides a formal language to write mathematical  
6883 definitions, executable algorithms and theorems together with an environment for semi-  
6884 interactive development of machine-checked proofs. See <https://coq.inria.fr/> for more  
6885 information.
- 6886 • **SystemVerilog** is a combined hardware description language and hardware verification language  
6887 based on Verilog.

## Annex B (informative)

### Composition (ACO)

The goal of this annex is to explain the concepts behind composition evaluations and the ACO criteria. This annex does not define the ASE criteria; this definition can be found in clause 9.

#### B.1 Necessity for composed TOE evaluations

The IT market is, on the whole, made up of vendors offering a particular type of product/technology. Although there is some overlap, where a PC hardware vendor may also offer application software and/or operating systems or a chip manufacturer may also develop a dedicated operating system for their own chipset, it is often the case that an IT solution is implemented by a variety of vendors.

There is sometimes a need for assurance in the combination (composition) of components in addition to the assurance of the individual components. Although there is cooperation between these vendors, in the dissemination of certain material required for the technical integration of the components, the agreements rarely stretch to the extent of providing detailed design information and development process/procedure evidence. This lack of information from the developer of a component on which another component relies means that the dependent component developer does not have access to the type of information necessary to perform an evaluation of both the dependent and base components at EAL2 or above. Therefore, while an evaluation of the dependent component can still be performed at any assurance level, to compose components with assurance at EAL2 or above it is necessary to reuse the evaluation evidence and results of evaluations performed for the component developer.

It is intended that the ACO criteria are applicable in the situation where one IT entity is dependent on another for the provision of security services. The entity providing the services is termed the "base component", and that receiving the services is termed the "dependent component". This relationship may exist in a number of contexts. For example, an application (dependent component) may use services provided by an operating system (base component). Alternatively, the relationship may be peer-to-peer, in the sense of two linked applications, either running in a common operating system environment, or on separate hardware platforms. If there is a dominant peer providing the services to the minor peer, the dominant peer is considered to be the base component and the minor peer the dependent component. If the peers provide services to each other in a mutual manner, each peer will be considered to be the base component for the services offered and dependent component for the services required. This will require iterations of the ACO components applying all requirements to each type of component peer.

The criteria are also intended to be more broadly applicable, stepwise (where a composed TOE comprised of a dependent component and a base component itself becomes the base component of another composed TOE), in more complex relationships, but this may require further interpretation.

It is still required for composed TOE evaluations that the individual components are evaluated independently, as the composition evaluation builds on the results of the individual component evaluations. The evaluation of the dependent component may still be in progress when the composed TOE evaluation commences. However, the dependent component evaluation must complete before the composed TOE evaluation completes.

6929 The composed evaluation activities may take place at the same time as the dependent component  
6930 evaluation. This is due to two factors:

6931 a) Economic/business drivers - the dependent component developer will either be sponsoring the  
6932 composition evaluation activities or supporting these activities as the evaluation deliverables from  
6933 the dependent component evaluation are required for composed evaluation activities.

6934 b) Technical drivers - the components consider whether the requisite assurance is provided by the  
6935 base component (e.g. considering the changes to the base component since completion of the  
6936 component evaluation) with the understanding that the dependent component has recently  
6937 undergone (is undergoing) component evaluation and all evaluation deliverables associated with  
6938 the evaluation are available. Therefore, there are no activities during composition requesting the  
6939 dependent component evaluation activities to be re-verified. Also, it is verified that the base  
6940 component forms (one of) the test configurations for the testing of the dependent component  
6941 during the dependent component evaluation, leaving ACO\_CTT to consider the base component in  
6942 this configuration.

6943 The evaluation evidence from the evaluation of the dependent component is required input into the  
6944 composed TOE evaluation activities. The only evaluation material from the evaluation of the base  
6945 component that is required as input into the composed TOE evaluation activities:

6946 a) Residual vulnerabilities in the base component, as reported during the base component  
6947 evaluation. This is required for the ACO\_VUL activities.

6948 No other evaluation evidence from the base component activities should be required for the composed  
6949 TOE evaluation, as the evaluation results from the component evaluation of the base component  
6950 should be reused. Additional information about the base component may be required if the composed  
6951 TOE TSF includes more of the base component than was considered to be TSF during component  
6952 evaluation of the base component.

6953 The component evaluation of the base and dependent components are assumed to be complete by the  
6954 time final verdicts are assigned for the ACO components.

6955 The ACO\_VUL components only consider resistance against an attacker with an attack potential up to  
6956 Enhanced-Basic. This is due to the level of design information that can be provided of how the base  
6957 component provides the services on which the dependent component relies through application of the  
6958 ACO\_DEV activities. Therefore, the confidence arising from composed TOE evaluations using CAPs is  
6959 limited to a level similar to that obtained from EAL4 component TOE evaluations. Although assurance  
6960 in the components that comprise the composed TOE may be higher than EAL4.

## 6961 **B.2 Performing Security Target evaluation for a composed TOE**

6962 An ST will be submitted by the developer for the evaluation of the composed (base component +  
6963 dependent component) TOE. This ST will identify the assurance package to be applied to the  
6964 composed TOE, providing assurance in the composed entity by drawing upon the assurance gained in  
6965 the component evaluations.

6966 The purpose of considering the composition of components within an ST is to validate the  
6967 compatibility of the components from the point of view of both the environment and the requirements,  
6968 and also to assess that the composed TOE ST is consistent with the component STs and the security  
6969 policies expressed within them. This includes determining that the component STs and the security  
6970 policies expressed within them are compatible.

6971 The composed TOE ST may refer out to the content of the component STs, or the ST author may chose  
6972 to reiterate the material of the component STs within the composed TOE ST providing a rationale of  
6973 how the component STs are represented in the composed TOE ST.

6974 During the conduct of the ASE\_CCL evaluation activities for a composed TOE ST the evaluator  
6975 determines that the component STs are accurately represented in the composed TOE ST. This is  
6976 achieved through determining that the composed TOE ST demonstrably conforms to the component  
6977 TOE STs. Also, the evaluator will need to determine that the dependencies of the dependent  
6978 component on the operational environment are adequately fulfilled in the composed TOE.

6979 The composed TOE description will describe the composed solution. The logical and physical scope  
6980 and boundary of the composed solution will be described, and the logical boundary(ies) between the  
6981 components will also be identified. The description will identify the security functionality to be  
6982 provided by each component.

6983 The statement of SFRs for the composed TOE will identify which component is to satisfy an SFR. If an  
6984 SFR is met by both components, then the statement will identify which component meets the different  
6985 aspects of the SFR. Similarly the composed TOE Summary Specification will identify which component  
6986 provides the security functionality described.

6987 The package of ASE: Security Target evaluation requirements applied to the composed TOE ST should  
6988 be consistent with the package of ASE: Security Target evaluation requirements used in the  
6989 component evaluations.

6990 Reuse of evaluation results from the evaluation of component STs can be made in the instances that  
6991 the composed TOE ST directly refers to the component STs. e.g. if the composed TOE ST refers to a  
6992 component ST for part of its statement of SFRs, the evaluator can understand that the requirement for  
6993 the completion of all assignment and selection operations (as stated in ASE\_REQ.\*.3C has been  
6994 satisfied in the component evaluations.

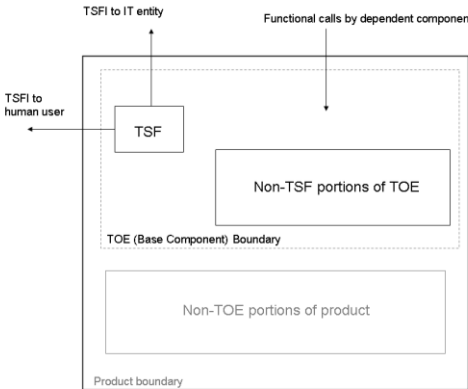
### 6995 **B.3 Interactions between composed IT entities**

6996 The TSF of the base component is often defined without knowledge of the dependencies of the  
6997 possible applications with which it may be composed. The TSF of this base component is defined to  
6998 include all parts of the base component that have to be relied upon for enforcement of the base  
6999 component SFRs. This will include all parts of the base component required to implement the base  
7000 component SFRs.

7001 The TSFI of this base component represents the interfaces provided by the TSF to the external entities  
7002 defined in the statement of SFRs to invoke a service of the TSF. This includes interfaces to the human  
7003 user and also interfaces to external IT entities. However, the TSFI only includes those interfaces to the  
7004 TSF, and therefore is not necessarily an exhaustive interface specification of all possible interfaces  
7005 available between an external entity and the base component. The base component may present  
7006 interfaces to services that were not considered security-relevant, either because of the inherent  
7007 purpose of the service (e.g., adjust type font) or because associated ISO/IEC 15408-2 SFRs are not  
7008 being claimed in the base component's ST (e.g. the login interface when no FIA: Identification and  
7009 authentication SFRs are claimed).

7010 The functional interfaces provided by the base component are in addition to the security interfaces  
7011 (TSFIs), and are not required to be considered during the base component evaluation. These often  
7012 include interfaces that are used by a dependent component to invoke a service provided by the base  
7013 component.

7014 The base component may include some indirect interfaces through which TSFIs may be called, e.g.  
7015 APIs that can be used to invoke a service of the TSF, which were not considered during the evaluation  
7016 of the base component.



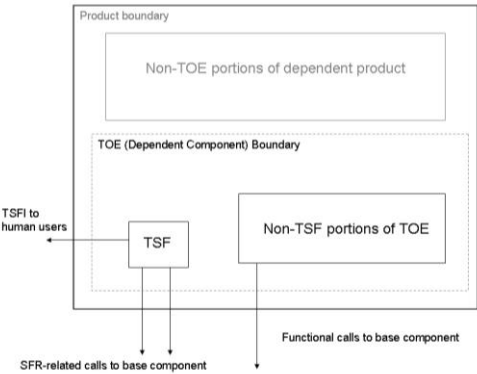
7017

7018 **Figure B.2B.1 — Base component abstraction**

7019 The dependent component, which relies on the base component, is similarly defined: interfaces to  
7020 external entities defined in the SFRs of the component ST are categorised as TSFI and are examined in  
7021 ADV\_FSP.

7022 Any call out from the dependent TSF to the environment in support of an SFR will indicate that the  
7023 dependent TSF requires some service from the environment in order to satisfy the enforcement of the  
7024 stated dependent component SFRs. Such a service is outside the dependent component boundary and  
7025 the base component is unlikely to be defined in the dependent ST as an external entity. Hence, the calls  
7026 for services made out by the dependent TSF to its underlying platform (the base component) will not  
7027 be analysed as part of the Functional specification (ADV\_FSP) activities. These dependencies on the  
7028 base component are expressed in the dependent component ST as security objectives for the  
7029 environment.

7030 This abstraction of the dependent component and the interfaces is shown in Figure B.2B.2 below.



**Figure B.3.2 — 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.

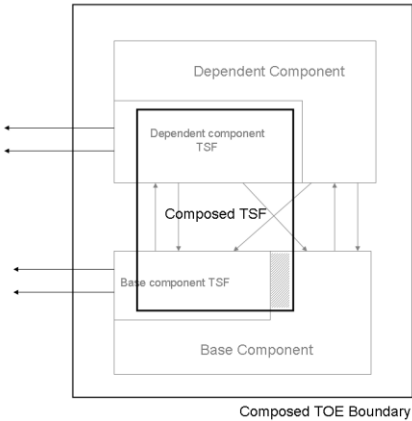


Figure B.3 — Composed TOE abstraction

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.

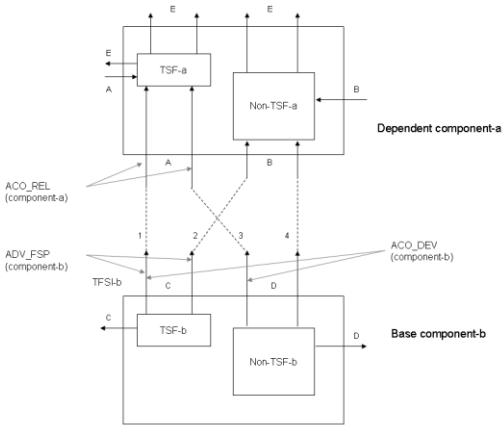


Figure B.4 — Composed component interfaces

- 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);

- 7060 b) Arrows coming *out* of 'base component-b' (C and D) = interfaces of services provided by the base  
7061 component to the environment;
- 7062 c) Broken lines between components = types of communication between pairs of interfaces;
- 7063 d) The other (grey) arrows = interfaces that are described by the given criteria.
- 7064 The following is a simplification, but explains the considerations that need to be made.
- 7065 There are components a ('dependent component-a') and b ('base component-b'): the arrows coming  
7066 *out* of TSF-a are services provided by TSF-a and are therefore TSFIs(a); likewise, the arrows coming  
7067 *out* of TSF-b ("C") are TSFIs(b). These are each detailed in their respective functional specs.  
7068 component-a is such that it requires services from its environment: those needed by the TSF(a) are  
7069 labelled "A"; the other (not related to TSF-a) services are labelled "B".
- 7070 When component-a and component-b are combined, there are four possible combinations of {services  
7071 needed by component-a} and {services provided by component-b}, shown as broken lines (types of  
7072 communication between pairs of interfaces). Any set of these might exist for a particular composition:
- 7073 a) TSF-a needs those services that are provided by TSF-b ("A" is connected to "C"): this is  
7074 straightforward: the details about "C" are in the FSP for component-b. In this instance the  
7075 interfaces should all be defined in the functional specifications for the component-b.
- 7076 b) Non-TSF-a needs those services that are provided by TSF-b ("B" is connected to "C"): this is  
7077 straightforward (again, the details about "C" are in the FSP for component-b), but unimportant:  
7078 security-wise.
- 7079 c) Non-TSF-a needs those services that are provided by non-TSF-b ("B" is connected to "D"): we have  
7080 no details about D, but there are no security implications about the use of these interfaces, so they  
7081 do not need to be considered in the evaluation, although they are likely to be an integration issue  
7082 for the developer.
- 7083 d) TSF-a needs those services that are provided by non-TSF-b ("A" is connected to "D"): this would  
7084 arise when component-a and component-b have different senses of what a "security service" is.  
7085 Perhaps component-b is making no claims about I&A (has no FIA SFRs in its ST), but component-a  
7086 needs authentication provided by its environment. There are no details about the "D" interfaces  
7087 available (they are not TSFI (b), so they are not in component-b's FSP).
- 7088 Note: if the kind of interaction described in case d above exists, then the TSF of the composed TOE  
7089 would be TSF-a + TSF-b + Non-TSF-b. Otherwise, the TSF of the composed TOE would be TSF-a + TSF-  
7090 b.
- 7091 Interfaces types 2 and 4 of Figure B.4 are not directly relevant to the evaluation of the composed TOE.  
7092 Interfaces 1 and 3 will be considered during the application of different families:
- 7093 a) Functional specification (ADV\_FSP) (for component-b) will describe the C interfaces.
- 7094 b) Reliance of dependent component (ACO\_REL) will describe the A interfaces.
- 7095 c) Development evidence (ACO\_DEV) will describe the C interfaces for connection type 1 and the D  
7096 interfaces for connection type 3.
- 7097 A typical example where composition may be applied is a database management system (DBMS) that  
7098 relies upon its underlying operating system (OS). During the evaluation of the DBMS component, there



7099 will be an assessment made of the security properties of that DBMS (to whatever degree of rigour is  
7100 dictated by the assurance components used in the evaluation): its TSF boundary will be identified, its  
7101 functional specification will be assessed to determine whether it describes the interfaces to the  
7102 security services provided by the TSF, perhaps additional information about the TSF (its design,  
7103 architecture, internal structure) will be provided, the TSF will be tested, aspects of its life-cycle and its  
7104 guidance documentation will be assessed, etc.

7105 However, the DBMS evaluation will not call for any evidence concerning the dependency the DBMS has  
7106 on the OS. The ST of the DBMS will most likely state assumptions about the OS in its Assumptions  
7107 subclause and state security objectives for the OS in its Environment subclause. The DBMS ST may  
7108 even instantiate those objectives for the environment in terms of SFRs for the OS. However, there will  
7109 be no specification for the OS that mirrors the detail in the functional specification, architecture  
7110 description, or other ADV evidence as for the DBMS. Reliance of dependent component (ACO\_REL) will  
7111 fulfil that need.

7112 Reliance of dependent component (ACO\_REL) describes the interfaces of the dependent TOE that make  
7113 the calls to the base component for the provision of services. These are the interfaces to which the  
7114 base component is to respond. The interface descriptions are provided from the dependent  
7115 component's viewpoint.

7116 Development evidence (ACO\_DEV) describes the interfaces provided by the base component, which  
7117 respond to the dependent component service requests. These interfaces are mapped to the relevant  
7118 dependent component interfaces that are identified in the reliance information. (The completeness of  
7119 this mapping, whether the base component interfaces described represent all dependent component  
7120 interfaces, is not verified here, but in Composition rationale (ACO\_COR)). At the higher levels of  
7121 ACO\_DEV the subsystems providing the interfaces are described.

7122 Any interfaces required by the dependent component that have not been described for the base  
7123 component are reported in the rationale for Composition rationale (ACO\_COR). The rationale also  
7124 reports whether the interfaces of the base component on which the dependent component relies were  
7125 considered within the base component evaluation. For any interfaces that were not considered in the  
7126 base component evaluation, a rationale is provided of the impact of using the interface on the base  
7127 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_CMC.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

