

<b>COMMITTEE DRAFT</b> <b>ISO/IEC CD 15408-3, revision</b>		Reference document: <b>SC 27 N18752</b>	
Date: <b>2018-06-25</b>		Supersedes document N18702, WG 3 N1469	
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 technology - Security techniques  Secretariat: Germany (DIN)		Circulated to P- and O-members, and to technical committees and organizations in liaison for comments by: <b>2018-08-20</b>  Please submit your comments via the online balloting application by the due date indicated.	
<b>ISO/IEC CD 15408-3, revision</b> <b>Title: IT Security techniques – Evaluation criteria for IT security — Part 3: Security assurance components</b> Project: 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 NP 15408-3 (revision)</b> <b>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]).
<b>ISO/IEC 15408-3 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 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. 2nd WD (WG 3 N1469).
<b>ISO/IEC 15408-3 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>CD Registration and Consideration</b> In accordance with resolution 6 (see SC 27 N18710) of the 30th SC 27 Plenary meeting held in Wuhan, China, 2018-04-23/24 the hereby attached document has been registered with the ISO Central Secretariat (ITTF) as 1st Committee Draft (CD) and is being circulated for a 1st CD 8 weeks letter ballot closing by  <b>2018-08-20</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 + 226			

Secretariat, ISO/IEC JTC 1/SC27 -

DIN Deutsches Institut für Normung e.V., Am DIN-Platz, Burggrafenstr. 6, D-10787 [D-10772 postal] Berlin , ermanyTelephone: + 49 2601-2652; Facsimile: + 49 2601-4-2652; E-mail: [krystyna.passia@din.de](mailto:krystyna.passia@din.de), <http://www.din.de/go/jtc1sc27>

Explanatory Report			
Status	SC 27 Decision	Reference documents	
		Input	Output
Study Period IT security testing, evaluation and assurance standards and techniques	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).

ISO/IEC 15408-3:XXXX(E)

ISO/IEC JTC 1/SC 27/WG 3 N1437

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

Case postale 401

CH-1214 Vernier, Geneva

Tel. + 41 22 749 01 11

Fax + 41 22 733 34 30

3, rue de Varembé

Case postale 131

CH-1211 Geneva 20

Tel. + 41 22 919 02 11

Fax + 41 22 919 03 00

E-mail [central@iso.org](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	<b>Contents</b>	Page
2	<b>1 Scope .....</b>	<b>1</b>
3	<b>2 Normative references .....</b>	<b>1</b>
4	<b>3 Terms and definitions, symbols and abbreviated terms.....</b>	<b>1</b>
5	<b>4 Overview .....</b>	<b>1</b>
6	<b>4.1 Organisation of this part of ISO/IEC 15408.....</b>	<b>1</b>
7	<b>5 Assurance paradigm.....</b>	<b>2</b>
8	<b>5.1 Introduction .....</b>	<b>2</b>
9	<b>5.2 ISO/IEC 15408 philosophy .....</b>	<b>2</b>
10	<b>5.3 Assurance approach .....</b>	<b>2</b>
11	<b>5.3.1 Introduction .....</b>	<b>2</b>
12	<b>5.3.2 Significance of vulnerabilities.....</b>	<b>2</b>
13	<b>5.3.3 Cause of vulnerabilities .....</b>	<b>3</b>
14	<b>5.3.4 ISO/IEC 15408 assurance .....</b>	<b>3</b>
15	<b>5.3.5 Assurance through evaluation.....</b>	<b>3</b>
16	<b>5.4 ISO/IEC 15408 evaluation assurance scale .....</b>	<b>4</b>
17	<b>6 Security assurance components.....</b>	<b>4</b>
18	<b>6.1 Security assurance classes, families and components structure .....</b>	<b>4</b>
19	<b>6.1.1 Assurance class structure .....</b>	<b>4</b>
20	<b>6.1.2 Assurance family structure.....</b>	<b>6</b>
21	<b>6.1.3 Assurance component structure .....</b>	<b>6</b>
22	<b>6.1.4 Assurance elements.....</b>	<b>8</b>
23	<b>6.1.5 Component taxonomy.....</b>	<b>9</b>
24	<b>7 Class APE: Protection Profile evaluation.....</b>	<b>9</b>
25	<b>7.1 Introduction .....</b>	<b>9</b>
26	<b>7.2 PP introduction (APE_INT).....</b>	<b>10</b>
27	<b>7.2.1 Objectives .....</b>	<b>10</b>
28	<b>7.2.2 APE_INT.1 PP introduction.....</b>	<b>10</b>
29	<b>7.3 Conformance claims (APE_CCL).....</b>	<b>10</b>
30	<b>7.3.1 Objectives .....</b>	<b>10</b>
31	<b>7.3.2 APE_CCL.1 Conformance claims.....</b>	<b>11</b>
32	<b>7.4 Security problem definition (APE_SPD) .....</b>	<b>12</b>
33	<b>7.4.1 Objectives .....</b>	<b>12</b>
34	<b>7.4.2 APE_SPD.1 Security problem definition .....</b>	<b>13</b>
35	<b>7.5 Security objectives (APE_OBJ).....</b>	<b>13</b>
36	<b>7.5.1 Objectives .....</b>	<b>13</b>
37	<b>7.5.2 Component levelling.....</b>	<b>13</b>
38	<b>7.5.3 APE_OBJ.1 Security objectives for the operational environment.....</b>	<b>13</b>
39	<b>7.5.4 APE_OBJ.2 Security objectives.....</b>	<b>14</b>
40	<b>7.6 Extended components definition (APE_ECD).....</b>	<b>15</b>
41	<b>7.6.1 Objectives .....</b>	<b>15</b>
42	<b>7.6.2 APE_ECD.1 Extended components definition .....</b>	<b>15</b>
43	<b>7.7 Security requirements (APE_REQ) .....</b>	<b>16</b>
44	<b>7.7.1 Objectives .....</b>	<b>16</b>
45	<b>7.7.2 Component levelling.....</b>	<b>16</b>
46	<b>7.7.3 APE_REQ.1 Stated security requirements.....</b>	<b>16</b>

47	7.7.4	APE_REQ.2 Derived security requirements.....	18
48	8	Class ACE: Protection Profile Configuration evaluation .....	20
49	8.1	Introduction .....	20
50	8.2	PP-Module introduction (ACE_INT).....	21
51	8.2.1	Objectives .....	21
52	8.2.2	ACE_INT.1 PP-Module introduction .....	21
53	8.3	PP-Module conformance claims (ACE_CCL).....	22
54	8.3.1	Objectives .....	22
55	8.3.2	ACE_CCL.1 PP-Module conformance claims.....	22
56	8.4	PP-Module Security problem definition (ACE_SPD).....	23
57	8.4.1	ACE_SPD.1 PP-Module Security problem definition.....	23
58	8.4.2	Application notes.....	23
59	8.5	PP-Module Security objectives (ACE_OBJ) .....	24
60	8.5.1	ACE_OBJ.1 PP-Module Security objectives .....	24
61	8.5.2	Application notes.....	24
62	8.6	PP-Module extended components definition (ACE_ECD).....	25
63	8.6.1	Objectives .....	25
64	8.6.2	ACE_ECD.1 PP-Module extended components definition.....	25
65	8.7	PP-Module security requirements (ACE_REQ) .....	26
66	8.7.1	Objectives .....	26
67	8.7.2	ACE_REQ.1 Stated security requirements .....	26
68	8.7.3	ACE_REQ.2 PP-Module security functional requirements .....	28
69	8.8	PP-Module consistency (ACE_MCO).....	30
70	8.8.1	Objectives .....	30
71	8.8.2	ACE_MCO.1 PP-Module consistency .....	30
72	8.9	PP-Configuration consistency (ACE_CCO) .....	31
73	8.9.1	Objectives .....	31
74	8.9.2	ACE_CCO.1 PP-Configuration consistency .....	31
75	9	Class ASE: Security Target evaluation .....	33
76	9.1	Introduction .....	33
77	9.2	ST introduction (ASE_INT).....	34
78	9.2.1	Objectives .....	34
79	9.2.2	ASE_INT.1 ST introduction.....	34
80	9.3	Conformance claims (ASE_CCL) .....	35
81	9.3.1	Objectives .....	35
82	9.3.2	ASE_CCL.1 Conformance claims .....	35
83	9.4	Security problem definition (ASE_SPD).....	37
84	9.4.1	Objectives .....	37
85	9.4.2	ASE_SPD.1 Security problem definition.....	37
86	9.5	Security objectives (ASE_OBJ) .....	38
87	9.5.1	Objectives .....	38
88	9.5.2	Component levelling.....	38
89	9.5.3	ASE_OBJ.1 Security objectives for the operational environment .....	38
90	9.5.4	ASE_OBJ.2 Security objectives .....	38
91	9.6	Extended components definition (ASE_ECD) .....	39
92	9.6.1	Objectives .....	39
93	9.6.2	ASE_ECD.1 Extended components definition.....	40
94	9.7	Security requirements (ASE_REQ).....	41
95	9.7.1	Objectives .....	41
96	9.7.2	Component levelling.....	41
97	9.7.3	ASE_REQ.1 Stated security requirements .....	41
98	9.7.4	ASE_REQ.2 Derived security requirements.....	42

99	<b>9.8</b>	<b>TOE summary specification (ASE_TSS) .....</b>	<b>44</b>	
100	<b>9.8.1</b>	<b>Objectives .....</b>	<b>44</b>	
101	<b>9.8.2</b>	<b>Component levelling.....</b>	<b>44</b>	
102	<b>9.8.3</b>	<b>ASE_TSS.1 TOE summary specification .....</b>	<b>44</b>	
103	<b>9.8.4</b>	<b>ASE_TSS.2 TOE summary specification with architectural design summary.....</b>	<b>45</b>	
104	<b>9.9</b>	<b>Consistency of composite product Security Target (ASE_COMP) .....</b>	<b>45</b>	
105	<b>9.9.1</b>	<b>Objectives .....</b>	<b>45</b>	
106	<b>9.9.2</b>	<b>ASE_COMP.1 .....</b>	<b>Consistency of Security Target</b>	<b>46</b>
107	<b>10</b>	<b>Class ADV: Development .....</b>	<b>48</b>	
108	<b>10.1</b>	<b>Introduction .....</b>	<b>48</b>	
109	<b>10.2</b>	<b>Security Architecture (ADV_ARC) .....</b>	<b>52</b>	
110	<b>10.2.1</b>	<b>Objectives .....</b>	<b>52</b>	
111	<b>10.2.2</b>	<b>Component levelling.....</b>	<b>53</b>	
112	<b>10.2.3</b>	<b>Application notes.....</b>	<b>53</b>	
113	<b>10.2.4</b>	<b>ADV_ARC.1 Security architecture description.....</b>	<b>53</b>	
114	<b>10.3</b>	<b>Functional specification (ADV_FSP) .....</b>	<b>54</b>	
115	<b>10.3.1</b>	<b>Objectives .....</b>	<b>54</b>	
116	<b>10.3.2</b>	<b>Component levelling.....</b>	<b>55</b>	
117	<b>10.3.3</b>	<b>Application notes.....</b>	<b>55</b>	
118	<b>10.3.4</b>	<b>ADV_FSP.1 Basic functional specification .....</b>	<b>57</b>	
119	<b>10.3.5</b>	<b>ADV_FSP.2 Security-enforcing functional specification.....</b>	<b>58</b>	
120	<b>10.3.6</b>	<b>ADV_FSP.3 Functional specification with complete summary .....</b>	<b>59</b>	
121	<b>10.3.7</b>	<b>ADV_FSP.4 Complete functional specification.....</b>	<b>60</b>	
122	<b>10.3.8</b>	<b>ADV_FSP.5 Complete semi-formal functional specification with additional error</b>		
123		<b>information.....</b>	<b>61</b>	
124	<b>10.3.9</b>	<b>ADV_FSP.6 Complete semi-formal functional specification with additional formal</b>		
125		<b>specification .....</b>	<b>63</b>	
126	<b>10.4</b>	<b>Implementation representation (ADV_IMP).....</b>	<b>64</b>	
127	<b>10.4.1</b>	<b>Objectives .....</b>	<b>64</b>	
128	<b>10.4.2</b>	<b>Component levelling.....</b>	<b>64</b>	
129	<b>10.4.3</b>	<b>Application notes.....</b>	<b>64</b>	
130	<b>10.4.4</b>	<b>ADV_IMP.1 Implementation representation of the TSF.....</b>	<b>65</b>	
131	<b>10.4.5</b>	<b>ADV_IMP.2 Complete mapping of the implementation representation of the TSF .....</b>	<b>66</b>	
132	<b>10.5</b>	<b>TSF internals (ADV_INT) .....</b>	<b>67</b>	
133	<b>10.5.1</b>	<b>Objectives .....</b>	<b>67</b>	
134	<b>10.5.2</b>	<b>Component levelling.....</b>	<b>67</b>	
135	<b>10.5.3</b>	<b>Application notes.....</b>	<b>67</b>	
136	<b>10.5.4</b>	<b>ADV_INT.1 Well-structured subset of TSF internals .....</b>	<b>68</b>	
137	<b>10.5.5</b>	<b>ADV_INT.2 Well-structured internals.....</b>	<b>69</b>	
138	<b>10.5.6</b>	<b>ADV_INT.3 Minimally complex internals .....</b>	<b>70</b>	
139	<b>10.6</b>	<b>Security policy modelling (ADV_SPM).....</b>	<b>71</b>	
140	<b>10.6.1</b>	<b>Objectives .....</b>	<b>71</b>	
141	<b>10.6.2</b>	<b>Component levelling.....</b>	<b>71</b>	
142	<b>10.6.3</b>	<b>Application notes.....</b>	<b>71</b>	
143	<b>10.6.4</b>	<b>ADV_SPM.1 Formal TOE security policy model.....</b>	<b>72</b>	
144	<b>10.7</b>	<b>TOE design (ADV_TDS).....</b>	<b>74</b>	
145	<b>10.7.1</b>	<b>Objectives .....</b>	<b>74</b>	
146	<b>10.7.2</b>	<b>Component levelling.....</b>	<b>74</b>	
147	<b>10.7.3</b>	<b>Application notes.....</b>	<b>74</b>	
148	<b>10.7.4</b>	<b>ADV_TDS.1 Basic design .....</b>	<b>76</b>	
149	<b>10.7.5</b>	<b>ADV_TDS.2 Architectural design .....</b>	<b>77</b>	
150	<b>10.7.6</b>	<b>ADV_TDS.3 Basic modular design .....</b>	<b>78</b>	

151	10.7.7 ADV_TDS.4 Semiformal modular design .....	79
152	10.7.8 ADV_TDS.5 Complete semiformal modular design.....	81
153	10.7.9 ADV_TDS.6 Complete semiformal modular design with formal high-level design	
154	presentation .....	82
155	10.8 Composite design compliance (ADV_COMP).....	84
156	10.8.1 Objectives .....	84
157	10.8.2 Component levelling.....	84
158	10.8.3 Application notes .....	84
159	10.8.4 ADV_COMP.1 Design compliance with the platform certification report, guidance and ETR_COMP 8	
160	11 Class AGD: Guidance documents .....	85
161	11.1 Introduction .....	85
162	11.2 Operational user guidance (AGD_OPE) .....	86
163	11.2.1 Objectives .....	86
164	11.2.2 Component levelling.....	86
165	11.2.3 Application notes .....	86
166	11.2.4 AGD_OPE.1 Operational user guidance .....	87
167	11.3 Preparative procedures (AGD_PRE) .....	88
168	11.3.1 Objectives .....	88
169	11.3.2 Component levelling.....	88
170	11.3.3 Application notes .....	88
171	11.3.4 AGD_PRE.1 Preparative procedures .....	88
172	12 Class ALC: Life-cycle support .....	89
173	12.1 Introduction .....	89
174	12.2 CM capabilities (ALC_CMC) .....	90
175	12.2.1 Objectives .....	90
176	12.2.2 Component levelling.....	91
177	12.2.3 Application notes .....	91
178	12.2.4 ALC_CMC.1 Labelling of the TOE .....	92
179	12.2.5 ALC_CMC.2 Use of the CM system.....	92
180	12.2.6 ALC_CMC.3 Authorisation controls .....	93
181	12.2.7 ALC_CMC.4 Production support, acceptance procedures and automation .....	95
182	12.2.8 ALC_CMC.5 Advanced support.....	97
183	12.3 CM scope (ALC_CMS) .....	100
184	12.3.1 Objectives .....	100
185	12.3.2 Component levelling.....	100
186	12.3.3 Application notes .....	100
187	12.3.4 ALC_CMS.1 TOE CM coverage .....	100
188	12.3.5 ALC_CMS.2 Parts of the TOE CM coverage .....	101
189	12.3.6 ALC_CMS.3 Implementation representation CM coverage.....	102
190	12.3.7 ALC_CMS.4 Problem tracking CM coverage .....	102
191	12.3.8 ALC_CMS.5 Development tools CM coverage.....	103
192	12.4 Delivery (ALC_DEL) .....	104
193	12.4.1 Objectives .....	104
194	12.4.2 Component levelling.....	105
195	12.4.3 Application notes .....	105
196	12.4.4 ALC_DEL.1 Delivery procedures .....	105
197	12.5 Developer environment security (ALC_DVS) .....	106
198	12.5.1 Objectives .....	106
199	12.5.2 Component levelling.....	106
200	12.5.3 Application notes .....	106
201	12.5.4 ALC_DVS.1 Identification of security controls .....	106
202	12.5.5 ALC_DVS.2 Sufficiency of security controls.....	107



203	<b>12.6</b>	<b>Flaw remediation (ALC_FLR)</b>	<b>108</b>	
204	<b>12.6.1</b>	<b>Objectives</b>	<b>108</b>	
205	<b>12.6.2</b>	<b>Component levelling</b>	<b>108</b>	
206	<b>12.6.3</b>	<b>Application notes</b>	<b>108</b>	
207	<b>12.6.4</b>	<b>ALC_FLR.1 Basic flaw remediation</b>	<b>108</b>	
208	<b>12.6.5</b>	<b>ALC_FLR.2 Flaw reporting procedures</b>	<b>109</b>	
209	<b>12.6.6</b>	<b>ALC_FLR.3 Systematic flaw remediation</b>	<b>110</b>	
210	<b>12.7</b>	<b>Development Life-cycle definition (ALC_LCD)</b>	<b>112</b>	
211	<b>12.7.1</b>	<b>Objectives</b>	<b>112</b>	
212	<b>12.7.2</b>	<b>Component levelling</b>	<b>112</b>	
213	<b>12.7.3</b>	<b>Application notes</b>	<b>113</b>	
214	<b>12.7.4</b>	<b>ALC_LCD.1 Developer defined life-cycle processes</b>	<b>113</b>	
215	<b>12.7.5</b>	<b>ALC_LCD.2 Measurable life-cycle model</b>	<b>114</b>	
216	<b>12.8</b>	<b>Practices for trustable development (ALC_PTD)</b>	<b>115</b>	
217	<b>12.8.1</b>	<b>Objectives</b>	<b>115</b>	
218	<b>12.8.2</b>	<b>Component levelling</b>	<b>116</b>	
219	<b>12.8.3</b>	<b>Application notes</b>	<b>116</b>	
220	<b>12.8.4</b>	<b>ALC_PTD.1 Uniquely identifying implementation representation</b>	<b>116</b>	
221	<b>12.8.5</b>	<b>ALC_PTD.2 Matching CMS scope of implementation representation</b>	<b>118</b>	
222	<b>12.8.6</b>	<b>ALC_PTD.3 Regenerate TOE with well-defined development tools</b>	<b>121</b>	
223	<b>12.9</b>	<b>Tools and techniques (ALC_TAT)</b>	<b>124</b>	
224	<b>12.9.1</b>	<b>Objectives</b>	<b>124</b>	
225	<b>12.9.2</b>	<b>Component levelling</b>	<b>124</b>	
226	<b>12.9.3</b>	<b>Application notes</b>	<b>124</b>	
227	<b>12.9.4</b>	<b>ALC_TAT.1 Well-defined development tools</b>	<b>124</b>	
228	<b>12.9.5</b>	<b>ALC_TAT.2 Compliance with implementation standards</b>	<b>125</b>	
229	<b>12.9.6</b>	<b>ALC_TAT.3 Compliance with implementation standards - all parts</b>	<b>126</b>	
230	<b>12.10</b>	<b>Integration of composition parts and consistency check of delivery procedures</b>		
231		<b>(ALC_COMP)</b>	<b>127</b>	
232	<b>12.10.1</b>	<b>Objectives</b>	<b>127</b>	
233	<b>12.10.2</b>	<b>Component levelling</b>	<b>127</b>	
234	<b>12.10.3</b>	<b>ALC_COMP.1</b>	<b>Inter</b>	
235	<b>13</b>	<b>Class ATE: Tests</b>	<b>128</b>	
236	<b>13.1</b>	<b>Introduction</b>	<b>128</b>	
237	<b>13.2</b>	<b>Coverage (ATE_COV)</b>	<b>129</b>	
238	<b>13.2.1</b>	<b>Objectives</b>	<b>129</b>	
239	<b>13.2.2</b>	<b>Component levelling</b>	<b>129</b>	
240	<b>13.2.3</b>	<b>Application notes</b>	<b>129</b>	
241	<b>13.2.4</b>	<b>ATE_COV.1 Evidence of coverage</b>	<b>129</b>	
242	<b>13.2.5</b>	<b>ATE_COV.2 Analysis of coverage</b>	<b>130</b>	
243	<b>13.2.6</b>	<b>ATE_COV.3 Rigorous analysis of coverage</b>	<b>131</b>	
244	<b>13.3</b>	<b>Depth (ATE_DPT)</b>	<b>131</b>	
245	<b>13.3.1</b>	<b>Objectives</b>	<b>131</b>	
246	<b>13.3.2</b>	<b>Component levelling</b>	<b>132</b>	
247	<b>13.3.3</b>	<b>Application notes</b>	<b>132</b>	
248	<b>13.3.4</b>	<b>ATE_DPT.1 Testing: basic design</b>	<b>132</b>	
249	<b>13.3.5</b>	<b>ATE_DPT.2 Testing: security enforcing modules</b>	<b>133</b>	
250	<b>13.3.6</b>	<b>ATE_DPT.3 Testing: modular design</b>	<b>134</b>	
251	<b>13.3.7</b>	<b>ATE_DPT.4 Testing: implementation representation</b>	<b>134</b>	
252	<b>13.4</b>	<b>Functional tests (ATE_FUN)</b>	<b>135</b>	
253	<b>13.4.1</b>	<b>Objectives</b>	<b>135</b>	
254	<b>13.4.2</b>	<b>Component levelling</b>	<b>136</b>	

255	13.4.3 Application notes .....	136
256	13.4.4 ATE_FUN.1 Functional testing .....	136
257	13.4.5 ATE_FUN.2 Ordered functional testing .....	137
258	13.5 Independent testing (ATE_IND) .....	138
259	13.5.1 Objectives .....	138
260	13.5.2 Component levelling .....	138
261	13.5.3 Application notes .....	138
262	13.5.4 ATE_IND.1 Independent testing - conformance .....	139
263	13.5.5 ATE_IND.2 Independent testing - sample .....	140
264	13.5.6 ATE_IND.3 Independent testing - complete .....	141
265	13.6 Composite functional testing (ATE_COMP) .....	142
266	13.6.1 Objectives .....	142
267	13.6.2 Application notes .....	142
268	13.6.3 ATE_COMP.1 Composite product functional testing .....	143
269	14 Class AVA: Vulnerability assessment .....	144
270	14.1 Introduction .....	144
271	14.2 Application notes .....	144
272	14.3 Vulnerability analysis (AVA_VAN) .....	145
273	14.3.1 Objectives .....	145
274	14.3.2 Component levelling .....	145
275	14.3.3 AVA_VAN.1 Vulnerability survey .....	145
276	14.3.4 AVA_VAN.2 Vulnerability analysis .....	146
277	14.3.5 AVA_VAN.3 Focused vulnerability analysis .....	147
278	14.3.6 AVA_VAN.4 Methodical vulnerability analysis .....	148
279	14.3.7 AVA_VAN.5 Advanced methodical vulnerability analysis .....	150
280	14.4 Composite vulnerability assessment (AVA_COMP) .....	151
281	14.4.1 Objectives .....	151
282	14.4.2 AVA_COMP.1 Composite product vulnerability assessment .....	151
283	15 Class ACO: Composition .....	152
284	15.1 Introduction .....	152
285	15.2 Composition rationale (ACO_COR) .....	155
286	15.2.1 Objectives .....	155
287	15.2.2 Component levelling .....	155
288	15.2.3 ACO_COR.1 Composition rationale .....	155
289	15.3 Development evidence (ACO_DEV) .....	155
290	15.3.1 Objectives .....	155
291	15.3.2 Component levelling .....	155
292	15.3.3 Application notes .....	156
293	15.3.4 ACO_DEV.1 Functional Description .....	156
294	15.3.5 ACO_DEV.2 Basic evidence of design .....	157
295	15.3.6 ACO_DEV.3 Detailed evidence of design .....	158
296	15.4 Reliance of dependent component (ACO_REL) .....	159
297	15.4.1 Objectives .....	159
298	15.4.2 Component levelling .....	159
299	15.4.3 Application notes .....	159
300	15.4.4 ACO_REL.1 Basic reliance information .....	160
301	15.4.5 ACO_REL.2 Reliance information .....	160
302	15.5 Composed TOE testing (ACO_CTT) .....	161
303	15.5.1 Objectives .....	161
304	15.5.2 Component levelling .....	161
305	15.5.3 Application notes .....	161
306	15.5.4 ACO_CTT.1 Interface testing .....	162

307	<b>15.5.5 ACO_CTT.2 Rigorous interface testing.....</b>	<b>163</b>
308	<b>15.6 Composition vulnerability analysis (ACO_VUL).....</b>	<b>164</b>
309	<b>15.6.1 Objectives .....</b>	<b>164</b>
310	<b>15.6.2 Component levelling.....</b>	<b>164</b>
311	<b>15.6.3 Application notes.....</b>	<b>164</b>
312	<b>15.6.4 ACO_VUL.1 Composition vulnerability review .....</b>	<b>165</b>
313	<b>15.6.5 ACO_VUL.2 Composition vulnerability analysis.....</b>	<b>166</b>
314	<b>15.6.6 ACO_VUL.3 Enhanced-Basic Composition vulnerability analysis.....</b>	<b>166</b>
315	<b>Annex A (informative) Development (ADV) .....</b>	<b>168</b>
316	<b>A.1 ADV_ARC: Supplementary material on security architectures .....</b>	<b>168</b>
317	<b>A.1.1 Security architecture properties .....</b>	<b>168</b>
318	<b>A.1.2 Security architecture descriptions.....</b>	<b>169</b>
319	<b>A.2 ADV_FSP: Supplementary material on functional specification .....</b>	<b>171</b>
320	<b>A.2.1 Non-TSF part of the TOE.....</b>	<b>172</b>
321	<b>A.2.2 Determining the TSFI .....</b>	<b>173</b>
322	<b>A.2.3 Example: A complex DBMS.....</b>	<b>175</b>
323	<b>A.2.4 Example Functional Specification.....</b>	<b>176</b>
324	<b>A.3 ADV_INT: Supplementary material on TSF internals.....</b>	<b>178</b>
325	<b>A.3.1 Structure of procedural software .....</b>	<b>179</b>
326	<b>A.3.2 Complexity of procedural software.....</b>	<b>181</b>
327	<b>A.4 ADV_TDS: Subsystems and Modules .....</b>	<b>181</b>
328	<b>A.4.1 Subsystems.....</b>	<b>181</b>
329	<b>A.4.2 Modules .....</b>	<b>182</b>
330	<b>A.4.3 Levelling Approach .....</b>	<b>185</b>
331	<b>A.4.4 Security relevance.....</b>	<b>187</b>
332	<b>A.5 Supplementary material on formal methods .....</b>	<b>188</b>
333	<b>Annex B (informative) Composition (ACO) .....</b>	<b>190</b>
334	<b>B.1 Necessity for composed TOE evaluations .....</b>	<b>190</b>
335	<b>B.2 Performing Security Target evaluation for a composed TOE .....</b>	<b>191</b>
336	<b>B.3 Interactions between composed IT entities.....</b>	<b>192</b>
337	<b>Annex C (informative) Cross reference of assurance component dependencies.....</b>	<b>198</b>
338		

## Foreword

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.

ISO/IEC 15408 consists of the following parts, under the general title *IT security techniques -- Evaluation criteria for IT security*:

— *Part 1: Introduction and general model*

— *Part 2: Security functional components*

— *Part 3: Security assurance components*

— *Part 4: Framework for the specification of evaluation methods and activities*

— *Part 5: Pre-defined packages of security requirements*

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.

371 **Legal Notice**

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

379	Australia/New Zealand:	The Defence Signals Directorate and the Government Communications
380		Security Bureau respectively;
381	Canada:	Communications Security Establishment;
382	France:	Agence nationale de la sécurité des systèmes d'information (ANSSI);
383	Germany:	Bundesamt für Sicherheit in der Informationstechnik;
384	Japan:	Information Technology Promotion Agency;
385	Netherlands:	Netherlands National Communications Security Agency;
386	Spain:	Ministerio de Administraciones Públicas and Centro Criptológico
387	Nacional;	
388	United Kingdom:	Communications-Electronic Security Group;
389	United States:	The National Security Agency and the National Institute of Standards
390	and	Technology.

## Introduction

Security assurance components, as defined in this part of ISO/IEC 15408, are the basis for the security assurance requirements expressed in a Protection Profile (PP) or a Security Target (ST).

These requirements establish a standard way of expressing the assurance requirements for TOEs. This part of ISO/IEC 15408 catalogues the set of assurance components, families and classes. This part of ISO/IEC 15408 also defines evaluation criteria for PPs and STs.

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

- a) Consumers, who use this part of ISO/IEC 15408 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 part of ISO/IEC 15408 when interpreting statements of assurance requirements and determining assurance approaches of TOEs.
- c) Evaluators, who use the assurance requirements defined in this part of ISO/IEC 15408 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 part of ISO/IEC 15408 defines the assurance requirements of ISO/IEC 15408. It includes the individual assurance components from which the assurance levels and packages contained in part 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. 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 part of ISO/IEC 15408

Clause 5 describes the paradigm used in the security assurance requirements of this part of ISO/IEC 15408.

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 part of ISO/IEC 15408 assurance classes.

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

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

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

445 0 provides a cross reference between PPs and the families and components of the APE class.

## 446 **5 Assurance paradigm**

### 447 **5.1 Introduction**

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

### 451 **5.2 ISO/IEC 15408 philosophy**

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

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

### 460 **5.3 Assurance approach**

#### 461 **5.3.1 Introduction**

462 ISO/IEC 15408 philosophy is to provide assurance based upon an evaluation (active investigation) of  
463 the IT product that is to be trusted. Evaluation has been the traditional means of providing assurance  
464 and is the basis for prior evaluation criteria documents. In aligning the existing approaches, ISO/IEC  
465 15408 adopts the same philosophy. ISO/IEC 15408 proposes measuring the validity of the  
466 documentation and of the resulting IT product by expert evaluators with increasing emphasis on  
467 scope, depth, and rigour.

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

#### 472 **5.3.2 Significance of vulnerabilities**

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



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

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

- 483 a) eliminated -- that is, active steps should be taken to expose, and remove or neutralise, all  
 484 exercisable vulnerabilities;
- 485 b) minimised -- that is, active steps should be taken to reduce, to an acceptable residual level, the  
 486 potential impact of any exercise of a vulnerability;
- 487 c) monitored -- that is, active steps should be taken to ensure that any attempt to exercise a residual  
 488 vulnerability will be detected so that steps can be taken to limit the damage.

### 489 **5.3.3 Cause of vulnerabilities**

490 Vulnerabilities can arise through failures in:

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

### 506 **5.3.4 ISO/IEC 15408 assurance**

507 Assurance is grounds for confidence that an IT product meets its security objectives. Assurance can be  
 508 derived from reference to sources such as unsubstantiated assertions, prior relevant experience, or  
 509 specific experience. However, ISO/IEC 15408 provides assurance through active investigation. Active  
 510 investigation is an evaluation of the IT product in order to determine its security properties.

### 511 **5.3.5 Assurance through evaluation**

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

- 514 a) analysis and checking of process(es) and procedure(s);
- 515 b) checking that process(es) and procedure(s) are being applied;

- 516 c) analysis of the correspondence between TOE design representations;
- 517 d) analysis of the TOE design representation against the requirements;
- 518 e) verification of proofs;
- 519 f) analysis of guidance documents;
- 520 g) analysis of functional tests developed and the results provided;
- 521 h) independent functional testing;
- 522 i) analysis for vulnerabilities (including flaw hypothesis);
- 523 j) penetration testing;
- 524 k) analysis of the delivery process.

## 525 **5.4 ISO/IEC 15408 evaluation assurance scale**

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

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

## 533 **6 Security assurance components**

### 534 **6.1 Security assurance classes, families and components structure**

535 The following subclauses describe the constructs used in representing the assurance classes, families,  
 536 and components.

537 Figure 1 illustrates the SARs defined in this part of ISO/IEC 15408. Note that the most abstract  
 538 collection of SARs is referred to as a class. Each class contains assurance families, which then contain  
 539 assurance components, which in turn contain assurance elements. Classes and families are used to  
 540 provide a taxonomy for classifying SARs, while components are used to specify SARs in a PP/ST.

#### 541 **6.1.1 Assurance class structure**

542 Figure 1 illustrates the assurance class structure.

##### 543 **6.1.1.1 Class name**

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

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

#### 549 6.1.1.2 Class introduction

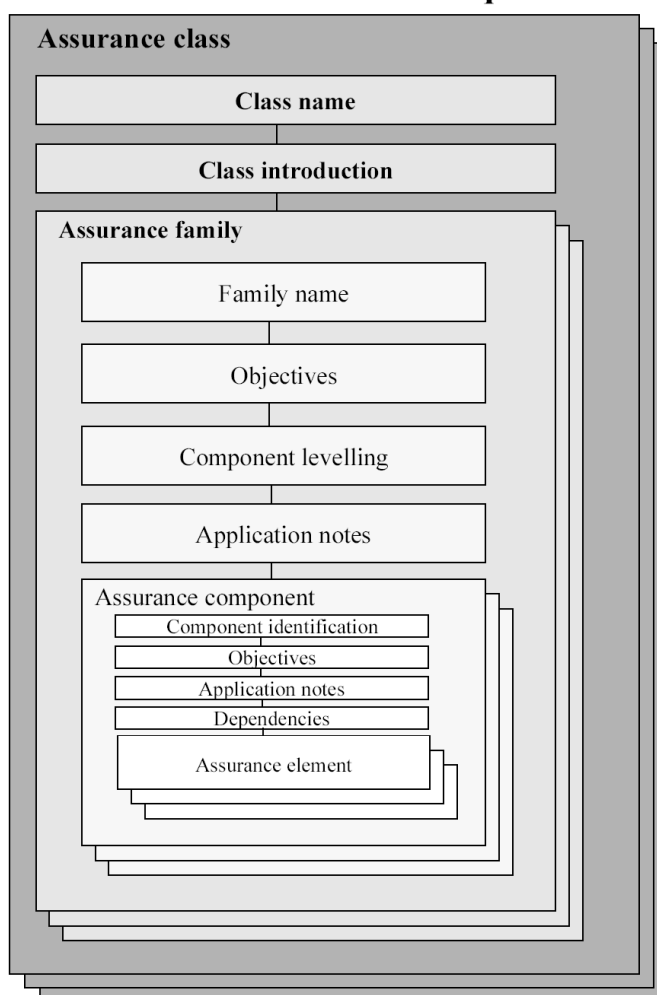
550 Each assurance class has an introductory subclause that describes the composition of the class and  
 551 contains supportive text covering the intent of the class.

#### 552 6.1.1.3 Assurance families

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

555 Figure 1 illustrates the assurance family structure.

### Common criteria assurance requirements



556  
 557 **Figure 1 — Assurance class/family/component/element hierarchy**

## 6.1.2 Assurance family structure

### 6.1.2.1 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.

### 6.1.2.2 Objectives

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

This subclause describes the objectives, particularly those related to ISO/IEC 15408 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.

### 6.1.2.3 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.

### 6.1.2.4 Application notes

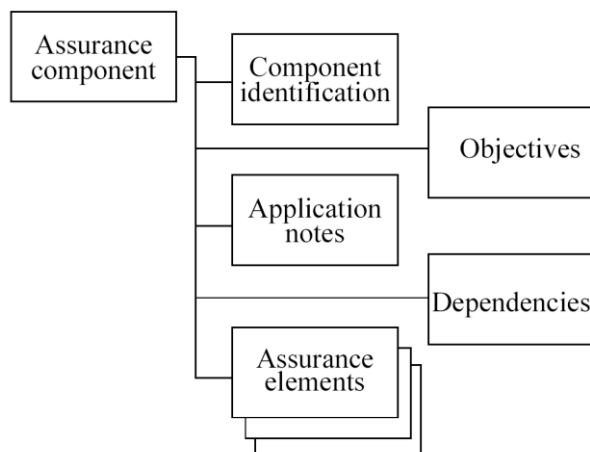
The application notes subclause of the assurance family, if present, contains additional information for the assurance family. This information should be of particular interest to users of the assurance family (e.g. PP and ST authors, designers of TOEs, evaluators). The presentation is informal and covers,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

### 6.1.2.5 Assurance components

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

## 6.1.3 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.

#### **6.1.3.1 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.

#### **6.1.3.2 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.

#### **6.1.3.3 Application notes**

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

#### **6.1.3.4 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.

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

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

#### **6.1.3.5 Assurance elements**

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

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

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

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

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

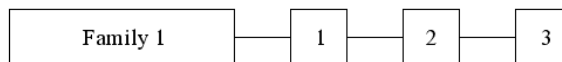
The developer action elements, content and presentation of evidence elements, and explicit evaluator action elements, identify the evaluator effort that shall be expended in verifying the security claims made in the ST of the TOE.

#### **6.1.4 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.

### 6.1.5 Component taxonomy

This part of ISO/IEC 15408 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 part of ISO/IEC 15408 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

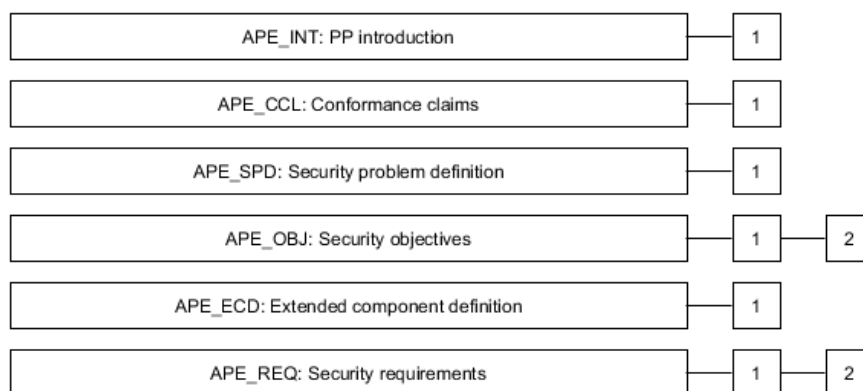
**Editor's Note (Mult-EAL approach):** The APE class must be extended to cover the conformity of a standard PP with one or more PPs/PP Configurations and potentially the addition of supplementary security problem, objectives and SFRs. The same kind of check as for PP-Modules and PP-Configurations apply. These updates will be provided once the proposed updates to the ACE class (in Section 3.2) have been agreed.

### 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.

This clause 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**

## 685 **7.2 PP introduction (APE\_INT)**

### 686 **7.2.1 Objectives**

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

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

### 690 **7.2.2 APE\_INT.1 PP introduction**

691 Dependencies: No dependencies.

#### 692 **7.2.2.1 Developer action elements**

##### 693 **7.2.2.1.1 APE\_INT.1.1D**

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

#### 695 **7.2.2.2 Content and presentation elements**

##### 696 **7.2.2.2.1 APE\_INT.1.1C**

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

##### 698 **7.2.2.2.2 APE\_INT.1.2C**

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

##### 700 **7.2.2.2.3 APE\_INT.1.3C**

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

##### 702 **7.2.2.2.4 APE\_INT.1.4C**

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

##### 704 **7.2.2.2.5 APE\_INT.1.5C**

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

#### 707 **7.2.2.3 Evaluator action elements**

##### 708 **7.2.2.3.1 APE\_INT.1.1E**

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

## 711 **7.3 Conformance claims (APE\_CCL)**

### 712 **7.3.1 Objectives**

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



715 **7.3.2 APE\_CCL.1 Conformance claims**

716 Dependencies: APE\_INT.1 PP introduction

717 APE\_ECD.1 Extended components definition

718 APE\_REQ.1 Stated security requirements

719 **7.3.2.1 Developer action elements**

720 **7.3.2.1.1 APE\_CCL.1.1D**

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

722 **7.3.2.1.2 APE\_CCL.1.2D**

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

724 **7.3.2.1.3 APE\_CCL.1.3D**

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

726 **7.3.2.2 Content and presentation elements**

727 **7.3.2.2.1 APE\_CCL.1.1C**

728 **The conformance claim shall contain an ISO/IEC 15408 conformance claim that identifies the**  
 729 **version of ISO/IEC 15408 to which the PP claims conformance.**

730 **7.3.2.2.2 APE\_CCL.1.2C**

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

733 **7.3.2.2.3 APE\_CCL.1.3C**

734 **ISO/IEC 15408 conformance claim shall describe the conformance of the PP to this part of**  
 735 **ISO/IEC 15408 as either this part of ISO/IEC 15408 conformant or this part of ISO/IEC 15408**  
 736 **extended.**

737 **7.3.2.2.4 APE\_CCL.1.4C**

738 **ISO/IEC 15408 conformance claim shall be consistent with the extended components**  
 739 **definition.**

740 **7.3.2.2.5 APE\_CCL.1.5C**

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

743 **7.3.2.2.6 APE\_CCL.1.6C**

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

746 **7.3.2.2.7 APE\_CCL.1.7C**

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

749 **7.3.2.2.8 APE\_CCL.1.8C**

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

753 **7.3.2.2.9 APE\_CCL.1.9C**

754 **The conformance claim rationale shall demonstrate that the statement of security objectives is**  
 755 **consistent with the statement of security objectives in the PPs for which conformance is being**  
 756 **claimed.**

757 **7.3.2.2.10 APE\_CCL.1.10C**

758 **The conformance claim rationale shall demonstrate that the statement of security**  
 759 **requirements is consistent with the statement of security requirements in the PPs for which**  
 760 **conformance is being claimed.**

761 **7.3.2.2.11 APE\_CCL.1.11C**

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

764 **7.3.2.2.12 APE\_CCL.1.12C**

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

767 **7.3.2.2.13 APE\_CCL.1.13C**

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

770 **7.3.2.3 Evaluator action elements**

771 **7.3.2.3.1 APE\_CCL.1.1E**

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

774 **7.4 Security problem definition (APE\_SPD)**

775 **7.4.1 Objectives**

776 **This part of the PP defines the security problem to be addressed by the TOE and the operational**  
 777 **environment of the TOE.**

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

780 **7.4.2 APE\_SPD.1 Security problem definition**

781 Dependencies: No dependencies.

782 **7.4.2.1 Developer action elements**

783 **7.4.2.1.1 APE\_SPD.1.1D**

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

785 **7.4.2.2 Content and presentation elements**

786 **7.4.2.2.1 APE\_SPD.1.1C**

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

788 **7.4.2.2.2 APE\_SPD.1.2C**

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

790 **7.4.2.2.3 APE\_SPD.1.3C**

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

792 **7.4.2.2.4 APE\_SPD.1.4C**

793 **The security problem definition shall describe the assumptions about the operational**  
 794 **environment of the TOE.**

795 **7.4.2.3 Evaluator action elements**

796 **7.4.2.3.1 APE\_SPD.1.1E**

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

799 **7.5 Security objectives (APE\_OBJ)**

800 **7.5.1 Objectives**

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

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

806 **7.5.2 Component levelling**

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

809 **7.5.3 APE\_OBJ.1 Security objectives for the operational environment**

810 Dependencies: No dependencies.

811 **7.5.3.1 Developer action elements**

812 **7.5.3.1.1 APE\_OBJ.1.1D**

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

814 **7.5.3.2 Content and presentation elements**

815 **7.5.3.2.1 APE\_OBJ.1.1C**

816 **The statement of security objectives shall describe the security objectives for the operational**  
 817 **environment.**

818

819 **7.5.3.3 Evaluator action elements**

820 **7.5.3.3.1 APE\_OBJ.1.1E**

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

823 **7.5.4 APE\_OBJ.2 Security objectives**

824 Dependencies: APE\_SPD.1 Security problem definition

825 **7.5.4.1 Developer action elements**

826 **7.5.4.1.1 APE\_OBJ.2.1D**

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

828 **7.5.4.1.2 APE\_OBJ.2.2D**

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

830 **7.5.4.2 Content and presentation elements**

831 **7.5.4.2.1 APE\_OBJ.2.1C**

832 **The statement of security objectives shall describe the security objectives for the TOE and the**  
 833 **security objectives for the operational environment.**

834 **7.5.4.2.2 APE\_OBJ.2.2C**

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

837 **7.5.4.2.3 APE\_OBJ.2.3C**

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

841 **7.5.4.2.4 APE\_OBJ.2.4C**

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

844 **7.5.4.2.5 APE\_OBJ.2.5C**

845 **The security objectives rationale shall demonstrate that the security objectives enforce all**  
 846 **OSPs.**

847 **7.5.4.2.6 APE\_OBJ.2.6C**

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

850 **7.5.4.3 Evaluator action elements**

851 **7.5.4.3.1 APE\_OBJ.2.1E**

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

854 **7.6 Extended components definition (APE\_ECD)**

855 **7.6.1 Objectives**

856 Extended security requirements are requirements that are not based on components from ISO/IEC  
 857 15408-2 or this part of ISO/IEC 15408, but are based on extended components: components defined  
 858 by the PP author.

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

862 **7.6.2 APE\_ECD.1 Extended components definition**

863 Dependencies: No dependencies.

864 **7.6.2.1 Developer action elements**

865 **7.6.2.1.1 APE\_ECD.1.1D**

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

867 **7.6.2.1.2 APE\_ECD.1.2D**

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

869 **7.6.2.2 Content and presentation elements**

870 **7.6.2.2.1 APE\_ECD.1.1C**

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

872 **7.6.2.2.2 APE\_ECD.1.2C**

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

875 **7.6.2.2.3 APE\_ECD.1.3C**

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

878 **7.6.2.2.4 APE\_ECD.1.4C**

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

881 **7.6.2.2.5 APE\_ECD.1.5C**

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

884 **7.6.2.3 Evaluator action elements**

885 **7.6.2.3.1 APE\_ECD.1.1E**

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

888 **7.6.2.3.2 APE\_ECD.1.2E**

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

891 **7.7 Security requirements (APE\_REQ)**

892 **7.7.1 Objectives**

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

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

898 **7.7.2 Component levelling**

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

901 **7.7.3 APE\_REQ.1 Stated security requirements**

902 **Dependencies: APE\_ECD.1 Extended components definition**

903 **7.7.3.1 Developer action elements**

904 **7.7.3.1.1 APE\_REQ.1.1D**

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

906 **7.7.3.1.2 APE\_REQ.1.2D**

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

908 **7.7.3.2 Content and presentation elements**

909 **7.7.3.2.1 APE\_REQ.1.1C**

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

911 **7.7.3.2.2 APE\_REQ.1.2C**

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

914 **7.7.3.2.3 APE\_REQ.1.3C**

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

919 **7.7.3.2.4 APE\_REQ.1.4C**

920 **The statement of security requirements shall identify all operations on the security**  
 921 **requirements.**

922 **7.7.3.2.5 APE\_REQ.1.5C**

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

924 **7.7.3.2.6 APE\_REQ.1.6C**

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

927 **7.7.3.2.7 APE\_REQ.1.7C**

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

930 **7.7.3.2.8 APE\_REQ.1.8C**

931 **The security requirements rationale shall trace each security objective for the operational**  
 932 **environment back to threats countered by that security objective, OSPs enforced by that**  
 933 **security objective, and assumptions upheld by that security objective.**

934 **7.7.3.2.9 APE\_REQ.1.9C**

935 **The security requirements rationale shall demonstrate that the SFRs counter all threats for the**  
 936 **TOE.**

937 **7.7.3.2.10 APE\_REQ.1.10C**

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

939 **7.7.3.2.11 APE\_REQ.1.11C**

940 **The security requirements rationale shall demonstrate that the security objectives for the**  
 941 **operational environment uphold all assumptions.**

942 **7.7.3.2.12 APE\_REQ.1.12C**

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

944 **7.7.3.3 Evaluator action elements**

945 **7.7.3.3.1 APE\_REQ.1.1E**

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

948 **7.7.4 APE\_REQ.2 Derived security requirements**

949 Dependencies: APE\_OBJ.2 Security objectives

950 APE\_ECD.1 Extended components definition

951 **7.7.4.1 Developer action elements**

952 **7.7.4.1.1 APE\_REQ.2.1D**

953 The developer shall provide a statement of security requirements.

954 **7.7.4.1.2 APE\_REQ.2.2D**

955 The developer shall provide a security requirements rationale.

956 **7.7.4.2 Content and presentation elements**

957 **7.7.4.2.1 APE\_REQ.2.1C**

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

959 **7.7.4.2.2 APE\_REQ.2.2C**

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

962 **7.7.4.2.3 APE\_REQ.2.3C**

963 **7.7.4.2.4** The statement of security requirements shall include a natural language description, part  
 964 of which describes how the SFRs combine together to provide security functionality in terms of the



- 965 architecture that is observable to Administrators and other users, or in terms of internal features or  
 966 properties.**APE\_REQ.2.4C**
- 967 The statement of security requirements shall identify all operations on the security requirements.
- 968 **7.7.4.2.5 APE\_REQ.2.5C**
- 969 All operations shall be performed correctly.
- 970 **7.7.4.2.6 APE\_REQ.2.6C**
- 971 Each dependency of the security requirements shall either be satisfied, or the security requirements  
 972 rationale shall justify the dependency not being satisfied.
- 973 **7.7.4.2.7 APE\_REQ.2.7C**
- 974 The security requirements rationale shall trace each SFR back to the security objectives threats  
 975 countered by that SFR and OSPs enforced by that SFR.
- 976 **7.7.4.2.8 APE\_REQ.2.8C**
- 977 The security requirements rationale shall trace each security objective for the operational  
 978 environment back to threats countered by that security objective, OSPs enforced by that security  
 979 objective, and assumptions upheld by that security objective.
- 980 **7.7.4.2.9 APE\_REQ.2.9C**
- 981 The security requirements rationale shall demonstrate that the SFRs counter all threats for the TOE.
- 982 **7.7.4.2.10 APE\_REQ.2.10C**
- 983 The security requirements rationale shall demonstrate that the SFRs enforce all OSPs.
- 984 **7.7.4.2.11 APE\_REQ.2.11C**
- 985 The security requirements rationale shall demonstrate that the security objectives for the operational  
 986 environment uphold all assumptions.
- 987 **7.7.4.2.12 APE\_REQ.2.12C**
- 988 **The security requirements rationale shall trace each SFR back to the security objectives for the**  
 989 **TOE.**
- 990 **7.7.4.2.13 APE\_REQ.2.13C**
- 991 **The security requirements rationale shall demonstrate that the SFRs meet all security**  
 992 **objectives for the TOE.**
- 993 **7.7.4.2.14 APE\_REQ.2.14C**
- 994 **The security requirements rationale shall explain why the SARs were chosen.**
- 995 **7.7.4.2.15 APE\_REQ.2.15C**
- 996 The statement of security requirements shall be internally consistent.

997 **7.7.4.3 Evaluator action elements**

998 **7.7.4.3.1 APE\_REQ.2.1E**

999 The evaluator shall confirm that the information provided meets all requirements for content and  
1000 presentation of evidence.

1001 **8 Class ACE: Protection Profile Configuration evaluation**

1002 **8.1 Introduction**

1003 Evaluating a PP-Configuration is required to demonstrate that the PP-Configuration is sound and  
1004 consistent. These properties are necessary for the PP-Configuration to be suitable for use as the basis  
1005 for writing an ST or another PP or PP-Configuration.

1006 The class ACE is defined for the evaluation of a PP-Configuration composed of one or more PPs and  
1007 one or more PP-Modules.

1008 This Clause should be used in conjunction with Annexes B and D in ISO/IEC 1540-1, as these Annexes  
1009 clarify the concepts here and provide many examples.

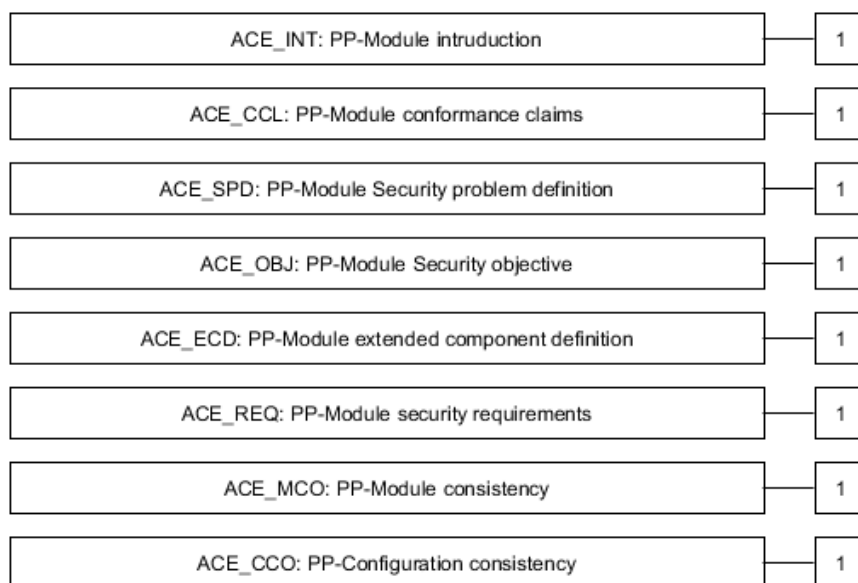
1010 This standard does not define Direct Rationale PP-Configuration evaluation package. There is only one  
1011 assurance package for PP-Configuration evaluation, equivalent to Standard PP evaluation package.

1012 The evaluator shall decide the order in which the unevaluated components of a PP-Configuration (PPs  
1013 and PP-Modules) are evaluated. Class APE addresses the evaluation of PPs. The present class ACE  
1014 defines the requirements for

- 1015 • Evaluating PP-Modules under the assumption that its basis is internally consistent.
- 1016 • Evaluating the consistency of the combination of all the PPs and PP-Modules that transitively  
1017 belong to the PP-Configurations.

1018 Note: Two PP-Modules may define each other in their basis, which means that a PP-Configuration that  
1019 contains one of them also contains the other.

1020 The ACE class is based on APE.



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

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

### **8.2.1 Objectives**

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

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

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

Dependencies: No dependencies.

#### **8.2.2.1 Application notes**

All content and presentation elements of APE\_INT.1 hold with PP-Module instead of PP.

#### **8.2.2.2 Developer action elements**

##### **8.2.2.2.1 ACE\_INT.1.1D**

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

#### **8.2.2.3 Content and presentation elements**

##### **8.2.2.3.1 ACE\_INT.1.1C**

**8.2.2.3.2** •The PP-Module introduction shall meet the content and presentation requirements for PP introduction as defined in APE\_INT.1.1C to APE\_INT.1.5C.**ACE\_INT.1.2C**

**The PP-Module introduction shall uniquely identify the base PPs and PP-Modules it depends on.**

1042 **8.2.2.3.3 ACE\_INT.1.3C**

1043 **The PP-Module introduction shall describe the dependency structure of the base PPs and PP-**  
1044 **Modules.**

1045 **8.2.2.3.4 ACE\_INT.1.4C**

1046 **The TOE overview shall describe the differences of the TOE with regard to the TOEs defined in**  
1047 **the base PPs and PP-Modules.**

1048 **8.2.2.4 Evaluator action elements**

1049 **8.2.2.4.1 ACE\_INT.1.1E**

1050 **The evaluator shall confirm that the information provided meets all requirements for content**  
1051 **and presentation of evidence.**

1052 **8.3 PP-Module conformance claims (ACE\_CCL)**

1053 **8.3.1 Objectives**

1054 The objective of this family is to determine the validity of the conformance claim and conformance  
1055 statement. Unlike standard Protection Profiles, a PP-Module cannot claim conformance to another PP  
1056 or PP-Module.

1057 **8.3.2 ACE\_CCL.1 PP-Module conformance claims**

1058 Dependencies: ACE\_INT.1 PP-Module introduction

1059 ACE\_ECD.1 PP-Module extended components definition

1060 ACE\_REQ.1 PP-Module security requirements

1061 **8.3.2.1 Application notes**

1062 **8.3.2.2 All content and presentation elements of APE\_CCL.1 hold, except the requirements**  
1063 **about conformance to a PP.Developer action elements**

1064 **8.3.2.2.1 ACE\_CCL.1.1D**

1065 **The developer shall provide a conformance claim.**

1066 **8.3.2.2.2 ACE\_CCL.1.2D**

1067 **The developer shall provide a conformance statement.**

1068 **8.3.2.3 Content and presentation elements**

1069 **8.3.2.3.1 ACE\_CCL.1.1C**

1070 **The PP-Module conformance claim shall meet the content and presentation requirements for**  
1071 **PP conformance claim as defined in APE\_CCL.1.1C to APE\_INT.1.4C and APE\_CCL.1.6C**

1072 **8.3.2.3.2 ACE\_CCL.1.2C**

1073 **The PP-Module conformance statement shall meet the content requirements for PP**  
 1074 **conformance statement as defined in APE\_CCL.1.10C to APE\_INT.1.13C.**

1075 **8.3.2.3.3 ACE\_CCL.1.3C**

1076 **The conformance claim shall identify all security requirement packages to which the PP-**  
 1077 **Module claims conformance.**

1078 **8.3.2.3.4 ACE\_CCL.1.4C**

1079 **If the PP-Module is one of demonstrable or strict conformance type, then the conformance**  
 1080 **claim shall define the PP-Module AL's name and content, i.e. the set of SARs that applies to the**  
 1081 **TOE.**

1082

1083 **Editor's note to himself: If the multi EAL-approach and the related version of ACE\_CCL.1.1C remains in the**  
 1084 **document, remove ACE\_CCL.1.6C as proposed in v0.3 of the concept.**

1085 **8.3.2.3.5 ACE\_CCL.1.6C**

1086 **The conformance claim shall describe any conformance of the PP to a package as either**  
 1087 **package-conformant or package-augmented.**

1088 **8.3.2.4 Evaluator action elements**

1089 **8.3.2.4.1 ACE\_CCL.1.1E**

1090 **The evaluator shall confirm that the information provided meets all requirements for content**  
 1091 **and presentation of evidence.**

1092 **8.4 PP-Module Security problem definition (ACE\_SPD)**

1093 **8.4.1 ACE\_SPD.1 PP-Module Security problem definition**

1094 Dependencies: No dependencies.

1095 **8.4.2 Application notes**

1096 All content and presentation elements of APE\_SPD.1 hold.

1097 **8.4.2.1 Developer action elements**

1098 **8.4.2.1.1 ACE\_SPD.1.1D**

1099 **The developer shall provide a security problem definition.**

1100 **8.4.2.2 Content and presentation elements**

1101 **8.4.2.2.1 ACE\_SPD.1.1C**

1102 **The PP-Module security problem definition shall meet the content and presentation**  
 1103 **requirements for PP security problem definition as defined in APE\_SPD.1.1C to APE\_SPD.1.4C.**

- 1104 **8.4.2.3 Evaluator action elements**
- 1105 **8.4.2.3.1 ACE\_SPD.1.1E**
- 1106 **The evaluator shall confirm that the information provided meets all requirements for content**  
 1107 **and presentation of evidence.**
- 1108 **8.5 PP-Module Security objectives (ACE\_OBJ)**
- 1109 **8.5.1 ACE\_OBJ.1 Direct Rationale PP-Module Security objectives**
- 1110 Dependencies: No dependencies.
- 1111 **8.5.2 Application notes**
- 1112 If the PP-Module uses the Direct Rationale approach (as determined in ACE\_CCO.1-2) then all the  
 1113 content and presentation elements of APE\_OBJ.1.1C hold.
- 1114 **8.5.2.1 Developer action elements**
- 1115 **8.5.2.1.1 ACE\_OBJ.1.1D**
- 1116 **The developer shall provide a statement of security objectives for the environment.**
- 1117 **8.5.2.2 Content and presentation elements**
- 1118 **8.5.2.2.1 ACE\_OBJ.1.1C**
- 1119 **The Direct Rationale PP-Module security objectives shall meet the content and presentation**  
 1120 **requirements for Direct Rationale PP security objectives as defined in APE\_OBJ.1.1C.**
- 1121 Application Note: Recall that in the Direct Rationale approach the traceability of the objectives to the  
 1122 SPD is not applicable.
- 1123 **8.5.2.3 Evaluator action elements**
- 1124 **8.5.2.3.1 ACE\_OBJ.1.1E**
- 1125 **The evaluator shall confirm that the information provided meets all requirements for content**  
 1126 **and presentation of evidence.**
- 1127 **8.5.3 ACE\_OBJ.2 PP-Module Security objectives**
- 1128 Dependencies: No dependencies.
- 1129 **8.5.4 Application notes**
- 1130 If the PP-Module does not use the Direct Rationale approach (as determined in ACE\_CCO.1-2) then all  
 1131 content and presentation elements of APE\_OBJ.2 hold.
- 1132 **8.5.4.1 Developer action elements**
- 1133 **8.5.4.1.1 ACE\_OBJ.2.1D**
- 1134 **The developer shall provide a statement of security objectives.**

1135 **8.5.4.1.2 ACE\_OBJ.2.2D**

1136 **The developer shall provide a security objectives rationale.**

1137 **8.5.4.2 Content and presentation elements**

1138 **8.5.4.2.1 ACE\_OBJ.2.1C**

1139 **The PP-Module security objectives and rationale shall meet the content and presentation**  
 1140 **requirements for PP security objectives and rationale as defined in APE\_OBJ.2.1C to**  
 1141 **APE\_OBJ.2.6C.**

1142 **8.5.4.3 Evaluator action elements**

1143 **8.5.4.3.1 ACE\_OBJ.2.1E**

1144 The evaluator shall confirm that the information provided meets all requirements for content and  
 1145 presentation of evidence.

1146 **8.6 PP-Module extended components definition (ACE\_ECD)**

1147 **8.6.1 Objectives**

1148 Extended security functional requirements are requirements that are not based on components from  
 1149 **ISO/IEC 15408-2**, but are based on extended components: components defined by the PP-Module  
 1150 author.

1151 Evaluation of the definition of extended functional components is necessary to determine that they are  
 1152 clear and unambiguous, and that they are necessary, i.e. they may not be clearly expressed using  
 1153 existing **ISO/IEC 15408-2** components.

1154 **8.6.2 ACE\_ECD.1 PP-Module extended components definition**

1155 Dependencies: No dependencies.

1156 **8.6.3 Application notes**

1157 All the actions, content and presentation elements of APE\_ECD.2 hold.

1158 **8.6.3.1 Developer action elements**

1159 **8.6.3.1.1 ACE\_ECD.1.1D**

1160 **The developer shall provide a statement of security requirements.**

1161 **8.6.3.1.2 ACE\_ECD.1.2D**

1162 **The developer shall provide an extended components definition.**

1163 **8.6.3.2 Content and presentation elements**

1164 **8.6.3.2.1 ACE\_ECD.1.1C**

1165 **The statement of security requirements and the extended components definition shall meet the**  
 1166 **content and presentation requirements for PP statement of security requirements and the**  
 1167 **extended components definition as defined in APE\_ECD.1.1C to APE\_ECD.1.5C.**

Editor's Note: This allows removing old ACE\_ECD.1.1C to ACE\_ECD.1.5C, which apply only to security functional requirements. In the multi-assurance framework, the PP-Modules can define extended SARs as well

### 8.6.3.3 Evaluator action elements

#### 8.6.3.3.1 ACE\_ECD.1.1E

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

#### 8.6.3.3.2 ACE\_ECD.1.2E

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

## 8.7 PP-Module security requirements (ACE\_REQ)

### 8.7.1 Objectives

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.

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

### 8.7.2 Component levelling

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

### 8.7.3 ACE\_REQ.1 PP-Module stated security requirements

Dependencies: APE\_ECD.1 Extended components definition

### 8.7.4 Application notes

All the actions, content and presentation elements of APE\_REQ.1 hold.

#### 8.7.4.1 Developer action elements

##### 8.7.4.1.1 ACE\_REQ.1.1D

The developer shall provide a statement of security requirements.

##### 8.7.4.1.2 ACE\_REQ.1.2D

The developer shall provide a security requirements rationale.

#### 8.7.4.2 Content and presentation elements

Editor's note: the first ACE\_REQ.1.1C comes from the multi-EAL-Approach and aims to replace the old ACE\_REQ.1.1C to ACE\_REQ.1.12C.



1200 As the “old” ACE\_REQ.1.xC elements are also new to this version, the editor decides to keep both versions  
 1201 in the document but to mark the older ones yellow.

#### 1202 8.7.4.2.1 ACE\_REQ.1.1C

1203 The statement of security requirements and the rationale shall meet the content and  
 1204 presentation requirements for PP statement of security requirements and rationale as defined  
 1205 in APE\_REQ.1.1C to APE\_REQ.1.12C.

#### 1206 8.7.4.2.2 ACE\_REQ.1.1C

1207 The statement of security requirements shall describe the SFRs that hold on the TOE.

#### 1208 8.7.4.2.3 ACE\_REQ.1.2C

1209 All subjects, objects, operations, security attributes, external entities and other terms that are  
 1210 used in the SFRs shall be defined.

#### 1211 8.7.4.2.4 ACE\_REQ.1.3C

1212 The statement of security requirements shall include a natural language description, part of  
 1213 which describes how the SFRs combine together to provide security functionality in terms of  
 1214 the architecture that is visible to Administrators and other users.

#### 1215 8.7.4.2.5 ACE\_REQ.1.4C

1216 The statement of security requirements shall identify all operations on the security  
 1217 requirements.

#### 1218 8.7.4.2.6 ACE\_REQ.1.5C

1219 All operations shall be performed correctly.

#### 1220 8.7.4.2.7 ACE\_REQ.1.6C

1221 Each dependency of the security requirements shall either be satisfied, or the security  
 1222 requirements rationale shall justify the dependency not being satisfied.

#### 1223 8.7.4.2.8 ACE\_REQ.1.7C

1224 The security requirements rationale shall trace each SFR back to the security objectives threats  
 1225 countered by that SFR and OSPs enforced by that SFR.

#### 1226 8.7.4.2.9 ACE\_REQ.1.8C

1227 The security requirements rationale shall trace each security objective for the operational  
 1228 environment back to threats countered by that security objective, OSPs enforced by that  
 1229 security objective, and assumptions upheld by that security objective.

#### 1230 8.7.4.2.10 ACE\_REQ.1.9C

1231 The security requirements rationale shall demonstrate that the SFRs counter all threats for the  
 1232 TOE.

1233 **8.7.4.2.11 ACE\_REQ.1.10C**1234 **The security requirements rationale shall demonstrate that the SFRs enforce all OSPs.**1235 **8.7.4.2.12 ACE\_REQ.1.11C**1236 **The security requirements rationale shall demonstrate that the security objectives for the**  
1237 **operational environment uphold all assumptions.**1238 **8.7.4.2.13 ACE\_REQ.1.12C**1239 **The statement of security requirements shall be internally consistent.**1240 **8.7.4.3 Evaluator action elements**1241 **8.7.4.3.1 ACE\_REQ.1.1E**1242 **The evaluator shall confirm that the information provided meets all requirements for content**  
1243 **and presentation of evidence.**1244 **Editor's note: The multi-EAL-approach proposes to remove the term "functional" in ACE\_REQ.2 as well as**  
1245 **some other words. If these terms are new in this version and the deletion cannot be tracked, the term is**  
1246 **marked yellow.**1247 **8.7.5 ACE\_REQ.2 PP-Module security functional requirements**

1248 Dependencies: ACE\_ECD.1 PP-Module extended components definition

1249 ACE\_OBJ.1 PP-Module Security objectives

1250 **8.7.6 Application note**

1251 All the actions, content and presentation elements of APE\_REQ.2 hold.

1252 **8.7.6.1 Developer action elements**1253 **8.7.6.1.1 ACE\_REQ.2.1D**

1254 The developer shall provide a statement of security requirements.

1255 **8.7.6.1.2 ACE\_REQ.2.2D**1256 The developer shall provide a security **functional** requirement rationale.1257 **8.7.6.2 Content and presentation elements**1258 **Editor's note: the first ACE\_REQ.2.1C comes from the multi-EAL-Approach and aims to replace the old**  
1259 **ACE\_REQ.1.1C to ACE\_REQ.1.12C.**1260 **As the "old" ACE\_REQ.2.xC elements are also new to this version, the editor decides to keep both versions**  
1261 **in the document but to mark the older ones yellow.**

1262 **8.7.6.2.1 ACE\_REQ.2.1C**

1263 The statement of security requirements and the rationale shall meet the content and presentation  
1264 requirements for PP statement of security requirements and rationale as defined in **APE\_REQ.2.1C** to  
1265 **APE\_REQ.1.15C**.

1266 **8.7.6.2.2 ACE\_REQ.2.1C**

1267 The statement of security **functional** requirements shall describe the SFRs that hold on the TOE.

1268 **8.7.6.2.3 ACE\_REQ.2.2C**

1269 All subjects, objects, operations, security attributes, external entities and other terms that are used in  
1270 the SFRs shall be defined.

1271 **8.7.6.2.4 ACE\_REQ.2.3C**

1272 The statement of security requirements shall include a natural language description, part of which  
1273 describes how the SFRs combine together to provide security functionality in terms of the architecture  
1274 that is visible to Administrators and other users.

1275 **8.7.6.2.5 ACE\_REQ.2.4C**

1276 The statement of security **functional** requirements shall identify all operations on the security  
1277 functional requirements.

1278 **8.7.6.2.6 ACE\_REQ.2.5C**

1279 All operations shall be performed correctly.

1280 **8.7.6.2.7 ACE\_REQ.2.6C**

1281 Each dependency of the security **functional** requirements shall either be satisfied, or the security  
1282 functional requirements rationale shall justify the dependency not being satisfied.

1283 **8.7.6.2.8 ACE\_REQ.2.7C**

1284 The security **functional** requirements rationale shall trace each SFR back to the security objectives for  
1285 the TOE.

1286 **8.7.6.2.9 ACE\_REQ.2.8C**

1287 The security **functional** requirements rationale shall trace each security objective for the operational  
1288 environment back to threats countered by that security objective, OSPs enforced by that security  
1289 objective, and assumptions upheld by that security objective.

1290 **8.7.6.2.10 ACE\_REQ.2.9C**

1291 The security **functional** requirements rationale shall demonstrate that the SFRs meet all security  
1292 objectives for the TOE.

1293 **8.7.6.2.11 ACE\_REQ.2.10C**

1294 The security **functional** requirements rationale shall demonstrate that the SFRs enforce all OSPs.

1295 **8.7.6.2.12 ACE\_REQ.2.11C**

1296 The security **functional** requirements rationale shall demonstrate that the security objectives for the  
 1297 operational environment uphold all assumptions.

1298 **8.7.6.2.13 ACE\_REQ.2.12C**

1299 The statement of security **functional** requirements shall be internally consistent.

1300 **8.7.6.3 Evaluator action elements**1301 **8.7.6.3.1 ACE\_REQ.2.1E**

1302 The evaluator shall confirm that the information provided meets all requirements for content  
 1303 and presentation of evidence.

1304 **8.8 PP-Module consistency (ACE\_MCO)**1305 **8.8.1 Objectives**

1306 The objective of this family is to determine the consistency of the PP-Module.

1307 **8.8.2 ACE\_MCO.1 PP-Module consistency**

1308 Dependencies: ACE\_INT.1 PP-Module introduction

1309 ACE\_SPD.1 PP-Module Security problem definition

1310 ACE\_OBJ.1 PP-Module Security objectives

1311 ACE\_REQ.1 PP-Module security requirements

1312 **8.8.2.1 Developer action elements**1313 **8.8.2.1.1 ACE\_MCO.1.1D**

1314 The developer shall provide a consistency rationale of the PP-Module for each of the alternative  
 1315 sets of base PPs and PP-Modules identified in the PP-Module introduction.

1316 **8.8.2.2 Content and presentation elements**1317 **8.8.2.2.1 ACE\_MCO.1.1C**

1318 The consistency rationale shall demonstrate that the TOE type of the PP-Module and its base  
 1319 PPs and PP-Modules are consistent.

1320 **8.8.2.2.1 ACE\_MCO.1.2C**

1321 The consistency rationale shall identify the assets of the PP-Module that also belong to one or  
 1322 more base PP or PP-Module and amongst them those for which the PP-Module and the base PP  
 1323 and PP-Modules define different security problems.

1324 Editor's Note: this is also meaningful for APE and ASE when the ST claims conformance to more than one PP  
 1325 or when the ST adds elements to the PPs it conforms to: The change has not been proposed yet in ASE/APE,  
 1326 but if experts agree, we suggest cascading this change in the next CD.

1327 **8.8.2.2.2 ACE\_MCO.1.3C**

1328 **The consistency rationale shall demonstrate that the statements of the security problem**  
 1329 **definition of the PP-Module and its base PPs and PP-Modules are consistent.**

1330 **8.8.2.2.3 ACE\_MCO.1.4C**

1331 **The consistency rationale shall demonstrate that the statements of the security objectives of**  
 1332 **the PP-Module and its base PPs and PP-Modules are consistent.**

1333 **8.8.2.2.4 ACE\_MCO.1.5C**

1334 **The consistency rationale shall demonstrate that the statements of the security functional**  
 1335 **requirements of the PP-Module and its base PPs and PP-Modules are consistent.**

1336 **8.8.2.2.5 ACE\_MCO.1.6C**

1337 **The consistency rationale shall demonstrate that the statement of the security assurance**  
 1338 **requirements of the PP-Module is consistent with the statements of the security assurance**  
 1339 **requirements in the base PPs and PP-Modules identified in the PP-Module introduction.**

1340 **The consistency rationale shall demonstrate that the statements of the security assurance**  
 1341 **requirements of the PP-Module and its base PPs and PP-Modules are consistent.**

1342 **8.8.2.3 Evaluator action elements**

1343 **8.8.2.3.1 ACE\_MCO.1.1E**

1344 **The evaluator shall confirm that the information provided meets all requirements for content**  
 1345 **and presentation of evidence. If the PP-Module specifies alternate sets of base PPs and PP-**  
 1346 **Modules, the evaluator shall perform this action for each consistency rationale.**

1347 **8.9 PP-Configuration consistency (ACE\_CCO)**

1348 **8.9.1 Objectives**

1349 The objective of this family is to determine the well-formedness and the consistency of the PP-  
 1350 Configuration.

1351 **8.9.2 ACE\_CCO.1 PP-Configuration consistency**

1352 Dependencies: ACE\_INT.1 PP-Module introduction

1353 ACE\_CCL.1 PP-Module conformance claims

1354 ACE\_SPD.1 PP-Module security problem definition

1355 ACE\_OBJ.1 PP-Module security objectives

1356 ACE\_ECD.1 PP-Module extended component definition

1357 ACE\_REQ.1 PP-Module security requirements

1358 ACE\_MCO.1 PP-Module consistency

1359 **8.9.2.1 Developer action elements**

1360 **8.9.2.1.1 ACE\_CCO.1.1D**

1361 **The developer shall provide the reference of the PP-Configuration.**

1362 **8.9.2.1.2 ACE\_CCO.1.2D**

1363 **The developer shall provide a components list.**

1364 **8.9.2.1.3 ACE\_CCO.1.3D**

1365 **The developer shall provide a a conformance claim.**

1366 **8.9.2.1.4 ACE\_CCO.1.4D**

- 1367 • **The developer shall provide a conformance claim rationale.**

1368 **8.9.2.1.5 ACE\_CCO.1.5D**

1369 **The developer shall provide a conformance statement.**

1370 **8.9.2.1.6 ACE\_CCO.1.6D**

1371 **The developer shall provide a consistency rationale.**

1372 **8.9.2.2 Content and presentation elements**

1373 **8.9.2.2.1 ACE\_CCO.1.1C**

1374 **The PP-Configuration reference shall uniquely identify the PP-Configuration.**

1375 **8.9.2.2.2 ACE\_CCO.1.2C**

1376 **The components list shall uniquely identify the PPs and PP-Modules that compose the PP-**  
 1377 **Configuration.**

1378 **8.9.2.2.3 ACE\_CCO.1.3C**

1379 **The conformance claim shall contain a CC conformance claim that identifies the version(s) of**  
 1380 **the CC to which the PP-Configuration and its underlying Protection Profile and PP-Module**  
 1381 **claim conformance.**

1382 **8.9.2.2.4 ACE\_CCO.1.4C**

1383 **If the PP-Configuration is one of demonstrable, strict or multiple conformance type, then the**  
 1384 **conformance claim shall define the PP-Configuration AL's name and content:**

- 1385 • **The set of PP ALs and PP-Modules ALs inherited from the PPs and PP-Modules that**  
 1386 **transitively belong to the PP-Configuration's components list, possibly augmented.**
- 1387 • **The global AL, i.e. the set of SARs that applies to the entire TOE.**

1388 **8.9.2.2.5 ACE\_CCO.1.5C**

1389 **For each PP-Module identified in the components list of the PP-Configuration, the list contains**  
 1390 **at least one of its sets of base PPs and PP-Modules.**

1391 **8.9.2.2.6 ACE\_CCO.1.6C**

1392 **The conformance statement shall specify the required conformance to the PP-Configuration as**  
 1393 **one of exact, strict, demonstrable or multiple**

1394 **8.9.2.2.7 ACE\_CCO.1.7C**

1395 **For a multiple conformance PP-Configuration, the conformance statement shall specify the list**  
 1396 **of conformance types inherited from the PPs and PP-Modules that transitively belong to the**  
 1397 **components list of the PP-Configuration.**

1398 **8.9.2.2.8 ACE\_CCO.1.8C**

1399 **The consistency rationale shall demonstrate that the union of all the PPs and PP-Modules that**  
 1400 **transitively belong to the PP-Configuration's components list is consistent.**

1401 **8.9.2.3 Evaluator action elements**

1402 **8.9.2.3.1 ACE\_CCO.1.1E**

1403 **The evaluator shall confirm that the information provided meets all requirements for content**  
 1404 **and presentation of evidence.**

1405 **8.9.2.3.2 ACE\_CCO.1.2E**

1406 **The evaluator shall check that the PP-Configuration made up of all the Protection Profiles and**  
 1407 **PP-Modules identified in the components statement of the PP-Configuration is consistent.**

1408 **9 Class ASE: Security Target evaluation**

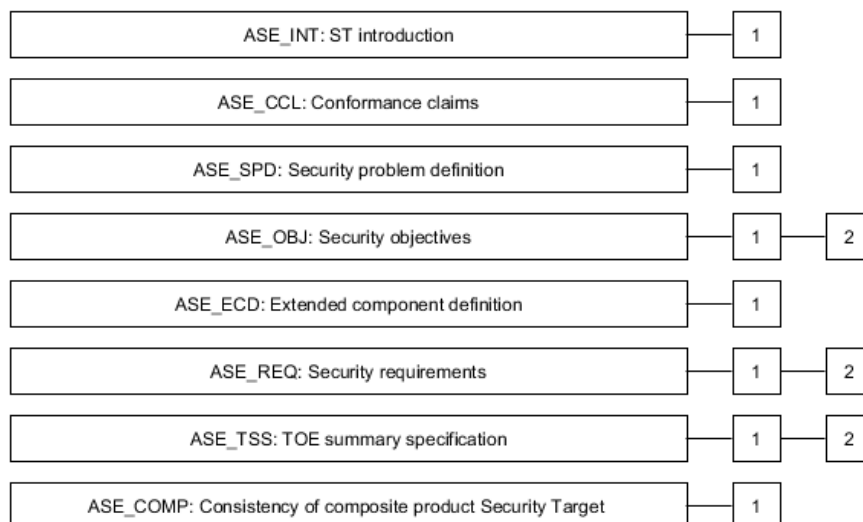
1409 **Editor's Note (Multi EAL approach): The ASE class must be extended to cover the conformity with one or**  
 1410 **more PPs/PP Configurations and potentially the addition of supplementary security problem, objectives**  
 1411 **and SFRs. The same kind of check as for PP-Modules and PP-Configurations apply. These updates will be**  
 1412 **provided once the proposed updates to the ACE class (in Section Fehler! Verweisquelle konnte nicht**  
 1413 **gefunden werden.) have been agreed.**

1414 **9.1 Introduction**

1415 Evaluating an ST is required to demonstrate that the ST is sound and internally consistent, and, if the  
 1416 ST is based on a PP-Configuration, or one or more PPs or packages, that the ST is a correct  
 1417 instantiation of the PP-Configuration, PPs, and packages. These properties are necessary for the ST to  
 1418 be suitable for use as the basis for a TOE evaluation.

1419 This clause should be used in conjunction with Annexes A, B and C in ISO/IEC 15408-1:2009, as these  
 1420 annexes clarify the concepts here and provide many examples.

1421 Figure 6 shows the families within this class, and the hierarchy of components within the families.



1422

1423 **Figure 6 — ASE: Security Target evaluation class decomposition**

## 1424 9.2 ST introduction (ASE\_INT)

### 1425 9.2.1 Objectives

1426 The objective of this family is to describe the TOE in a narrative way on three levels of abstraction:  
1427 TOE reference, TOE overview and TOE description.

1428 Evaluation of the ST introduction is required to demonstrate that the ST and the TOE are correctly  
1429 identified, that the TOE is correctly described at three levels of abstraction and that these three  
1430 descriptions are consistent with each other.

### 1431 9.2.2 ASE\_INT.1 ST introduction

1432 Dependencies: No dependencies.

#### 1433 9.2.2.1 Developer action elements

##### 1434 9.2.2.1.1 ASE\_INT.1.1D

1435 **The developer shall provide an ST introduction.**

#### 1436 9.2.2.2 Content and presentation elements

##### 1437 9.2.2.2.1 ASE\_INT.1.1C

1438 **The ST introduction shall contain an ST reference, a TOE reference, a TOE overview and a TOE**  
1439 **description.**

##### 1440 9.2.2.2.2 ASE\_INT.1.2C

1441 **The ST reference shall uniquely identify the ST.**



- 1442 **9.2.2.2.3 ASE\_INT.1.3C**
- 1443 **The TOE reference shall uniquely identify the TOE.**
- 1444 **9.2.2.2.4 ASE\_INT.1.4C**
- 1445 **The TOE overview shall summarise the usage and major security features of the TOE.**
- 1446 **9.2.2.2.5 ASE\_INT.1.5C**
- 1447 **The TOE overview shall identify the TOE type.**
- 1448 **9.2.2.2.6 ASE\_INT.1.6C**
- 1449 **The TOE overview shall identify any non-TOE hardware/software/firmware required by the**  
 1450 **TOE.**
- 1451 **9.2.2.2.7 ASE\_INT.1.7C**
- 1452 **The TOE description shall describe the physical scope of the TOE.**
- 1453 **9.2.2.2.8 ASE\_INT.1.8C**
- 1454 **The TOE description shall describe the logical scope of the TOE.**
- 1455 **9.2.2.3 Evaluator action elements**
- 1456 **9.2.2.3.1 ASE\_INT.1.1E**
- 1457 **The evaluator shall confirm that the information provided meets all requirements for content**  
 1458 **and presentation of evidence.**
- 1459 **9.2.2.3.2 ASE\_INT.1.2E**
- 1460 **The evaluator shall confirm that the TOE reference, the TOE overview, and the TOE description**  
 1461 **are consistent with each other.**
- 1462 **9.3 Conformance claims (ASE\_CCL)**
- 1463 **9.3.1 Objectives**
- 1464 The objective of this family is to determine the validity of the conformance claim. In addition, this  
 1465 family specifies how STs are to claim conformance with the PP or PP-Configuration..
- 1466 **9.3.2 ASE\_CCL.1 Conformance claims**
- 1467 Dependencies: ASE\_INT.1 ST introduction
- 1468 ASE\_ECD.1 Extended components definition
- 1469 ASE\_REQ.1 Stated security requirements

1470 **9.3.2.1 Developer action elements**

1471 **9.3.2.1.1 ASE\_CCL.1.1D**

1472 **The developer shall provide a conformance claim.**

1473 **9.3.2.1.2 ASE\_CCL.1.2D**

1474 **The developer shall provide a conformance claim rationale.**

1475 **9.3.2.2 Content and presentation elements**

1476 **9.3.2.2.1 ASE\_CCL.1.1C**

1477 **The conformance claim shall contain an ISO/IEC 15408 conformance claim that identifies the**  
 1478 **version of ISO/IEC 15408 to which the ST and the TOE claim conformance.**

1479 **9.3.2.2.2 ASE\_CCL.1.2C**

1480 **ISO/IEC 15408 conformance claim shall describe the conformance of the ST to ISO/IEC 15408-2**  
 1481 **as either ISO/IEC 15408-2 conformant or ISO/IEC 15408-2 extended.**

1482 **9.3.2.2.3 ASE\_CCL.1.3C**

1483 **ISO/IEC 15408 conformance claim shall describe the conformance of the ST to this part of**  
 1484 **ISO/IEC 15408 as either this part of ISO/IEC 15408 conformant or this part of ISO/IEC 15408**  
 1485 **extended.**

1486 **9.3.2.2.4 ASE\_CCL.1.4C**

1487 **ISO/IEC 15408 conformance claim shall be consistent with the extended components**  
 1488 **definition.**

1489 **9.3.2.2.5 ASE\_CCL.1.5C**

1490 **The conformance claim shall identify a PP-Configuration, or all PPs and security requirement**  
 1491 **packages to which the ST claims conformance.**

1492 **9.3.2.2.6 ASE\_CCL.1.6C**

1493 **The conformance claim shall describe any conformance of the ST to a package as either**  
 1494 **package-conformant or package-augmented.**

1495 **9.3.2.2.7 ASE\_CCL.1.7C**

1496 **The conformance claim rationale shall demonstrate that the TOE type is consistent with the**  
 1497 **TOE type in the PP-Configuration or PPs for which conformance is being claimed.**

1498 **9.3.2.2.8 ASE\_CCL.1.8C**

1499 **The conformance claim rationale shall demonstrate that the statement of the security problem**  
 1500 **definition is consistent with the statement of the security problem definition in the PP-**  
 1501 **Configuration or PPs for which conformance is being claimed.**

1502 **9.3.2.2.9 ASE\_CCL.1.9C**

1503 **The conformance claim rationale shall demonstrate that the statement of security objectives is**  
 1504 **consistent with the statement of security objectives in the PP-Configuration or PPs for which**  
 1505 **conformance is being claimed.**

1506 **9.3.2.2.10 ASE\_CCL.1.10C**

1507 **The conformance claim rationale shall demonstrate that the statement of security**  
 1508 **requirements is consistent with the statement of security requirements in the PP-Configuration**  
 1509 **or PPs for which conformance is being claimed.**

1510 **9.3.2.2.11 ASE\_CCL.1.11C**

1511 The conformance claim for PP(s) and PP-Configuration(s) shall be exact, strict, or demonstrable.

1512 **9.3.2.3 Evaluator action elements**

1513 **9.3.2.3.1 ASE\_CCL.1.1E**

1514 **The evaluator shall confirm that the information provided meets all requirements for content**  
 1515 **and presentation of evidence.**

1516 **9.4 Security problem definition (ASE\_SPD)**

1517 **9.4.1 Objectives**

1518 This part of the ST defines the security problem to be addressed by the TOE and the operational  
 1519 environment of the TOE.

1520 Evaluation of the security problem definition is required to demonstrate that the security problem  
 1521 intended to be addressed by the TOE and its operational environment, is clearly defined.

1522 **9.4.2 ASE\_SPD.1 Security problem definition**

1523 Dependencies: No dependencies.

1524 **9.4.2.1 Developer action elements**

1525 **9.4.2.1.1 ASE\_SPD.1.1D**

1526 **The developer shall provide a security problem definition.**

1527 **9.4.2.2 Content and presentation elements**

1528 **9.4.2.2.1 ASE\_SPD.1.1C**

1529 **The security problem definition shall describe the threats.**

1530 **9.4.2.2.2 ASE\_SPD.1.2C**

1531 **All threats shall be described in terms of a threat agent, an asset, and an adverse action.**

1532 **9.4.2.2.3 ASE\_SPD.1.3C**

1533 **The security problem definition shall describe the OSPs.**

1534 **9.4.2.2.4 ASE\_SPD.1.4C**

1535 **The security problem definition shall describe the assumptions about the operational**  
 1536 **environment of the TOE.**

1537 **9.4.2.3 Evaluator action elements**

1538 **9.4.2.3.1 ASE\_SPD.1.1E**

1539 **The evaluator shall confirm that the information provided meets all requirements for content**  
 1540 **and presentation of evidence.**

1541 **9.5 Security objectives (ASE\_OBJ)**

1542 **9.5.1 Objectives**

1543 The security objectives are a concise statement of the intended response to the security problem  
 1544 defined through the Security problem definition (ASE\_SPD) family.

1545 Evaluation of the security objectives is required to demonstrate that the security objectives adequately  
 1546 and completely address the security problem definition, that the division of this problem between the  
 1547 TOE and its operational environment is clearly defined.

1548 **9.5.2 Component levelling**

1549 The components in this family are levelled on whether they prescribe only security objectives for the  
 1550 operational environment, or also security objectives for the TOE.

1551 **9.5.3 ASE\_OBJ.1 Security objectives for the operational environment**

1552 Dependencies: No dependencies

1553 **9.5.3.1 Developer action elements**

1554 **9.5.3.1.1 ASE\_OBJ.1.1D**

1555 **The developer shall provide a statement of security objectives.**

1556 **9.5.3.2 Content and presentation elements**

1557 **9.5.3.2.1 ASE\_OBJ.1.1C**

1558 **The statement of security objectives shall describe the security objectives for the operational**  
 1559 **environment.**

1560 **9.5.3.3 Evaluator action elements**

1561 **9.5.3.3.1 ASE\_OBJ.1.1E**

1562 **The evaluator shall confirm that the information provided meets all requirements for content**  
 1563 **and presentation of evidence.**

1564 **9.5.4 ASE\_OBJ.2 Security objectives**

1565 Dependencies: ASE\_SPD.1 Security problem definition

1566 **9.5.4.1 Developer action elements**

1567 **9.5.4.1.1 ASE\_OBJ.2.1D**

1568 The developer shall provide a statement of security objectives.

1569 **9.5.4.1.2 ASE\_OBJ.2.2D**

1570 **The developer shall provide a security objectives rationale.**

1571 **9.5.4.2 Content and presentation elements**

1572 **9.5.4.2.1 ASE\_OBJ.2.1C**

1573 **The statement of security objectives shall describe the security objectives for the TOE and the**  
 1574 **security objectives for the operational environment.**

1575 **9.5.4.2.2 ASE\_OBJ.2.2C**

1576 **The security objectives rationale shall trace each security objective for the TOE back to threats**  
 1577 **countered by that security objective and OSPs enforced by that security objective.**

1578 **9.5.4.2.3 ASE\_OBJ.2.3C**

1579 **The security objectives rationale shall trace each security objective for the operational**  
 1580 **environment back to threats countered by that security objective, OSPs enforced by that**  
 1581 **security objective, and assumptions upheld by that security objective.**

1582 **9.5.4.2.4 ASE\_OBJ.2.4C**

1583 The security objectives **rationale** shall **demonstrate that** the security objectives **counter all threats.**

1584 **9.5.4.2.5 ASE\_OBJ.2.5C**

1585 **The security objectives rationale shall demonstrate that the security objectives enforce all**  
 1586 **OSP.**

1587 **9.5.4.2.6 ASE\_OBJ.2.6C**

1588 **The security objectives rationale shall demonstrate that the security objectives for the**  
 1589 **operational environment uphold all assumptions.**

1590 **9.5.4.3 Evaluator action elements**

1591 **9.5.4.3.1 ASE\_OBJ.2.1E**

1592 The evaluator shall confirm that the information provided meets all requirements for content and  
 1593 presentation of evidence.

1594 **9.6 Extended components definition (ASE\_ECD)**

1595 **9.6.1 Objectives**

1596 Extended security requirements are requirements that are not based on components from ISO/IEC  
 1597 15408-2 or this part of ISO/IEC 15408, but are based on extended components: components defined  
 1598 by the ST author.

1599 Evaluation of the definition of extended components is necessary to determine that they are clear and  
 1600 unambiguous, and that they are necessary, i.e. they may not be clearly expressed using existing  
 1601 ISO/IEC 15408-2 or this part of ISO/IEC 15408 components.

## 1602 **9.6.2 ASE\_ECD.1 Extended components definition**

1603 Dependencies: No dependencies.

### 1604 **9.6.2.1 Developer action elements**

#### 1605 **9.6.2.1.1 ASE\_ECD.1.1D**

1606 **The developer shall provide a statement of security requirements.**

#### 1607 **9.6.2.1.2 ASE\_ECD.1.2D**

1608 **The developer shall provide an extended components definition.**

### 1609 **9.6.2.2 Content and presentation elements**

#### 1610 **9.6.2.2.1 ASE\_ECD.1.1C**

1611 **The statement of security requirements shall identify all extended security requirements.**

#### 1612 **9.6.2.2.2 ASE\_ECD.1.2C**

1613 **The extended components definition shall define an extended component for each extended**  
 1614 **security requirement.**

#### 1615 **9.6.2.2.3 ASE\_ECD.1.3C**

1616 **The extended components definition shall describe how each extended component is related to**  
 1617 **the existing ISO/IEC 15408 components, families, and classes.**

#### 1618 **9.6.2.2.4 ASE\_ECD.1.4C**

1619 **The extended components definition shall use the existing ISO/IEC 15408 components,**  
 1620 **families, classes, and methodology as a model for presentation.**

#### 1621 **9.6.2.2.5 ASE\_ECD.1.5C**

1622 **The extended components shall consist of measurable and objective elements such that**  
 1623 **conformance or nonconformance to these elements can be demonstrated.**

### 1624 **9.6.2.3 Evaluator action elements**

#### 1625 **9.6.2.3.1 ASE\_ECD.1.1E**

1626 **The evaluator shall confirm that the information provided meets all requirements for content**  
 1627 **and presentation of evidence.**

#### 1628 **9.6.2.3.2 ASE\_ECD.1.2E**

1629 **The evaluator shall confirm that no extended component can be clearly expressed using**  
 1630 **existing components.**

## 1631 **9.7 Security requirements (ASE\_REQ)**

### 1632 **9.7.1 Objectives**

1633 The SFRs form a clear, unambiguous and well-defined description of the expected security behaviour  
 1634 of the TOE. The SARs form a clear, unambiguous and canonical description of the expected activities  
 1635 that will be undertaken to gain assurance in the TOE.

1636 Evaluation of the security requirements is required to ensure that they are clear, unambiguous and  
 1637 well-defined.

### 1638 **9.7.2 Component levelling**

1639 The components in this family are levelled on whether they are stated as is.

### 1640 **9.7.3 ASE\_REQ.1 Stated security requirements**

1641 Dependencies: ASE\_ECD.1 Extended components definition

#### 1642 **9.7.3.1 Developer action elements**

##### 1643 **9.7.3.1.1 ASE\_REQ.1.1D**

1644 **The developer shall provide a statement of security requirements.**

##### 1645 **9.7.3.1.2 ASE\_REQ.1.2D**

1646 **The developer shall provide a security requirements rationale.**

#### 1647 **9.7.3.2 Content and presentation elements**

##### 1648 **9.7.3.2.1 ASE\_REQ.1.1C**

1649 **The statement of security requirements shall describe the SFRs and the SARs.**

##### 1650 **9.7.3.2.2 ASE\_REQ.1.2C**

1651 **All subjects, objects, operations, security attributes, external entities and other terms that are**  
 1652 **used in the SFRs and the SARs shall be defined.**

##### 1653 **9.7.3.2.3 ASE\_REQ.1.3C**

1654 **The statement of security requirements shall include a natural language description, part of**  
 1655 **which describes how the SFRs combine together to provide security functionality in terms of**  
 1656 **the architecture that is observable to Administrators and other users, or in terms of internal**  
 1657 **features or properties.**

##### 1658 **9.7.3.2.4 ASE\_REQ.1.4C**

1659 **The statement of security requirements shall identify all operations on the security**  
 1660 **requirements.**

##### 1661 **9.7.3.2.5 ASE\_REQ.1.5C**

1662 **All operations shall be performed correctly.**

1663 **9.7.3.2.6 ASE\_REQ.1.6C**

1664 **Each dependency of the security requirements shall either be satisfied, or the security**  
 1665 **requirements rationale shall justify the dependency not being satisfied.**

1666 **9.7.3.2.7 ASE\_REQ.1.7C**

1667 **The security requirements rationale shall trace each SFR back to the security objectives threats**  
 1668 **countered by that SFR and OSPs enforced by that SFR.**

1669 **9.7.3.2.8 ASE\_REQ.1.8C**

1670 **The security requirements rationale shall trace each security objective for the operational**  
 1671 **environment back to threats countered by that security objective, OSPs enforced by that**  
 1672 **security objective, and assumptions upheld by that security objective.**

1673 **9.7.3.2.9 ASE\_REQ.1.9C**

1674 **The security requirements rationale shall demonstrate that the SFRs counter all threats for the**  
 1675 **TOE.**

1676 **9.7.3.2.10 ASE\_REQ.1.10C**

1677 **The security requirements rationale shall demonstrate that the SFRs enforce all OSPs.**

1678 **9.7.3.2.11 ASE\_REQ.1.11C**

1679 **The security requirements rationale shall demonstrate that the security objectives for the**  
 1680 **operational environment uphold all assumptions.**

1681 **9.7.3.2.12 ASE\_REQ.1.12C**

1682 **The statement of security requirements shall be internally consistent.**

1683 **9.7.3.3 Evaluator action elements**

1684 **9.7.3.3.1 ASE\_REQ.1.1E**

1685 **The evaluator shall confirm that the information provided meets all requirements for content**  
 1686 **and presentation of evidence.**

1687 **9.7.4 ASE\_REQ.2 Derived security requirements**

1688 Dependencies: ASE\_OBJ.2 Security objectives

1689 ASE\_ECD.1 Extended components definition

1690 **9.7.4.1 Developer action elements**

1691 **9.7.4.1.1 ASE\_REQ.2.1D**

1692 The developer shall provide a statement of security requirements.

1693 **9.7.4.1.2 ASE\_REQ.2.2D**

1694 The developer shall provide a security requirements rationale.



1695 **9.7.4.2 Content and presentation elements**

1696 **9.7.4.2.1 ASE\_REQ.2.1C**

1697 The statement of security requirements shall describe the SFRs and the SARs.

1698 **9.7.4.2.2 ASE\_REQ.2.2C**

1699 All subjects, objects, operations, security attributes, external entities and other terms that are used in  
1700 the SFRs and the SARs shall be defined.

1701 **9.7.4.2.3 ASE\_REQ.2.3C**

1702 The statement of security requirements shall include a natural language description, part of which  
1703 describes how the SFRs combine together to provide security functionality in terms of the architecture  
1704 that is observable to Administrators and other users, or in terms of internal features or properties.

1705 **9.7.4.2.4 ASE\_REQ.2.4C**

1706 The statement of security requirements shall identify all operations on the security requirements.

1707 **9.7.4.2.5 ASE\_REQ.2.5C**

1708 All operations shall be performed correctly.

1709 **9.7.4.2.6 ASE\_REQ.2.6C**

1710 Each dependency of the security requirements shall either be satisfied, or the security requirements  
1711 rationale shall justify the dependency not being satisfied.

1712 **9.7.4.2.7 ASE\_REQ.2.7C**

1713 The security requirements rationale shall trace each SFR back to the security objectives **for the TOE**.

1714 **9.7.4.2.8 ASE\_REQ.2.8C**

1715 **The security requirements rationale shall demonstrate that the SFRs meet all security**  
1716 **objectives for the TOE.**

1717 **9.7.4.2.9 ASE\_REQ.2.9C**

1718 **The security requirements rationale shall explain why the SARs were chosen.**

1719 **9.7.4.2.10 ASE\_REQ.2.10C**

1720 The statement of security requirements shall be internally consistent.

1721 **9.7.4.2.11 ASE\_REQ.2.11C**

1722 **9.7.4.3** The security requirements rationale shall demonstrate that the security objectives for the  
1723 operational environment uphold all assumptions.**Evaluator action elements**

1724 **9.7.4.3.1 ASE\_REQ.2.1E**

1725 The evaluator shall confirm that the information provided meets all requirements for content and  
1726 presentation of evidence.

1727 **9.8 TOE summary specification (ASE\_TSS)**

1728 **9.8.1 Objectives**

1729 The TOE summary specification enables evaluators and potential consumers to gain a general  
1730 understanding of how the TOE is implemented.

1731 Evaluation of the TOE summary specification is necessary to determine whether it is adequately  
1732 described how the TOE:

- 1733 • meets its SFRs;
- 1734 • protects itself against interference, logical tampering and bypass;

1735 and whether the TOE summary specification is consistent with other narrative descriptions of the  
1736 TOE.

1737 **9.8.2 Component levelling**

1738 The components in this family are levelled on whether the TOE summary specification only needs to  
1739 describe how the TOE meets the SFRs, or whether the TOE summary specification also needs to  
1740 describe how the TOE protects itself against logical tampering and bypass. This additional description  
1741 may be used in special circumstances where there might be a specific concern regarding the TOE  
1742 security architecture.

1743 **9.8.3 ASE\_TSS.1 TOE summary specification**

1744 Dependencies: ASE\_INT.1 ST introduction

1745 ASE\_REQ.1 Stated security requirements

1746 ADV\_FSP.1 Basic functional specification

1747 **9.8.3.1 Developer action elements**

1748 **9.8.3.1.1 ASE\_TSS.1.1D**

1749 **The developer shall provide a TOE summary specification.**

1750 **9.8.3.2 Content and presentation elements**

1751 **9.8.3.2.1 ASE\_TSS.1.1C**

1752 **The TOE summary specification shall describe how the TOE meets each SFR.**

1753 **9.8.3.3 Evaluator action elements**

1754 **9.8.3.3.1 ASE\_TSS.1.1E**

1755 **The evaluator shall confirm that the information provided meets all requirements for content**  
1756 **and presentation of evidence.**

1757 **9.8.3.3.2 ASE\_TSS.1.2E**

1758 **The evaluator shall confirm that the TOE summary specification is consistent with the TOE**  
 1759 **overview and the TOE description.**

1760 **9.8.4 ASE\_TSS.2 TOE summary specification with architectural design summary**

1761 Dependencies: ASE\_INT.1 ST introduction

1762 ASE\_REQ.1 Stated security requirements

1763 ADV\_ARC.1 Security architecture description

1764 **9.8.4.1 Developer action elements**

1765 **9.8.4.1.1 ASE\_TSS.2.1D**

1766 The developer shall provide a TOE summary specification.

1767 **9.8.4.2 Content and presentation elements**

1768 **9.8.4.2.1 ASE\_TSS.2.1C**

1769 The TOE summary specification shall describe how the TOE meets each SFR.

1770 **9.8.4.2.2 ASE\_TSS.2.2C**

1771 **The TOE summary specification shall describe how the TOE protects itself against interference**  
 1772 **and logical tampering.**

1773 **9.8.4.2.3 ASE\_TSS.2.3C**

1774 **The TOE summary specification shall describe how the TOE protects itself against bypass.**

1775 **9.8.4.3 Evaluator action elements**

1776 **9.8.4.3.1 ASE\_TSS.2.1E**

1777 The evaluator shall confirm that the information provided meets all requirements for content and  
 1778 presentation of evidence.

1779 **9.8.4.3.2 ASE\_TSS.2.2E**

1780 The evaluator shall confirm that the TOE summary specification is consistent with the TOE overview  
 1781 and the TOE description.

1782 **9.9 Consistency of composite product Security Target (ASE\_COMP)**

1783 **9.9.1 Objectives**

1784 The aim of this activity is to determine whether the Security Target of the composite product<sup>1</sup> does not  
 1785 contradict the Security Target of the underlying platform<sup>2</sup>.

---

<sup>1</sup> denoted by Composite-ST in the following

## 1786 9.9.2 ASE\_COMP.1 Consistency of Security Target

1787 Dependencies: No dependencies

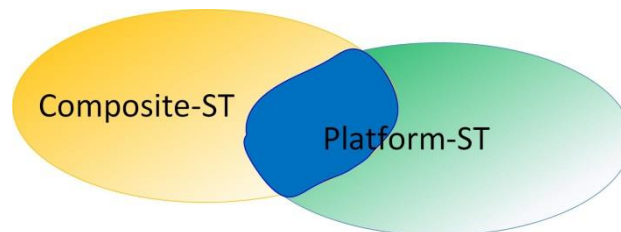
### 1788 9.9.2.1 Application Notes

1789 These application notes aid the developer to create as well as the evaluator to analyse a composite  
1790 Security Target and describe a general methodology for it. For detailed information / guidance please  
1791 refer to the single work units below.

1792 In order to create a composite Security Target, the developer should perform the following steps:

1793 Step 1: The developer formulates a preliminary Security Target for the composite product (the  
1794 Composite-ST) using the standard code of practice. The Composite-ST can be formulated  
1795 independently of the Security Target of the underlying platform (Platform-ST) – at least as long as  
1796 there are no formal PP conformance claims.

1797 Step 2: The developer determines the overlap between Platform-ST and Composite-ST through  
1798 analysing and comparing their TOE Security Functionality (TSF) <sup>34</sup>:



1799

1800 **Figure 7 - Overlap between Platform-ST and Composite-ST**

1801 Step 3: The developer determines under which conditions he can trust in and rely on the Platform-  
1802 TSF being used by the Composite-ST without a new examination.

1803 Having undertaken these steps the developer completes the preliminary Security Target for the  
1804 composite product.

1805 It is not mandatory that the platform and the composite TOE are being certified according to same  
1806 version of the CC. It is due to the fact that the application can rely on some security services of the  
1807 platform, if (i) the assurance level of the platform covers the intended assurance level of the composite  
1808 TOE and (ii) the platform's security certificate is valid and up-to-date. Equivalence of single assurance  
1809 components (and, hence, of assurance levels) belonging to different **ISO/IEC 15408 editions** shall be  
1810 established / acknowledged by the Composite Product Certification Body, cf. chapter D.3 **D.3**.

1811 If a PP conformance is claimed (e.g. composite ST claim conformance to a PP that claims conformance  
1812 to a hardware PP), the consistency check can be reduced to the elements of the Security Target having  
1813 not already been covered by these Protection Profiles.

<sup>2</sup> denoted by Platform-ST in the following. Generally, a Security Target expresses a security policy for the TOE defined.

<sup>3</sup> because the TSF enforce the Security Target (together with organisational measures enforcing security objectives for the operational environment of the TOE).

<sup>4</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.

1814 The fact of compliance to a PP is not sufficient to avoid inconsistencies. Assume the following situation,  
1815 where → stands for “complies with”

1816 Composite-ST → SW PP → HW PP ← platform-ST

1817 The SW PP may require any kind of conformance<sup>5</sup>, but this does not change the ‘additional elements’  
1818 that the platform-ST may introduce to the HW PP. In conclusion, these additions are not necessarily  
1819 consistent with the composite-ST/SW PP additions: There is no scenario that ensures the consistency  
1820 ‘by construction’.

1821 Note that consistency may not be direct matching: e.g. objectives for the platform environment may  
1822 become objectives for the composite TOE.

### 1823 **9.9.2.2 Developer action elements**

#### 1824 **9.9.2.2.1 ASE\_COMP.1.1D**

1825 **The developer shall provide a statement of compatibility between the Composite Security**  
1826 **Target and the Platform Security Target. This statement can be provided within the Composite**  
1827 **Product Security Target.**

### 1828 **9.9.2.3 Content and presentation elements**

#### 1829 **9.9.2.3.1 ASE\_COMP.1.1C**

1830 **The statement of compatibility shall describe the separation of the Platform-TSF into relevant**  
1831 **Platform-TSF being used by the Composite-ST and others.**

#### 1832 **9.9.2.3.2 ASE\_COMP.1.2C**

1833 **The statement of compatibility between the Composite Security Target and the Platform**  
1834 **Security Target shall show (e.g. in form of a mapping) that the Security Targets of the composite**  
1835 **product and of the underlying platform match, i.e. that there is no conflict between security**  
1836 **environments, security objectives, and security requirements of the Composite Security Target**  
1837 **and the Platform Security Target. It can be provided by indicating of the concerned elements**  
1838 **directly in the Security Target for the composite product followed by explanatory text, if**  
1839 **necessary.**

### 1840 **9.9.2.4 Evaluator action elements**

#### 1841 **9.9.2.4.1 ASE\_COMP.1.1E**

1842 **The evaluator shall confirm that the information provided meets all requirements for content**  
1843 **and presentation of evidence.**

---

<sup>5</sup> e.g. “strict” or “demonstrable” according to CC V3.

## 10 Class ADV: Development

### 10.1 Introduction

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

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

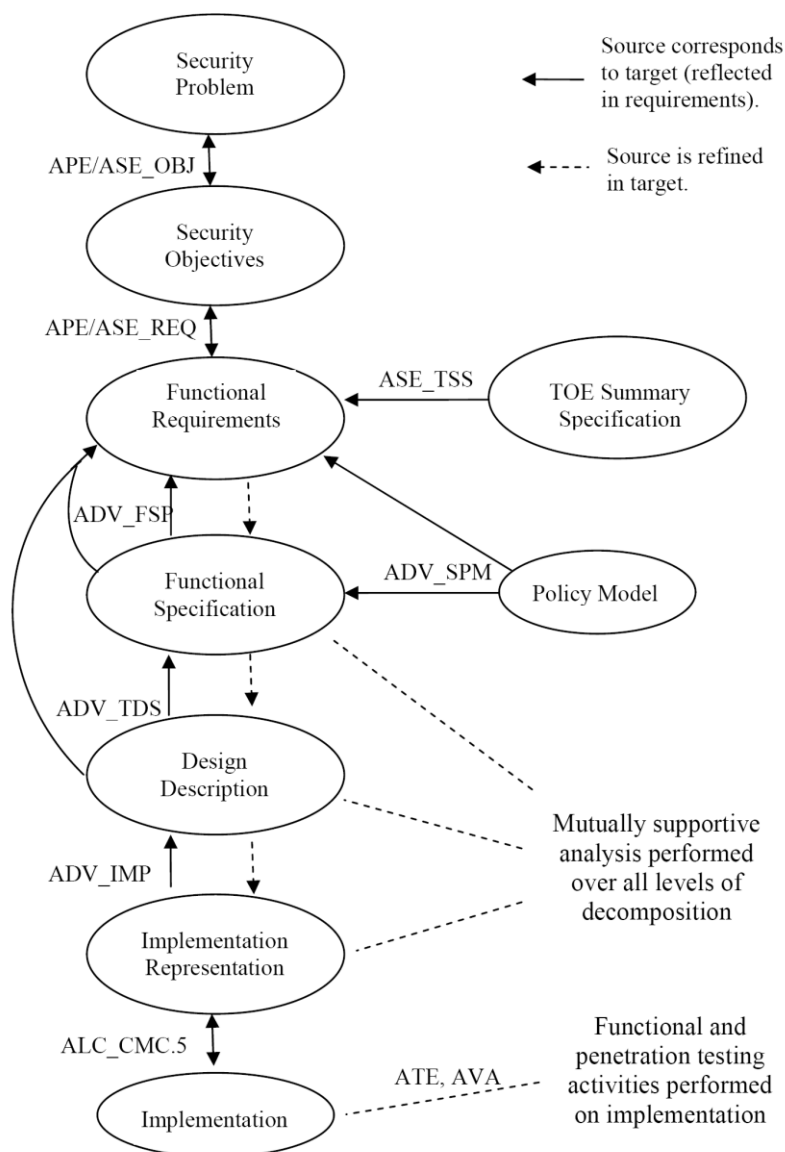
- requirements for the description (at the various levels of abstraction) of the design and implementation of the SFRs (ADV\_FSP, ADV\_TDS, ADV\_IMP and ADV\_COMP)
- requirements for the description of the architecture-oriented features of domain separation, TSF self-protection and non-bypassability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

When documenting the security functionality of a TOE, there are two properties that need to be demonstrated. The first property is that the security functionality works correctly; that is, it performs as specified. The second property, and one that is arguably harder to demonstrate, is that the TOE cannot be used in a way such that the security functionality can be corrupted or bypassed. These two properties require somewhat different approaches in analysis, and so the families in ADV are structured to support these different approaches. The families Functional specification (ADV\_FSP), TOE design (ADV\_TDS), Implementation representation (ADV\_IMP), and Security policy modelling (ADV\_SPM) deal with the first property: the specification of the security functionality. The families Security Architecture (ADV\_ARC) and TSF internals (ADV\_INT) deal with the second property: the specification of the design of the TOE demonstrating the security functionality cannot be corrupted or bypassed. It should be noted that both properties need to be realised: the more confidence one has that the properties are satisfied, the more trustworthy the TOE is. The components in the families are designed so that more assurance can be gained as the components hierarchically increase.

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

Figure 8 indicates the relationships among the various TSF representations of the ADV class, as well as their relationships with other classes. As the figure indicates, the APE and ASE classes define the requirements for the correspondence between the SFRs and the security objectives for the TOE. Class ASE also defines requirements for the correspondence between both the security objectives and SFRs,

1887 and for the TOE summary specification which explains how the TOE meets its SFRs. The activities of  
 1888 ALC\_CMC.5.2E include the verification that the TSF that is tested under the ATE and AVA classes is in  
 1889 fact the one described by all of the ADV decomposition levels.



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

1892 The requirements for all other correspondence shown in Figure 8 are defined in the ADV class for the  
 1893 TOE. The Security policy modelling (ADV\_SPM) family defines the requirements for formally modelling  
 1894 selected SFRs, and providing correspondence between the functional specification and the formal  
 1895 model. Each assurance family specific to a TSF representation (i.e., Functional specification (ADV\_FSP),  
 1896 TOE design (ADV\_TDS) and Implementation representation (ADV\_IMP)) defines requirements relating  
 1897 that TSF representation to the SFRs. All decompositions must accurately reflect all other  
 1898 decompositions (i.e., be mutually supportive); the developer supplies the tracings in the last .C  
 1899 elements of the components. Assurance relating to this factor is obtained during the analysis for each  
 1900 of the levels of decomposition by referring to other levels of decomposition (in a recursive fashion)  
 1901 while the analysis of a particular level of decomposition is being performed; the evaluator verifies the

- 1902 correspondence as part of the second E element. The understanding gained from these levels of  
1903 decomposition form the basis of the functional and penetration testing efforts.
- 1904 The ADV\_INT family is not represented in this figure, as it is related to the internal structure of the TSF,  
1905 and is only indirectly related to the process of refinement of the TSF representations. Similarly, the  
1906 ADV\_ARC family is not represented in the figure because it relates to the architectural soundness,  
1907 rather than representation, of the TSF. Both ADV\_INT and ADV\_ARC relate to the analysis of the  
1908 property that the TOE cannot be made to circumvent or corrupt its security functionality.
- 1909 The TOE security functionality (TSF) consists of all parts of the TOE that have to be relied upon for  
1910 enforcement of the SFRs. The TSF includes both functionality that directly enforces the SFRs, as well as  
1911 functionality that, while not directly enforcing the SFRs, contributes to their enforcement in a more  
1912 indirect manner, including functionality with the capability to cause the SFRs to be violated. This  
1913 includes portions of the TOE that are invoked on start-up that are responsible for putting the TSF into  
1914 its initial secure state.
- 1915 Several important concepts were used in the development of the components of the ADV families.  
1916 These concepts, while introduced briefly here, are explained more fully in the application notes for the  
1917 families.
- 1918 One over-riding notion is that, as more information becomes available, greater assurance can be  
1919 obtained that the security functionality 1) is correctly implemented; 2) cannot be corrupted; and 3)  
1920 cannot be bypassed. This is done through the verification that the documentation is correct and  
1921 consistent with other documentation, and by providing information that can be used to ensure that the  
1922 testing activities (both functional and penetration testing) are comprehensive. This is reflected in the  
1923 levelling of the components of the families. In general, components are levelled based on the amount of  
1924 information that is to be provided (and subsequently analysed).
- 1925 While not true for all TOEs, it is generally the case that the TSF is sufficiently complex that there are  
1926 portions of the TSF that deserve more intense examination than other portions of the TSF.  
1927 Determining those portions is unfortunately somewhat subjective, thus terminology and components  
1928 have been defined such that as the level of assurance increases, the responsibility for determining  
1929 what portions of the TSF need to be examined in detail shifts from the developer to the evaluator. To  
1930 aid in expressing this concept, the following terminology is introduced. It should be noted that in the  
1931 families of the class, this terminology is used when expressing SFR-related portions of the TOE (that is,  
1932 elements and work units embodied in the Functional specification (ADV\_FSP), TOE design (ADV\_TDS),  
1933 and Implementation representation (ADV\_IMP) families). While the general concept (that some  
1934 portions of the TOE are more *interesting* than others) applies to other families, the criteria are  
1935 expressed differently in order to obtain the assurance required.
- 1936 All portions of the TSF are *security relevant*, meaning that they must preserve the security of the TOE  
1937 as expressed by the SFRs and requirements for domain separation and non-bypassability. One aspect  
1938 of security relevance is the degree to which a portion of the TSF enforces a security requirement. Since  
1939 different portions of the TOE play different roles (or no apparent role at all) in enforcing security  
1940 requirements, this creates a continuum of SFR relevance: at one end of this continuum are portions of  
1941 the TOE that are termed *SFR-enforcing*. Such portions play a direct role in implementing any SFR on  
1942 the TOE. Such SFRs refer to any functionality provided by one of the SFRs contained in the ST. It should  
1943 be noted that the definition of *plays a role in* for SFR-enforcing functionality is impossible to express  
1944 quantitatively. For example, in the implementation of a Discretionary Access Control (DAC)  
1945 mechanism, a very narrow view of *SFR-enforcing* might be the several lines of code that actually  
1946 perform the check of a subject's attributes against the object's attributes. A broader view would  
1947 include the software entity (e.g., C function) that contained the several lines of code. A broader view  
1948 still would include callers of the C function, since they would be responsible for enforcing the decision



- 1949 returned by the attribute check. A still broader view would include any code in the call tree (or  
 1950 programming equivalent for the implementation language used) for that C function (e.g., a sort  
 1951 function that sorted access control list entries in a first-match algorithm implementation). At some  
 1952 point, the component is not so much *enforcing* the security policy but rather plays a *supporting* role;  
 1953 such components are termed *SFR supporting*. One of the characteristics of SFR-supporting  
 1954 functionality is that it is trusted to preserve the correctness of the SFR implementation by operating  
 1955 without error. Such functionality may be depended on by SFR-enforcing functionality, but the  
 1956 dependence is generally at a functional level; for example, memory management, buffer management,  
 1957 etc. Further down on the security relevance continuum is functionality termed *SFR non-interfering*.  
 1958 Such functionality has no role in implementing the SFRs, and is likely part of the TSF because of its  
 1959 environment; for example, any code running in a privileged hardware mode on an operating system. It  
 1960 needs to be considered part of the TSF because, if compromised (or replaced by malicious code), it  
 1961 could compromise the correct operation of an SFR by virtue of its operating in the privileged hardware  
 1962 mode. An example of SFR non-interfering functionality might be a set of mathematical floating point  
 1963 operations implemented in kernel mode for speed considerations.
- 1964 The architecture family (Security Architecture (ADV\_ARC)) provides for requirements and analysis of  
 1965 the TOE based on properties of domain separation, self-protection, and non-bypassability. These  
 1966 properties relate to the SFRs in that, if these properties are not present, it will likely lead to the failure  
 1967 of mechanisms implementing SFRs. Functionality and design relating to these properties *is not*  
 1968 considered a part of the continuum described above, but instead is treated separately due to its  
 1969 fundamentally different nature and analysis requirements.
- 1970 The difference in analysis of the implementation of SFRs (SFR-enforcing and SFR-supporting  
 1971 functionality) and the implementation of somewhat fundamental security properties of the TOE, which  
 1972 include the initialisation, self-protection, and non-bypassability concerns, is that the SFR-related  
 1973 functionality is more or less directly visible and relatively easy to test, while the above-mentioned  
 1974 properties require varying degrees of analysis on a much broader set of functionality. Further, the  
 1975 depth of analysis for such properties will vary depending on the design of the TOE. The ADV families  
 1976 are constructed to address this by a separate family (Security Architecture (ADV\_ARC)) devoted to  
 1977 analysis of the initialisation, self-protection, and non-bypassability requirements, while the other  
 1978 families are concerned with analysis of the functionality supporting SFRs.
- 1979 Even in cases where different descriptions are necessary for the multiple levels of abstraction, it is not  
 1980 absolutely necessary for each and every TSF representation to be in a separate document. Indeed, it  
 1981 may be the case that a single document meets the documentation requirements for more than one TSF  
 1982 representation, since it is the information about each of these TSF representations that is required,  
 1983 rather than the resulting document structure. In cases where multiple TSF representations are  
 1984 combined within a single document, the developer should indicate which portions of the documents  
 1985 meet which requirements.
- 1986 Three types of specification style are mandated by this class: informal, semiformal and formal. The  
 1987 functional specification and TOE design documentation are always written in either informal or  
 1988 semiformal style. A semiformal style reduces the ambiguity in these documents over an informal  
 1989 presentation. A formal specification may also be required *in addition to* the semi-formal presentation;  
 1990 the value is that a description of the TSF in more than one way will add increased assurance that the  
 1991 TSF has been completely and accurately specified.
- 1992 An informal specification is written as prose in natural language. Natural language is used here as  
 1993 meaning communication in any commonly spoken tongue (e.g. Spanish, German, French, English,  
 1994 Dutch). An informal specification is not subject to any notational or special restrictions other than  
 1995 those required as ordinary conventions for that language (e.g. grammar and syntax). While no

1996 notational restrictions apply, the informal specification is also required to provide defined meanings  
 1997 for terms that are used in a context other than that accepted by normal usage.

1998 The difference between semiformal and informal documents is only a matter of formatting or  
 1999 presentation: a semiformal notation includes such things as an explicit glossary of terms, a  
 2000 standardised presentation format, etc. A semiformal specification is written to a standard presentation  
 2001 template. The presentation should use terms consistently if written in a natural language. The  
 2002 presentation may also use more structured languages/diagrams (e.g. data-flow diagrams, state  
 2003 transition diagrams, entity-relationship diagrams, data structure diagrams, and process or program  
 2004 structure diagrams). Whether based on diagrams or natural language, a set of conventions must be  
 2005 used in the presentation. The glossary explicitly identifies the words that are being used in a precise  
 2006 and constant manner; similarly, the standardised format implies that extreme care has been taken in  
 2007 methodically preparing the document in a manner that maximises clarity. It should be noted that  
 2008 fundamentally different portions of the TSF may have different semiformal notation conventions and  
 2009 presentation styles (as long as the number of different “semiformal notations” is small); this still  
 2010 conforms to the concept of a *semiformal presentation*.

2011 A formal specification is written in a notation based upon well-established mathematical concepts, and  
 2012 is typically accompanied by supporting explanatory (informal) prose. These mathematical concepts  
 2013 are used to define the syntax and semantics of the notation and the proof rules that support logical  
 2014 reasoning. The syntactic and semantic rules supporting a formal notation should define how to  
 2015 recognise constructs unambiguously and determine their meaning. There needs to be evidence that it  
 2016 is impossible to derive contradictions, and all rules supporting the notation need to be defined or  
 2017 referenced.

2018 Figure 9 shows the families within this class, and the hierarchy of components within the families.

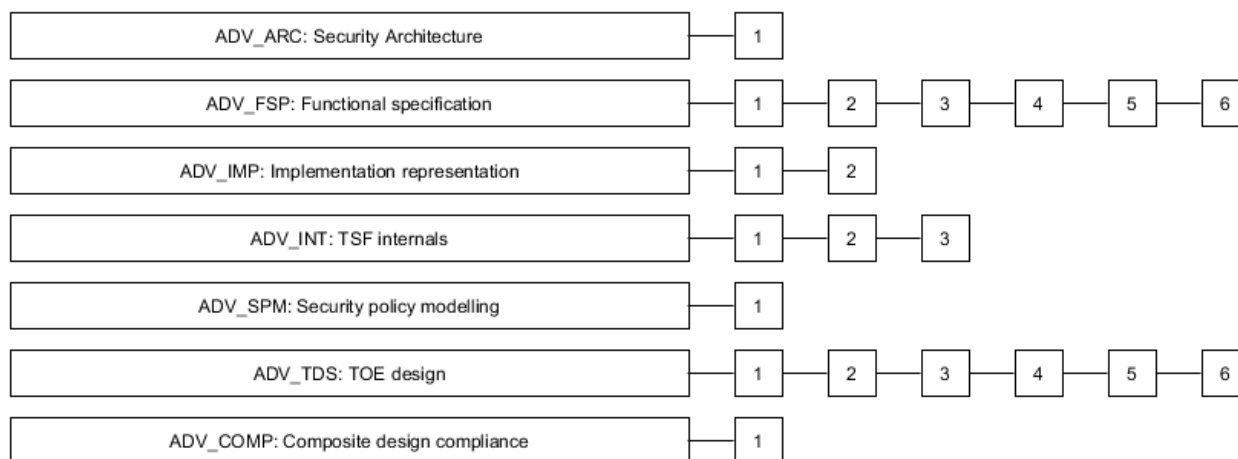


Figure 9 — ADV: Development class decomposition

## 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

2027 examining the TSF; without a sound architecture, the entire TOE functionality would have to be  
2028 examined.

### 2029 **10.2.2 Component levelling**

2030 This family contains only one component.

### 2031 **10.2.3 Application notes**

2032 The properties of self-protection, domain separation, and non-bypassability are distinct from security  
2033 functionality expressed by **ISO/IEC 15408-2** SFRs because self-protection and non-bypassability  
2034 largely have no directly observable interface at the TSF. Rather, they are properties of the TSF that are  
2035 achieved through the design of the TOE and TSF, and enforced by the correct implementation of that  
2036 design.

2037 The approach used in this family is for the developer to design and provide a TSF that exhibits the  
2038 above-mentioned properties, and to provide evidence (in the form of documentation) that explains  
2039 these properties of the TSF. This explanation is provided at the same level of detail as the description  
2040 of the SFR-enforcing elements of the TOE in the TOE design document. The evaluator has the  
2041 responsibility for looking at the evidence and, coupled with other evidence delivered for the TOE and  
2042 TSF, determining that the properties are achieved.

2043 Specification of security functionality implementing the SFRs (in the Functional specification  
2044 (ADV\_FSP) and TOE design (ADV\_TDS)) will not necessarily describe mechanisms employed in  
2045 implementing self-protection and non-bypassability (e.g. memory management mechanisms).  
2046 Therefore, the material needed to provide the assurance that these requirements are being achieved is  
2047 better suited to a presentation separate from the design decomposition of the TSF as embodied in  
2048 ADV\_FSP and ADV\_TDS. This is not to imply that the security architecture description called for by this  
2049 component cannot reference or make use of the design decomposition material; but it is likely that  
2050 much of the detail present in the decomposition documentation will not be relevant to the argument  
2051 being provided for the security architecture description document.

2052 The description of architectural soundness can be thought of as a developer's vulnerability analysis, in  
2053 that it provides the justification for why the TSF is sound and enforces all of its SFRs. Where the  
2054 soundness is achieved through specific security mechanisms, these will be tested as part of the Depth  
2055 (ATE\_DPT) requirements; where the soundness is achieved solely through the architecture, the  
2056 behaviour will be tested as part of the AVA: Vulnerability assessment requirements.

2057 This family consists of requirements for a security architecture description that describes the self-  
2058 protection, domain separation, non-bypassability principles, including a description of how these  
2059 principles are supported by the parts of the TOE that are used for TSF initialisation.

2060 Additional information on the security architecture properties of self-protection, domain separation,  
2061 and non-bypassability can be found in Annex A.1, ADV\_ARC: Supplementary material on security  
2062 architectures.

### 2063 **10.2.4 ADV\_ARC.1 Security architecture description**

2064 Dependencies: ADV\_FSP.1 Basic functional specification

2065 ADV\_TDS.1 Basic design

2066 **10.2.4.1 Developer action elements**

2067 **10.2.4.1.1 ADV\_ARC.1.1D**

2068 **The developer shall design and implement the TOE so that the security features of the TSF**  
2069 **cannot be bypassed.**

2070 **10.2.4.1.2 ADV\_ARC.1.2D**

2071 **The developer shall design and implement the TSF so that it is able to protect itself from**  
2072 **tampering by untrusted active entities.**

2073 **10.2.4.1.3 ADV\_ARC.1.3D**

2074 **The developer shall provide a security architecture description of the TSF.**

2075 **10.2.4.2 Content and presentation elements**

2076 **10.2.4.2.1 ADV\_ARC.1.1C**

2077 **The security architecture description shall be at a level of detail commensurate with the**  
2078 **description of the SFR-enforcing abstractions described in the TOE design document.**

2079 **10.2.4.2.2 ADV\_ARC.1.2C**

2080 **The security architecture description shall describe the security domains maintained by the**  
2081 **TSF consistently with the SFRs.**

2082 **10.2.4.2.3 ADV\_ARC.1.3C**

2083 **The security architecture description shall describe how the TSF initialisation process is**  
2084 **secure.**

2085 **10.2.4.2.4 ADV\_ARC.1.4C**

2086 **The security architecture description shall demonstrate that the TSF protects itself from**  
2087 **tampering.**

2088 **10.2.4.2.5 ADV\_ARC.1.5C**

2089 **The security architecture description shall demonstrate that the TSF prevents bypass of the**  
2090 **SFR-enforcing functionality.**

2091 **10.2.4.3 Evaluator action elements**

2092 **10.2.4.3.1 ADV\_ARC.1.1E**

2093 **The evaluator shall confirm that the information provided meets all requirements for content**  
2094 **and presentation of evidence.**

2095 **10.3 Functional specification (ADV\_FSP)**

2096 **10.3.1 Objectives**

2097 **This family levies requirements upon the functional specification, which describes the TSF interfaces**  
2098 **(TSFIs). The TSFI consists of all means by which external entities (or subjects in the TOE but outside of**

2099 the TSF) supply data to the TSF, receive data from the TSF and invoke services from the TSF. It does  
 2100 *not* describe how the TSF processes those service requests, nor does it describe the communication  
 2101 when the TSF invokes services from its operational environment; this information is addressed by the  
 2102 TOE design (ADV\_TDS) and Reliance of dependent component (ACO\_REL) families, respectively.

2103 This family provides assurance directly by allowing the evaluator to understand how the TSF meets  
 2104 the claimed SFRs. It also provides assurance indirectly, as input to other assurance families and  
 2105 classes:

- 2106 • ADV\_ARC, where the description of the TSFIs may be used to gain better understanding of how the  
 2107 TSF is protected against corruption (i.e. subversion of self-protection or domain separation)  
 2108 and/or bypass;
- 2109 • ATE, where the description of the TSFIs is an important input for both developer and evaluator  
 2110 testing;
- 2111 • AVA, where the description of the TSFIs is used to search for vulnerabilities.

### 2112 10.3.2 Component levelling

2113 The components in this family are levelled on the degree of detail required of the description of the  
 2114 TSFIs, and the degree of formalism required of the description of the TSFIs.

### 2115 10.3.3 Application notes

2116 Once the TSFIs are determined (see XXX for guidance and examples of determining TSFI), they are  
 2117 described. At lower-level components, developers focus their documentation (and evaluators focus  
 2118 their analysis) on the more security-relevant aspects of the TOE. Three categories of TSFIs are defined,  
 2119 based upon the relevance the services available through them have to the SFRs being claimed:

- 2120 • If a service available through an interface can be traced to one of the SFRs levied on the TSF,  
 2121 then that interface is termed *SFR-enforcing*. Note that it is possible that an interface may have  
 2122 various services and results, some of which may be SFR-enforcing and some of which may not.
- 2123 • Interfaces to (or services available through an interface relating to) services that SFR-enforcing  
 2124 functionality depends upon, but need only to function correctly in order for the security  
 2125 policies of the TOE to be preserved, are termed *SFR-supporting*.
- 2126 • Interfaces to services on which SFR-enforcing functionality has no dependence are termed *SFR*  
 2127 *non-interfering*.

2128 It should be noted that in order for an interface to be SFR-supporting or SFR non-interfering it must  
 2129 have *no* SFR-enforcing services or results. In contrast, an SFR-enforcing interface may have SFR-  
 2130 supporting services (for example, the ability to set the system clock may be an SFR-enforcing service of  
 2131 an interface, but if that same interface is used to display the system date that service may be only SFR-  
 2132 supporting). An example of a purely SFR-supporting interface is a system call interface that is used  
 2133 both by users and by a portion of the TSF that is running on behalf of users.

2134 As more information about the TSFIs becomes available, the greater the assurance that can be gained  
 2135 that the interfaces are correctly categorised/analysed. The requirements are structured such that, at  
 2136 the lowest level, the information required for SFR non-interfering interfaces is the minimum necessary  
 2137 in order for the evaluator to make this determination in an effective manner. At higher levels, more  
 2138 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.

2183 The *error message description* identifies the condition that generated it, what the message is, and the  
 2184 meaning of any error codes. An error message is generated by the TSF to signify that a problem or  
 2185 irregularity of some degree has been encountered. The requirements in this family refer to different  
 2186 kinds of error messages:

- 2187 • a “direct” error message is a security-relevant response through a specific TSFI invocation.
- 2188 • an “indirect” error cannot be tied to a specific TSFI invocation because it results from system-wide  
 2189 conditions (e.g. resource exhaustion, connectivity interruptions, etc.). Error messages that are not  
 2190 security-relevant are also considered “indirect”.
- 2191 • “remaining” errors are any other errors, such as those that might be referenced within the code.  
 2192 For example, the use of condition-checking code that checks for conditions that would not logically  
 2193 occur (e.g. a final “else” after a list of “case” statements), would provide for generating a catch-all  
 2194 error message; in an operational TOE, these error messages should never be seen.

2195 An example functional specification is provided in A.2.4.

### 2196 **10.3.3.2 Components of this Family**

2197 Increasing assurance through increased completeness and accuracy in the interface specification is  
 2198 reflected in the documentation required from the developer as detailed in the various hierarchical  
 2199 components of this family.

2200 At ADV\_FSP.1 Basic functional specification, the only documentation required is a characterisation of  
 2201 all TSFIs and a high level description of SFR-enforcing and SFR-supporting TSFIs. To provide some  
 2202 assurance that the “important” aspects of the TSF have been correctly characterised at the TSFIs, the  
 2203 developer is required to provide the purpose and method of use, parameters for the SFR-enforcing and  
 2204 SFR-supporting TSFIs.

2205 At ADV\_FSP.2 Security-enforcing functional specification, the developer is required to provide the  
 2206 purpose, method of use, parameters, and parameter descriptions for all TSFIs. Additionally, for the  
 2207 SFR-enforcing TSFIs the developer has to describe the SFR-enforcing actions and direct error  
 2208 messages.

2209 At ADV\_FSP.3 Functional specification with complete summary, the developer must now, in addition to  
 2210 the information required at ADV\_FSP.2, provide enough information about the SFR-supporting and  
 2211 SFR-non-interfering actions to show that they are not SFR-enforcing. Further, the developer must now  
 2212 document all of the direct error messages resulting from the invocation of SFR-enforcing TSFIs.

2213 At ADV\_FSP.4 Complete functional specification, all TSFIs - whether SFR-enforcing, SFR-supporting or  
 2214 SFR-non-interfering - must be described to the same degree, including all of the direct error messages.

2215 At ADV\_FSP.5 Complete semi-formal functional specification with additional error information, the  
 2216 TSFIs descriptions also include error messages that do not result from an invocation of a TSFI.

2217 At ADV\_FSP.6 Complete semi-formal functional specification with additional formal specification, in  
 2218 addition to the information required by ADV\_FSP.5, all remaining error messages are included. The  
 2219 developer must also provide a formal description of the TSFI. This provides an alternative view of the  
 2220 TSFI that may expose inconsistencies or incomplete specification.

### 2221 **10.3.4 ADV\_FSP.1 Basic functional specification**

2222 Dependencies: No dependencies.

- 2223 **10.3.4.1 Developer action elements**
- 2224 **10.3.4.1.1 ADV\_FSP.1.1D**
- 2225 **The developer shall provide a functional specification.**
- 2226 **10.3.4.1.2 ADV\_FSP.1.2D**
- 2227 **The developer shall provide a tracing from the functional specification to the SFRs.**
- 2228 **10.3.4.2 Content and presentation elements**
- 2229 **10.3.4.2.1 ADV\_FSP.1.1C**
- 2230 **The functional specification shall describe the purpose and method of use for each SFR-**  
 2231 **enforcing and SFR-supporting TSFI.**
- 2232 **10.3.4.2.2 ADV\_FSP.1.2C**
- 2233 **The functional specification shall identify all parameters associated with each SFR-enforcing**  
 2234 **and SFR-supporting TSFI.**
- 2235 **10.3.4.2.3 ADV\_FSP.1.3C**
- 2236 **The functional specification shall provide rationale for the implicit categorisation of interfaces**  
 2237 **as SFR-non-interfering.**
- 2238 **10.3.4.2.4 ADV\_FSP.1.4C**
- 2239 **The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.**
- 2240 **10.3.4.3 Evaluator action elements**
- 2241 **10.3.4.3.1 ADV\_FSP.1.1E**
- 2242 **The evaluator shall confirm that the information provided meets all requirements for content**  
 2243 **and presentation of evidence.**
- 2244 **10.3.4.3.2 ADV\_FSP.1.2E**
- 2245 **The evaluator shall determine that the functional specification is an accurate and complete**  
 2246 **instantiation of the SFRs.**
- 2247 **10.3.5 ADV\_FSP.2 Security-enforcing functional specification**
- 2248 Dependencies: ADV\_TDS.1 Basic design
- 2249 **10.3.5.1 Developer action elements**
- 2250 **10.3.5.1.1 ADV\_FSP.2.1D**
- 2251 **The developer shall provide a functional specification.**
- 2252 **10.3.5.1.2 ADV\_FSP.2.2D**
- 2253 **The developer shall provide a tracing from the functional specification to the SFRs.**



- 2254 **10.3.5.2 Content and presentation elements**
- 2255 **10.3.5.2.1 ADV\_FSP.2.1C**
- 2256 **The functional specification shall completely represent the TSF.**
- 2257 **10.3.5.2.2 ADV\_FSP.2.2C**
- 2258 The functional specification shall describe the purpose and method of use for **all** TSFI.
- 2259 **10.3.5.2.3 ADV\_FSP.2.3C**
- 2260 The functional specification shall identify **and describe** all parameters associated with each TSFI.
- 2261 **10.3.5.2.4 ADV\_FSP.2.4C**
- 2262 **For each SFR-enforcing TSFI, the functional specification shall describe the SFR-enforcing actions**  
 2263 **associated with the TSFI.**
- 2264 **10.3.5.2.5 ADV\_FSP.2.5C**
- 2265 **For each SFR-enforcing TSFI, the functional specification shall describe direct error messages**  
 2266 **resulting from processing associated with the SFR-enforcing actions.**
- 2267 **10.3.5.2.6 ADV\_FSP.2.6C**
- 2268 The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.
- 2269 **10.3.5.3 Evaluator action elements**
- 2270 **10.3.5.3.1 ADV\_FSP.2.1E**
- 2271 The evaluator shall confirm that the information provided meets all requirements for content and  
 2272 presentation of evidence.
- 2273 **10.3.5.3.2 ADV\_FSP.2.2E**
- 2274 The evaluator shall determine that the functional specification is an accurate and complete  
 2275 instantiation of the SFRs.
- 2276 **10.3.6 ADV\_FSP.3 Functional specification with complete summary**
- 2277 Dependencies: ADV\_TDS.1 Basic design
- 2278 **10.3.6.1 Developer action elements**
- 2279 **10.3.6.1.1 ADV\_FSP.3.1D**
- 2280 The developer shall provide a functional specification.
- 2281 **10.3.6.1.2 ADV\_FSP.3.2D**
- 2282 The developer shall provide a tracing from the functional specification to the SFRs.

- 2283 **10.3.6.2 Content and presentation elements**
- 2284 **10.3.6.2.1 ADV\_FSP.3.1C**
- 2285 The functional specification shall completely represent the TSF.
- 2286 **10.3.6.2.2 ADV\_FSP.3.2C**
- 2287 The functional specification shall describe the purpose and method of use for all TSFI.
- 2288 **10.3.6.2.3 ADV\_FSP.3.3C**
- 2289 The functional specification shall identify and describe all parameters associated with each TSFI.
- 2290 **10.3.6.2.4 ADV\_FSP.3.4C**
- 2291 For each SFR-enforcing TSFI, the functional specification shall describe the SFR-enforcing actions  
2292 associated with the TSFI.
- 2293 **10.3.6.2.5 ADV\_FSP.3.5C**
- 2294 For each SFR-enforcing TSFI, the functional specification shall describe direct error messages resulting  
2295 from **SFR-enforcing actions and exceptions** associated with **invocation of the TSFI**.
- 2296 **10.3.6.2.6 ADV\_FSP.3.6C**
- 2297 **The functional specification shall summarise the SFR-supporting and SFR-non-interfering**  
2298 **actions associated with each TSFI.**
- 2299 **10.3.6.2.7 ADV\_FSP.3.7C**
- 2300 The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.
- 2301 **10.3.6.3 Evaluator action elements**
- 2302 **10.3.6.3.1 ADV\_FSP.3.1E**
- 2303 The evaluator shall confirm that the information provided meets all requirements for content and  
2304 presentation of evidence.
- 2305 **10.3.6.3.2 ADV\_FSP.3.2E**
- 2306 The evaluator shall determine that the functional specification is an accurate and complete  
2307 instantiation of the SFRs.
- 2308 **10.3.7 ADV\_FSP.4 Complete functional specification**
- 2309 Dependencies: ADV\_TDS.1 Basic design
- 2310 **10.3.7.1 Developer action elements**
- 2311 **10.3.7.1.1 ADV\_FSP.4.1D**
- 2312 The developer shall provide a functional specification.

2313 **10.3.7.1.2 ADV\_FSP.4.2D**

2314 The developer shall provide a tracing from the functional specification to the SFRs.

2315 **10.3.7.2 Content and presentation elements**

2316 **10.3.7.2.1 ADV\_FSP.4.1C**

2317 The functional specification shall completely represent the TSF.

2318 **10.3.7.2.2 ADV\_FSP.4.2C**

2319 The functional specification shall describe the purpose and method of use for all TSFI.

2320 **10.3.7.2.3 ADV\_FSP.4.3C**

2321 The functional specification shall identify and describe all parameters associated with each TSFI.

2322 **10.3.7.2.4 ADV\_FSP.4.4C**

2323 The functional specification shall **describe all** actions associated with each TSFI.

2324 **10.3.7.2.5 ADV\_FSP.4.5C**

2325 **The functional specification shall describe all direct error messages that may result from an**  
 2326 **invocation of each TSFI.**

2327 **10.3.7.2.6 ADV\_FSP.4.6C**

2328 The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

2329 **10.3.7.3 Evaluator action elements**

2330 **10.3.7.3.1 ADV\_FSP.4.1E**

2331 The evaluator shall confirm that the information provided meets all requirements for content and  
 2332 presentation of evidence.

2333 **10.3.7.3.2 ADV\_FSP.4.2E**

2334 The evaluator shall determine that the functional specification is an accurate and complete  
 2335 instantiation of the SFRs.

2336 **10.3.8 ADV\_FSP.5 Complete semi-formal functional specification with additional error**  
 2337 **information**

2338 Dependencies: ADV\_TDS.1 Basic design

2339 ADV\_IMP.1 Implementation representation of the TSF

2340 **10.3.8.1 Developer action elements**

2341 **10.3.8.1.1 ADV\_FSP.5.1D**

2342 The developer shall provide a functional specification.

2343 **10.3.8.1.2 ADV\_FSP.5.2D**

2344 The developer shall provide a tracing from the functional specification to the SFRs.

2345 **10.3.8.2 Content and presentation elements**

2346 **10.3.8.2.1 ADV\_FSP.5.1C**

2347 The functional specification shall completely represent the TSF.

2348 **10.3.8.2.2 ADV\_FSP.5.2C**

2349 **The functional specification shall describe the TSFI using a semi-formal style.**

2350 **10.3.8.2.3 ADV\_FSP.5.3C**

2351 The functional specification shall describe the purpose and method of use for all TSFI.

2352 **10.3.8.2.4 ADV\_FSP.5.4C**

2353 The functional specification shall identify and describe all parameters associated with each TSFI.

2354 **10.3.8.2.5 ADV\_FSP.5.5C**

2355 The functional specification shall describe all actions associated with each TSFI.

2356 **10.3.8.2.6 ADV\_FSP.5.6C**

2357 The functional specification shall describe all direct error messages that may result from an invocation  
2358 of each TSFI.

2359 **10.3.8.2.7 ADV\_FSP.5.7C**

2360 **The functional specification shall describe all error messages that do not result from an**  
2361 **invocation of a TSFI.**

2362 **10.3.8.2.8 ADV\_FSP.5.8C**

2363 **The functional specification shall provide a rationale for each error message contained in the**  
2364 **TSF implementation yet does not result from an invocation of a TSFI.**

2365 **10.3.8.2.9 ADV\_FSP.5.9C**

2366 The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

2367 **10.3.8.3 Evaluator action elements**

2368 **10.3.8.3.1 ADV\_FSP.5.1E**

2369 The evaluator shall confirm that the information provided meets all requirements for content and  
2370 presentation of evidence.

2371 **10.3.8.3.2 ADV\_FSP.5.2E**

2372 The evaluator shall determine that the functional specification is an accurate and complete  
2373 instantiation of the SFRs.

2374	<b>10.3.9 ADV_FSP.6 Complete semi-formal functional specification with additional formal</b>
2375	<b>specification</b>
2376	Dependencies: ADV_TDS.1 Basic design
2377	ADV_IMP.1 Implementation representation of the TSF
2378	<b>10.3.9.1 Developer action elements</b>
2379	<b>10.3.9.1.1 ADV_FSP.6.1D</b>
2380	The developer shall provide a functional specification.
2381	<b>10.3.9.1.2 ADV_FSP.6.2D</b>
2382	<b>The developer shall provide a formal presentation of the functional specification of the TSF.</b>
2383	<b>10.3.9.1.3 ADV_FSP.6.3D</b>
2384	The developer shall provide a tracing from the functional specification to the SFRs.
2385	<b>10.3.9.2 Content and presentation elements</b>
2386	<b>10.3.9.2.1 ADV_FSP.6.1C</b>
2387	The functional specification shall completely represent the TSF.
2388	<b>10.3.9.2.2 ADV_FSP.6.2C</b>
2389	The functional specification shall describe the TSFI using a <b>formal</b> style.
2390	<b>10.3.9.2.3 ADV_FSP.6.3C</b>
2391	The functional specification shall describe the purpose and method of use for all TSFI.
2392	<b>10.3.9.2.4 ADV_FSP.6.4C</b>
2393	The functional specification shall identify and describe all parameters associated with each TSFI.
2394	<b>10.3.9.2.5 ADV_FSP.6.5C</b>
2395	The functional specification shall describe all actions associated with each TSFI.
2396	<b>10.3.9.2.6 ADV_FSP.6.6C</b>
2397	The functional specification shall describe all direct error messages that may result from an invocation
2398	of each TSFI.
2399	<b>10.3.9.2.7 ADV_FSP.6.7C</b>
2400	The functional specification shall describe all error messages <b>contained in the TSF implementation</b>
2401	<b>representation.</b>

2402 **10.3.9.2.8 ADV\_FSP.6.8C**

2403 The functional specification shall provide a rationale for each error message contained in the TSF  
 2404 implementation **that is not otherwise described in the functional specification justifying why it is**  
 2405 **not associated with** a TSFI.

2406 **10.3.9.2.9 ADV\_FSP.6.9C**

2407 **The formal presentation of the functional specification of the TSF shall describe the TSFI using**  
 2408 **a formal style, supported by informal, explanatory text where appropriate.**

2409 **10.3.9.2.10 ADV\_FSP.6.10C**

2410 The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

2411 **10.3.9.3 Evaluator action elements**

2412 **10.3.9.3.1 ADV\_FSP.6.1E**

2413 The evaluator shall confirm that the information provided meets all requirements for content and  
 2414 presentation of evidence.

2415 **10.3.9.3.2 ADV\_FSP.6.2E**

2416 The evaluator shall determine that the functional specification is an accurate and complete  
 2417 instantiation of the SFRs.

2418 **10.4 Implementation representation (ADV\_IMP)**

2419 **10.4.1 Objectives**

2420 The function of the Implementation representation (ADV\_IMP) family is for the developer to make  
 2421 available the implementation representation (and, at higher levels, the implementation itself) of the  
 2422 TOE in a form that can be analysed by the evaluator. The implementation representation is used in  
 2423 analysis activities for other families (analysing the TOE design, for instance) to demonstrate that the  
 2424 TOE conforms its design and to provide a basis for analysis in other areas of the evaluation (e.g., the  
 2425 search for vulnerabilities). The implementation representation is expected to be in a form that  
 2426 captures the detailed internal workings of the TSF. This may be software source code, firmware source  
 2427 code, hardware diagrams and/or IC hardware design language code or layout data.

2428 **10.4.2 Component levelling**

2429 The components in this family are levelled on the amount of implementation that is mapped to the  
 2430 TOE design description.

2431 **10.4.3 Application notes**

2432 Source code or hardware diagrams and/or IC hardware design language code or layout data that are  
 2433 used to build the actual hardware are examples of parts of an implementation representation. It is  
 2434 important to note that while the implementation representation must be made available to the  
 2435 evaluator, this does not imply that the evaluator needs to possess that representation. For instance,  
 2436 the developer may require that the evaluator review the implementation representation at a site of the  
 2437 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

2485	ALC_TAT.1 Well-defined development tools
2486	<b>10.4.4.1 Developer action elements</b>
2487	<b>10.4.4.1.1 ADV_IMP.1.1D</b>
2488	<b>The developer shall make available the implementation representation for the entire TSF.</b>
2489	<b>10.4.4.1.2 ADV_IMP.1.2D</b>
2490	<b>The developer shall provide a mapping between the TOE design description and the sample of</b>
2491	<b>the implementation representation.</b>
2492	<b>10.4.4.2 Content and presentation elements</b>
2493	<b>10.4.4.2.1 ADV_IMP.1.1C</b>
2494	<b>The implementation representation shall define the TSF to a level of detail such that the TSF</b>
2495	<b>can be generated without further design decisions.</b>
2496	<b>10.4.4.2.2 ADV_IMP.1.2C</b>
2497	<b>The implementation representation shall be in the form used by the development personnel.</b>
2498	<b>10.4.4.2.3 ADV_IMP.1.3C</b>
2499	<b>The mapping between the TOE design description and the sample of the implementation</b>
2500	<b>representation shall demonstrate their correspondence.</b>
2501	<b>10.4.4.3 Evaluator action elements</b>
2502	<b>10.4.4.3.1 ADV_IMP.1.1E</b>
2503	<b>The evaluator shall confirm that, for the selected sample of the implementation representation,</b>
2504	<b>the information provided meets all requirements for content and presentation of evidence.</b>
2505	<b>10.4.5 ADV_IMP.2 Complete mapping of the implementation representation of the TSF</b>
2506	Dependencies: ADV_TDS.3 Basic modular design
2507	ALC_TAT.1 Well-defined development tools
2508	ALC_CMC.5 Advanced support
2509	<b>10.4.5.1 Developer action elements</b>
2510	<b>10.4.5.1.1 ADV_IMP.2.1D</b>
2511	<b>The developer shall make available the implementation representation for the entire TSF.</b>
2512	<b>10.4.5.1.2 ADV_IMP.2.2D</b>
2513	<b>The developer shall provide a mapping between the TOE design description and the <b>entire</b></b>
2514	<b>implementation representation.</b>



2515 **10.4.5.2 Content and presentation elements**

2516 **10.4.5.2.1 ADV\_IMP.2.1C**

2517 The implementation representation shall define the TSF to a level of detail such that the TSF can be  
2518 generated without further design decisions.

2519 **10.4.5.2.2 ADV\_IMP.2.2C**

2520 The implementation representation shall be in the form used by the development personnel.

2521 **10.4.5.2.3 ADV\_IMP.2.3C**

2522 The mapping between the TOE design description and the **entire** implementation representation shall  
2523 demonstrate their correspondence.

2524 **10.4.5.3 Evaluator action elements**

2525 **10.4.5.3.1 ADV\_IMP.2.1E**

2526 The evaluator shall confirm that the information provided meets all requirements for content and  
2527 presentation of evidence.

2528 **10.5 TSF internals (ADV\_INT)**

2529 **10.5.1 Objectives**

2530 This family addresses the assessment of the internal structure of the TSF. A TSF whose internals are  
2531 well-structured is easier to implement and less likely to contain flaws that could lead to  
2532 vulnerabilities; it is also easier to maintain without the introduction of flaws.

2533 **10.5.2 Component levelling**

2534 The components in this family are levelled on the basis of the amount of structure and minimisation of  
2535 complexity required. ADV\_INT.1 Well-structured subset of TSF internals places requirements for well-  
2536 structured internals on only selected parts of the TSF. This component is not included in an EAL  
2537 because this component is viewed for use in special circumstances (e.g., the sponsor has a specific  
2538 concern regarding a cryptographic module, which is isolated from the rest of the TSF) and would not  
2539 be widely applicable.

2540 At the next level, the requirements for well-structured internals are placed on the entire TSF. Finally,  
2541 minimisation of complexity is introduced in the highest component.

2542 **10.5.3 Application notes**

2543 These requirements, when applied to the internal structure of the TSF, typically result in  
2544 improvements that aid both the developer and the evaluator in understanding the TSF, and also  
2545 provide the basis for designing and evaluating test suites. Further, improving understandability of the  
2546 TSF should assist the developer in simplifying its maintainability.

2547 The requirements in this family are presented at a fairly abstract level. The wide variety of TOEs  
2548 makes it impossible to codify anything more specific than “well-structured” or “minimum complexity”.  
2549 Judgements on structure and complexity are expected to be derived from the specific technologies  
2550 used in the TOE. For example, software is likely to be considered well-structured if it exhibits the  
2551 characteristics cited in the software engineering disciplines. The components within this family call for

2552 identifying the standards for measuring the characteristic of being well-structured and not overly-  
2553 complex.

#### 2554 **10.5.4 ADV\_INT.1 Well-structured subset of TSF internals**

2555 Dependencies: ADV\_IMP.1 Implementation representation of the TSF

2556 ADV\_TDS.3 Basic modular design

2557 ALC\_TAT.1 Well-defined development tools

##### 2558 **10.5.4.1 Objectives**

2559 The objective of this component is to provide a means for requiring specific portions of the TSF to be  
2560 well-structured. The intent is that the entire TSF has been designed and implemented using sound  
2561 engineering principles, but the analysis is performed upon only a specific subset.

##### 2562 **10.5.4.2 Application notes**

2563 This component requires the PP or ST author to fill in an assignment with the subset of the TSF. This  
2564 subset may be identified in terms of the internals of the TSF at any layer of abstraction. For example:

2565 a) the structural elements of the TSF as identified in the TOE design (e.g. "The developer shall design  
2566 and implement *the audit subsystem* such that it has well-structured internals.")

2567 b) the implementation (e.g. "The developer shall design and implement *the encrypt.c and decrypt.c*  
2568 *files* such that it has well-structured internals." or "The developer shall design and implement *the*  
2569 *6227 IC chip* such that it has well-structured internals.")

2570 It is likely this would not be readily accomplished by referencing the claimed SFRs (e.g. "The developer  
2571 shall design and implement *the portion of the TSF that provide anonymity as defined in FPR\_ANO.2* such  
2572 that it has well-structured internals.") because this does not indicate where to focus the analysis.

2573 This component has limited value and would be suitable in cases where potentially-malicious  
2574 users/subjects have limited or strictly controlled access to the TSFIs or where there is another means  
2575 of protection (e.g., domain separation) that ensures the chosen subset of the TSF cannot be adversely  
2576 affected by the rest of the TSF (e.g., the cryptographic functionality, which is isolated from the rest of  
2577 the TSF, is well-structured).

##### 2578 **10.5.4.3 Developer action elements**

###### 2579 **10.5.4.3.1 ADV\_INT.1.1D**

2580 **The developer shall design and implement [assignment: *subset of the TSF*] such that it has well-**  
2581 **structured internals.**

###### 2582 **10.5.4.3.2 ADV\_INT.1.2D**

2583 **The developer shall provide an internals description and justification.**

- 2584 **10.5.4.4 Content and presentation elements**
- 2585 **10.5.4.4.1 ADV\_INT.1.1C**
- 2586 **The justification shall explain the characteristics used to judge the meaning of “well-**  
 2587 **structured”.**
- 2588 **10.5.4.4.2 ADV\_INT.1.2C**
- 2589 **The TSF internals description shall demonstrate that the assigned subset of the TSF is well-**  
 2590 **structured.**
- 2591 **10.5.4.5 Evaluator action elements**
- 2592 **10.5.4.5.1 ADV\_INT.1.1E**
- 2593 **The evaluator shall confirm that the information provided meets all requirements for content**  
 2594 **and presentation of evidence.**
- 2595 **10.5.4.5.2 ADV\_INT.1.2E**
- 2596 **The evaluator shall perform an internals analysis on the assigned subset of the TSF.**
- 2597 **10.5.5 ADV\_INT.2 Well-structured internals**
- 2598 Dependencies: ADV\_IMP.1 Implementation representation of the TSF
- 2599 ADV\_TDS.3 Basic modular design
- 2600 ALC\_TAT.1 Well-defined development tools
- 2601 **10.5.5.1 Objectives**
- 2602 The objective of this component is to provide a means for requiring the TSF to be well-structured. The  
 2603 intent is that the entire TSF has been designed and implemented using sound engineering principles.
- 2604 **10.5.5.2 Application notes**
- 2605 Judgements on the adequacy of the structure are expected to be derived from the specific technologies  
 2606 used in the TOE. This component calls for identifying the standards for measuring the characteristic of  
 2607 being well-structured.
- 2608 **10.5.5.3 Developer action elements**
- 2609 **10.5.5.3.1 ADV\_INT.2.1D**
- 2610 The developer shall design and implement the **entire TSF** such that it has well-structured internals.
- 2611 **10.5.5.3.2 ADV\_INT.2.2D**
- 2612 The developer shall provide an internals description and justification.

2613 **10.5.5.4 Content and presentation elements**

2614 **10.5.5.4.1 ADV\_INT.2.1C**

2615 The justification shall **describe** the characteristics used to judge the meaning of “well-structured”.

2616 **10.5.5.4.2 ADV\_INT.2.2C**

2617 The TSF internals description shall demonstrate that the **entire** TSF is well-structured.

2618 **10.5.5.5 Evaluator action elements**

2619 **10.5.5.5.1 ADV\_INT.2.1E**

2620 The evaluator shall confirm that the information provided meets all requirements for content and  
2621 presentation of evidence.

2622 **10.5.5.5.2 ADV\_INT.2.2E**

2623 The evaluator shall perform an internals analysis on the TSF.

2624 **10.5.6 ADV\_INT.3 Minimally complex internals**

2625 Dependencies: ADV\_IMP.1 Implementation representation of the TSF

2626 ADV\_TDS.3 Basic modular design

2627 ALC\_TAT.1 Well-defined development tools

2628 **10.5.6.1 Objectives**

2629 The objective of this component is to provide a means for requiring the TSF to be well-structured and  
2630 of minimal complexity. The intent is that the entire TSF has been designed and implemented using  
2631 sound engineering principles.

2632 **10.5.6.2 Application notes**

2633 Judgements on the adequacy of the structure and complexity are expected to be derived from the  
2634 specific technologies used in the TOE. This component calls for identifying the standards for  
2635 measuring the structure and complexity.

2636 **10.5.6.3 Developer action elements**

2637 **10.5.6.3.1 ADV\_INT.3.1D**

2638 The developer shall design and implement the entire TSF such that it has well-structured internals.

2639 **10.5.6.3.2 ADV\_INT.3.2D**

2640 The developer shall provide an internals description and justification.

2641 **10.5.6.4 Content and presentation elements**

2642 **10.5.6.4.1 ADV\_INT.3.1C**

2643 The justification shall describe the characteristics used to judge the meaning of “well-structured” **and**  
2644 **“complex”**.

2645 **10.5.6.4.2 ADV\_INT.3.2C**

2646 The TSF internals description shall demonstrate that the entire TSF is well-structured **and is not**  
2647 **overly complex**.

2648 **10.5.6.5 Evaluator action elements**

2649 **10.5.6.5.1 ADV\_INT.3.1E**

2650 The evaluator shall confirm that the information provided meets all requirements for content and  
2651 presentation of evidence.

2652 **10.5.6.5.2 ADV\_INT.3.2E**

2653 The evaluator shall perform an internals analysis on the **entire** TSF.

2654 **10.6 Security policy modelling (ADV\_SPM)**

2655 **10.6.1 Objectives**

2656 It is the objective of this family to provide additional assurance from the development of a formal  
2657 *security policy model* of the TSFI behaviour of the TSF, and establishing a correspondence between the  
2658 functional specification and this security policy model. Preserving internal consistency the security  
2659 policy model is expected to formally establish the security principles from its characteristics by means  
2660 of a mathematical proof.

2661 **10.6.2 Component levelling**

2662 This family contains only one component.

2663 **10.6.3 Application notes**

2664 Inadequacies in a TOE can result either from a failure in understanding the security requirements or  
2665 from a flawed implementation of those security requirements. Defining the security requirements  
2666 adequately to ensure their understanding may be problematic because the definition must be  
2667 sufficiently precise to prevent undesired results or subtle flaws during implementation of the TOE.  
2668 Throughout the design, implementation, and review processes, the modelled security requirements  
2669 may and should be used as precise design and implementation guidance, thereby providing increased  
2670 assurance that the modelled security requirements modelled via the TSFI behaviour are satisfied by  
2671 the TOE. The precision of the model and resulting guidance is significantly improved by casting the  
2672 model in a formal language and verifying the security requirements by automated formal proof  
2673 techniques.

2674 The creation of a formal security policy model helps to identify and eliminate ambiguous, inconsistent,  
2675 incomplete, contradictory, or unenforceable security policy elements. Once the TOE has been built, the  
2676 formal model serves the evaluation effort by contributing to the evaluator's judgement of how well the  
2677 developer has understood the security functionality being implemented and whether there are  
2678 inconsistencies between the security requirements and the TOE design. The confidence in the model is

2679 accompanied by a correspondence analysis for model elements and the functional specification, and a  
2680 proof that the model contains no inconsistencies.

2681 A formal security policy model is a precise formal presentation of the important aspects of security  
2682 and their relationship to the behaviour of the TOE; it identifies the set of rules and practises that  
2683 regulates how the TSF manages, protects, and otherwise controls the system resources. The model  
2684 includes the set of restrictions and properties that specify how information and computing resources  
2685 are prevented from being used to violate the SFRs, accompanied by a persuasive set of engineering  
2686 arguments showing that these restrictions and properties play a key role in the enforcement of the  
2687 SFRs. It consists both of the formalisms that express the security functionality, as well as ancillary text  
2688 to explain the model and to provide it with context. The security behaviour of the TSF is modelled both  
2689 in terms of external behaviour (i.e. how the TSF interacts with the rest of the TOE and with its  
2690 operational environment), as well as its internal behaviour.

2691 The security policy model of the TOE is informally abstracted from its realisation by considering the  
2692 TSFI behaviour defined in the functional specification, which is strongly connected to the SFRs and  
2693 security policies expressed in the ST. The purpose of formal methods lies within the enhancement of  
2694 the rigour of enforcement. Informal arguments are always prone to fallacies; especially if relationships  
2695 among subjects, objects and operations get more and more involved. In order to minimise the risk of  
2696 insecure state reachability the rules and characteristics of the security policy model are mapped to  
2697 respective properties and features within some formal system, whose rigour and strength can  
2698 afterwards be used to obtain the security properties by means of theorems and formal proof.

2699 While the term “formal security policy model” is used in academic circles, ISO/IEC 15408's approach  
2700 has no fixed definition of “security”; it would equate to whatever SFRs are being claimed. Therefore,  
2701 the formal security policy model is merely a formal representation of the set of SFRs being claimed by  
2702 the TOE.

2703 The term *security policy* has traditionally been associated with only access control policies, whether  
2704 label-based (mandatory access control) or user-based (discretionary access control). However, a  
2705 security policy is not limited to access control; there are also audit policies, identification policies,  
2706 authentication policies, encryption policies, management policies, and any other security policies that  
2707 are enforced by the TOE, as described in the PP/ST. **ADV\_SPM.1.1D** contains an assignment for  
2708 identifying the SFRs and security policies that are formally modelled through the corresponding TSFIs.

#### 2709 **10.6.4 ADV\_SPM.1 Formal TOE security policy model**

2710 Dependencies: ADV\_FSP.5 Complete semi-formal functional specification with additional error  
2711 information

2712 ADV\_FSP.6 Complete semi-formal functional specification with additional formal  
2713 specification

##### 2714 **10.6.4.1 Developer action elements**

###### 2715 **10.6.4.1.1 ADV\_SPM.1.1D**

2716 **The developer shall provide a formal security policy model for the TSFI behaviour of the TOE.**

###### 2717 **10.6.4.1.2 ADV\_SPM.1.2D**

2718 **The developer shall determine all TSFIs and analyze for each TSFI whether its behaviour can be**  
2719 **modelled by the formal security policy model. If a TSFI cannot be modelled, for example caused**  
2720 **by technical limitations, the developer shall analyze the impact of not modelling the TSFI**

- 2721 behaviour on the security of the TOE. If parts of the TSFI behaviour cannot be modelled due to  
2722 technical limitations, the remaining parts shall nevertheless be covered by the formal model.
- 2723 **10.6.4.1.3 ADV\_SPM.1.3D**
- 2724 The formal security policy model shall identify the modelled TSFIs. For each TSFI covered by  
2725 the formal security policy model, the model shall identify the related SFRs and security policies  
2726 in the ST. For each SFR covered by the formal security policy model, the model shall identify the  
2727 relevant portions of the statement of SFRs.
- 2728 **10.6.4.1.4 ADV\_SPM.1.4D**
- 2729 For all TSFIs that are not modelled by the formal security policy model, the developer shall  
2730 identify the affected SFRs and security policies in the ST.
- 2731 **10.6.4.1.5 ADV\_SPM.1.5D**
- 2732 The developer shall provide a formal proof of correspondence between the model and any  
2733 formal functional specification. The proof of correspondence shall relate model elements and  
2734 TSFIs. With ADV\_SPM.1.3D, the given proof of correspondence thereby implicitly provides a  
2735 correspondence between model elements and SFRs, as well as model elements and security  
2736 policies. The developer defines a structured process for identifying and presenting  
2737 corresponding items formally.
- 2738 **10.6.4.1.6 ADV\_SPM.1.6D**
- 2739 The developer shall provide a demonstration of correspondence between the model and the  
2740 functional specification. This item shall demonstrate the correspondence between model  
2741 elements and TSFIs.
- 2742 **10.6.4.2 Content and presentation elements**
- 2743 **10.6.4.2.1 ADV\_SPM.1.1C**
- 2744 The model shall define security for the TOE and provide a formal proof that the TOE cannot  
2745 reach a state that is not secure.
- 2746 **10.6.4.2.2 ADV\_SPM.1.2C**
- 2747 The developer shall provide an analysis why the chosen modelling formalism is appropriate.
- 2748 **10.6.4.2.3 ADV\_SPM.1.3C**
- 2749 If tool support is used, the developer shall identify the tool chain used to verify the formal  
2750 security policy model, including environments and version numbers. The developer shall  
2751 provide arguments why the tool chain is suited and trustworthy.
- 2752 **10.6.4.2.4 ADV\_SPM.1.4C**
- 2753 The developer shall define how the formal analysis of the formal security policy model can be  
2754 reproduced (for example, applying an interactive theorem prover to prove correctness of the  
2755 formal security policy model).

#### 2756 10.6.4.2.5 ADV\_SPM.1.5C

2757 The model shall be in a formal style, supported by explanatory text as required, and identify  
 2758 the TSFIs that are modelled. Additionally, the SFRs and security policies of the TSF that are  
 2759 modelled via the TSFI behaviour shall be presented. The model shall identify all TSFIs that are  
 2760 not modelled (compare ADV\_SPM.1.2D) and present the affected SFRs and security policies. The  
 2761 model shall explain the reason for not modelling TSFIs and provide an impact analysis which  
 2762 shows that correctness of the formal model is not affected.

#### 2763 10.6.4.2.6 ADV\_SPM.1.6C

2764 The correspondence between the model and the functional specification shall be at the correct  
 2765 level of formality. The developer shall describe the correspondence analysis process and define  
 2766 the applied understanding of correspondence. If a semi-formal functional specification is  
 2767 provided, the correspondence must be shown semi-formally. If a formal functional  
 2768 specification is provided, the correspondence must be shown formally.

#### 2769 10.6.4.2.7 ADV\_SPM.1.7C

2770 The correspondence shall show that the model is consistent and complete with respect to the  
 2771 functional specification.

### 2772 10.6.4.3 Evaluator action elements

#### 2773 10.6.4.3.1 ADV\_SPM.1.1E

2774 The evaluator shall confirm that the information provided meets all requirements for content  
 2775 and presentation of evidence.

## 2776 10.7 TOE design (ADV\_TDS)

### 2777 10.7.1 Objectives

2778 The design description of a TOE provides both context for a description of the TSF, and a thorough  
 2779 description of the TSF. As assurance needs increase, the level of detail provided in the description also  
 2780 increases. As the size and complexity of the TSF increase, multiple levels of decomposition are  
 2781 appropriate. The design requirements are intended to provide information (commensurate with the  
 2782 given assurance level) so that a determination can be made that the security functional requirements  
 2783 are realised.

### 2784 10.7.2 Component levelling

2785 The components in this family are levelled on the basis of the amount of information that is required  
 2786 to be presented with respect to the TSF, and on the degree of formalism required of the design  
 2787 description.

### 2788 10.7.3 Application notes

2789 The goal of design documentation is to provide sufficient information to determine the TSF boundary,  
 2790 and to describe *how* the TSF implements the Security Functional Requirements. The amount and  
 2791 structure of the design documentation will depend on the complexity of the TOE and the number of  
 2792 SFRs; in general, a very complex TOE with a large number of SFRs will require more design  
 2793 documentation than a very simple TOE implementing only a few SFRs. Very complex TOEs will benefit  
 2794 (in terms of the assurance provided) from the production of differing levels of decomposition in



2795 describing the design, while very simple TOEs do not require both high-level and low-level  
2796 descriptions of its implementation.

2797 This family uses two levels of decomposition: the *subsystem* and the *module*. A module is the most  
2798 specific description of functionality: it is a description of the implementation. A developer should be  
2799 able to implement the part of the TOE described by the module with no further design decisions. A  
2800 subsystem is a description of the design of the TOE; it helps to provide a high-level description of what  
2801 a portion of the TOE is doing and how. As such, a subsystem may be further divided into lower-level  
2802 subsystems, or into modules. Very complex TOEs might require several levels of subsystems in order  
2803 to adequately convey a useful description of how the TOE works. Very simple TOEs, in contrast, might  
2804 not require a subsystem level of description; the module might clearly describe how the TOE works.

2805 The general approach adopted for design documentation is that, as the level of assurance increases,  
2806 the emphasis of description shifts from the general (subsystem level) to more (module level) detail. In  
2807 cases where a module-level of abstraction is appropriate because the TOE is simple enough to be  
2808 described at the module level, yet the level of assurance calls for a subsystem level of description, the  
2809 module-level description alone will suffice. For complex TOEs, however, this is not the case: an  
2810 enormous amount of (module-level) detail would be incomprehensible without an accompanying  
2811 subsystem level of description.

2812 This approach follows the general paradigm that providing additional detail about the implementation  
2813 of the TSF will result in greater assurance that the SFRs are implemented correctly, and provide  
2814 information that can be used to demonstrate this in testing (ATE: Tests).

2815 In the requirements for this family, the term *interface* is used as the means of communication  
2816 (between two subsystems or modules). It describes how the communication is invoked; this is similar  
2817 to the details of TSFI (see Functional specification (ADV\_FSP)). The term *interaction* is used to identify  
2818 the purpose for communication; it identifies why two subsystems or modules are communicating.

### 2819 10.7.3.1 Detail about the Subsystems and Modules

2820 The requirements define collections of details about subsystems and modules to be provided:

- 2821 a) The subsystems and modules are *identified* with a simple list of what they are.
- 2822 b) Subsystems and modules may be *categorised* (either implicitly or explicitly) as “SFR-enforcing”,  
2823 “SFR-supporting”, or “SFR-non-interfering”; these terms are used the same as they are used in  
2824 Functional specification (ADV\_FSP).
- 2825 c) A subsystem's *behaviour* is what it does. The behaviour may also be categorised as SFR-enforcing,  
2826 SFR-supporting, or SFR-non-interfering. The behaviour of the subsystem is never categorised as  
2827 more SFR-relevant than the category of the subsystem itself. For example, an SFR-enforcing  
2828 subsystem can have SFR-enforcing behaviour as well as SFR-supporting or SFR-non-interfering  
2829 behaviour.
- 2830 d) A *behaviour summary* of a subsystem is an overview of the actions it performs (e.g. “The TCP  
2831 subsystem assembles IP datagrams into reliable byte streams”).
- 2832 e) A *behaviour description* of a subsystem is an explanation of everything it does. This description  
2833 should be at a level of detail that one can readily determine whether the behaviour has any  
2834 relevance to the enforcement of the SFRs.
- 2835 f) A *description of interactions* among or between subsystems or modules identifies the reason that  
2836 subsystems or modules communicate, and characterises the information that is passed. It need not

- 2837 define the information to the same level of detail as an interface specification. For example, it  
 2838 would be sufficient to say “subsystem X requests a block of memory from the memory manager,  
 2839 which responds with the location of the allocated memory.
- 2840 g) A *description of interfaces* provides the details of how the interactions among modules are  
 2841 achieved. Rather than describing the reason the modules are communicating or the purpose of  
 2842 their communication (that is, the description of interactions), the description of interfaces  
 2843 describes the details of how that communication is accomplished, in terms of the structure and  
 2844 contents of the messages, semaphores, internal process communications, etc.
- 2845 h) The *purpose* describes how a module provides their functionality. It provides sufficient detail that  
 2846 no further design decisions are needed. The correspondence between the implementation  
 2847 representation that implements the module, and the purpose of the module should be readily  
 2848 apparent.
- 2849 i) A module is otherwise *described* in terms of whatever is identified in the element.
- 2850 Subsystems and modules, and “SFR-enforcing”, etc. are all further explained in greater detail in A.4,  
 2851 ADV\_TDS: Subsystems and Modules.
- 2852 **10.7.4 ADV\_TDS.1 Basic design**
- 2853 Dependencies: ADV\_FSP.2 Security-enforcing functional specification
- 2854 **10.7.4.1 Developer action elements**
- 2855 **10.7.4.1.1 ADV\_TDS.1.1D**
- 2856 **The developer shall provide the design of the TOE.**
- 2857 **10.7.4.1.2 ADV\_TDS.1.2D**
- 2858 **The developer shall provide a mapping from the TSFI of the functional specification to the**  
 2859 **lowest level of decomposition available in the TOE design.**
- 2860 **10.7.4.2 Content and presentation elements**
- 2861 **10.7.4.2.1 ADV\_TDS.1.1C**
- 2862 **The design shall describe the structure of the TOE in terms of subsystems.**
- 2863 **10.7.4.2.2 ADV\_TDS.1.2C**
- 2864 **The design shall identify all subsystems of the TSF.**
- 2865 **10.7.4.2.3 ADV\_TDS.1.3C**
- 2866 **The design shall provide the behaviour summary of each SFR-supporting or SFR-non-**  
 2867 **interfering TSF subsystem.**
- 2868 **10.7.4.2.4 ADV\_TDS.1.4C**
- 2869 **The design shall summarise the SFR-enforcing behaviour of the SFR-enforcing subsystems.**

2870 **10.7.4.2.5 ADV\_TDS.1.5C**

2871 **The design shall provide a description of the interactions among SFR-enforcing subsystems of**  
 2872 **the TSF, and between the SFR-enforcing subsystems of the TSF and other subsystems of the TSF.**

2873 **10.7.4.2.6 ADV\_TDS.1.6C**

2874 **The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE**  
 2875 **design that they invoke.**

2876 **10.7.4.3 Evaluator action elements**

2877 **10.7.4.3.1 ADV\_TDS.1.1E**

2878 **The evaluator shall confirm that the information provided meets all requirements for content**  
 2879 **and presentation of evidence.**

2880 **10.7.4.3.2 ADV\_TDS.1.2E**

2881 **The evaluator shall determine that the design is an accurate and complete instantiation of all**  
 2882 **security functional requirements.**

2883 **10.7.5 ADV\_TDS.2 Architectural design**

2884 Dependencies: ADV\_FSP.3 Functional specification with complete summary

2885 **10.7.5.1 Developer action elements**

2886 **10.7.5.1.1 ADV\_TDS.2.1D**

2887 The developer shall provide the design of the TOE.

2888 **10.7.5.1.2 ADV\_TDS.2.2D**

2889 The developer shall provide a mapping from the TSFI of the functional specification to the lowest level  
 2890 of decomposition available in the TOE design.

2891 **10.7.5.2 Content and presentation elements**

2892 **10.7.5.2.1 ADV\_TDS.2.1C**

2893 The design shall describe the structure of the TOE in terms of subsystems.

2894 **10.7.5.2.2 ADV\_TDS.2.2C**

2895 The design shall identify all subsystems of the TSF.

2896 **10.7.5.2.3 ADV\_TDS.2.3C**

2897 **The design shall provide the behaviour summery of each SFR non-interfering subsystem of the**  
 2898 **TSF.**

2899 **10.7.5.2.4 ADV\_TDS.2.4C**

2900 The design shall **describe** the SFR-enforcing behaviour of the SFR-enforcing subsystems.

2901 **10.7.5.2.5 ADV\_TDS.2.5C**

2902 The design shall summarise the **SFR-supporting and SFR-non-interfering** behaviour of the SFR-  
2903 enforcing subsystems.

2904 **10.7.5.2.6 ADV\_TDS.2.6C**

2905 The design shall summarise the behaviour of the **SFR-supporting** subsystems.

2906 **10.7.5.2.7 ADV\_TDS.2.7C**

2907 **The design shall provide a description of the interactions among all subsystems of the TSF.**

2908 **10.7.5.2.8 ADV\_TDS.2.8C**

2909 The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE design that  
2910 they invoke.

2911 **10.7.5.3 Evaluator action elements**

2912 **10.7.5.3.1 ADV\_TDS.2.1E**

2913 The evaluator shall confirm that the information provided meets all requirements for content and  
2914 presentation of evidence.

2915 **10.7.5.3.2 ADV\_TDS.2.2E**

2916 The evaluator shall determine that the design is an accurate and complete instantiation of all security  
2917 functional requirements.

2918 **10.7.6 ADV\_TDS.3 Basic modular design**

2919 Dependencies: ADV\_FSP.4 Complete functional specification

2920 **10.7.6.1 Developer action elements**

2921 **10.7.6.1.1 ADV\_TDS.3.1D**

2922 The developer shall provide the design of the TOE.

2923 **10.7.6.1.2 ADV\_TDS.3.2D**

2924 The developer shall provide a mapping from the TSFI of the functional specification to the lowest level  
2925 of decomposition available in the TOE design.

2926 **10.7.6.2 Content and presentation elements**

2927 **10.7.6.2.1 ADV\_TDS.3.1C**

2928 The design shall describe the structure of the TOE in terms of subsystems.

2929 **10.7.6.2.2 ADV\_TDS.3.2C**

2930 **The design shall describe the TSF in terms of modules.**

2931 **10.7.6.2.3 ADV\_TDS.3.3C**

2932 The design shall identify all subsystems of the TSF.

2933 **10.7.6.2.4 ADV\_TDS.3.4C**

2934 The design shall **provide a description of each subsystem of the TSF.**

2935 **10.7.6.2.5 ADV\_TDS.3.5C**

2936 The design shall provide a description of the interactions among all subsystems of the TSF.

2937 **10.7.6.2.6 ADV\_TDS.3.6C**

2938 **The design shall provide a mapping from the subsystems of the TSF to the modules of the TSF.**

2939 **10.7.6.2.7 ADV\_TDS.3.7C**

2940 The design shall describe each **SFR-enforcing module in terms of its purpose and relationship**  
2941 **with other modules.**

2942 **10.7.6.2.8 ADV\_TDS.3.8C**

2943 **The design shall describe each SFR-enforcing module in terms of its SFR-related interfaces,**  
2944 **return values from those interfaces, interaction with other modules and called SFR-related**  
2945 **interfaces to other SFR-enforcing modules.**

2946 **10.7.6.2.9 ADV\_TDS.3.9C**

2947 The design shall describe each **SFR-supporting and SFR-non-interfering module in terms of its**  
2948 **purpose and interaction with other modules.**

2949 **10.7.6.2.10 ADV\_TDS.3.10C**

2950 The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE design that  
2951 they invoke.

2952 **10.7.6.3 Evaluator action elements**

2953 **10.7.6.3.1 ADV\_TDS.3.1E**

2954 The evaluator shall confirm that the information provided meets all requirements for content and  
2955 presentation of evidence.

2956 **10.7.6.3.2 ADV\_TDS.3.2E**

2957 The evaluator shall determine that the design is an accurate and complete instantiation of all security  
2958 functional requirements.

2959 **10.7.7 ADV\_TDS.4 Semiformal modular design**

2960 Dependencies: ADV\_FSP.5 Complete semi-formal functional specification with additional error  
2961 information

2962 **10.7.7.1 Developer action elements**

2963 **10.7.7.1.1 ADV\_TDS.4.1D**

2964 The developer shall provide the design of the TOE.

2965 **10.7.7.1.2 ADV\_TDS.4.2D**

2966 The developer shall provide a mapping from the TSFI of the functional specification to the lowest level  
2967 of decomposition available in the TOE design.

2968 **10.7.7.2 Content and presentation elements**

2969 **10.7.7.2.1 ADV\_TDS.4.1C**

2970 The design shall describe the structure of the TOE in terms of subsystems.

2971 **10.7.7.2.2 ADV\_TDS.4.2C**

2972 The design shall describe the TSF in terms of modules, **designating each module as SFR-enforcing,**  
2973 **SFR-supporting, or SFR-non-interfering.**

2974 **10.7.7.2.3 ADV\_TDS.4.3C**

2975 The design shall identify all subsystems of the TSF.

2976 **10.7.7.2.4 ADV\_TDS.4.4C**

2977 The design shall provide a **semiformal** description of each subsystem of the TSF, **supported by**  
2978 **informal, explanatory text where appropriate.**

2979 **10.7.7.2.5 ADV\_TDS.4.5C**

2980 The design shall provide a description of the interactions among all subsystems of the TSF.

2981 **10.7.7.2.6 ADV\_TDS.4.6C**

2982 The design shall provide a mapping from the subsystems of the TSF to the modules of the TSF.

2983 **10.7.7.2.7 ADV\_TDS.4.7C**

2984 The design shall describe each SFR-enforcing **and SFR-supporting** module in terms of its purpose and  
2985 relationship with other modules.

2986 **10.7.7.2.8 ADV\_TDS.4.8C**

2987 The design shall describe each SFR-enforcing **and SFR-supporting** module in terms of its SFR-related  
2988 interfaces, return values from those interfaces, interaction with other modules and called SFR-related  
2989 interfaces to other SFR-enforcing **or SFR-supporting** modules.

2990 **10.7.7.2.9 ADV\_TDS.4.9C**

2991 The design shall describe each SFR-non-interfering module in terms of its purpose and interaction  
2992 with other modules.

2993 **10.7.7.2.10 ADV\_TDS.4.10C**

2994 The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE design that  
2995 they invoke.

2996 **10.7.7.3 Evaluator action elements**

2997 **10.7.7.3.1 ADV\_TDS.4.1E**

2998 The evaluator shall confirm that the information provided meets all requirements for content and  
2999 presentation of evidence.

3000 **10.7.7.3.2 ADV\_TDS.4.2E**

3001 The evaluator shall determine that the design is an accurate and complete instantiation of all security  
3002 functional requirements.

3003 **10.7.8 ADV\_TDS.5 Complete semiformal modular design**

3004 Dependencies: ADV\_FSP.5 Complete semi-formal functional specification with additional error  
3005 information

3006 **10.7.8.1 Developer action elements**

3007 **10.7.8.1.1 ADV\_TDS.5.1D**

3008 The developer shall provide the design of the TOE.

3009 **10.7.8.1.2 ADV\_TDS.5.2D**

3010 The developer shall provide a mapping from the TSFI of the functional specification to the lowest level  
3011 of decomposition available in the TOE design.

3012 **10.7.8.2 Content and presentation elements**

3013 **10.7.8.2.1 ADV\_TDS.5.1C**

3014 The design shall describe the structure of the TOE in terms of subsystems.

3015 **10.7.8.2.2 ADV\_TDS.5.2C**

3016 The design shall describe the TSF in terms of modules, designating each module as SFR-enforcing, SFR-  
3017 supporting, or SFR-non-interfering.

3018 **10.7.8.2.3 ADV\_TDS.5.3C**

3019 The design shall identify all subsystems of the TSF.

3020 **10.7.8.2.4 ADV\_TDS.5.4C**

3021 The design shall provide a semiformal description of each subsystem of the TSF, supported by  
3022 informal, explanatory text where appropriate.

3023 **10.7.8.2.5 ADV\_TDS.5.5C**

3024 The design shall provide a description of the interactions among all subsystems of the TSF.

3025 **10.7.8.2.6 ADV\_TDS.5.6C**

3026 The design shall provide a mapping from the subsystems of the TSF to the modules of the TSF.

3027 **10.7.8.2.7 ADV\_TDS.5.7C**

3028 The design shall **provide a semiformal description of** each module in terms of its **purpose,**  
 3029 **interaction,** interfaces, return values from those interfaces, and called interfaces to other modules,  
 3030 **supported by informal, explanatory text where appropriate.**

3031 **10.7.8.2.8 ADV\_TDS.5.8C**

3032 The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE design that  
 3033 they invoke.

3034 **10.7.8.3 Evaluator action elements**

3035 **10.7.8.3.1 ADV\_TDS.5.1E**

3036 The evaluator shall confirm that the information provided meets all requirements for content and  
 3037 presentation of evidence.

3038 **10.7.8.3.2 ADV\_TDS.5.2E**

3039 The evaluator shall determine that the design is an accurate and complete instantiation of all security  
 3040 functional requirements.

3041 **10.7.9 ADV\_TDS.6 Complete semiformal modular design with formal high-level design**  
 3042 **presentation**

3043 Dependencies: ADV\_FSP.6 Complete semi-formal functional specification with additional formal  
 3044 specification

3045 **10.7.9.1 Developer action elements**

3046 **10.7.9.1.1 ADV\_TDS.6.1D**

3047 The developer shall provide the design of the TOE.

3048 **10.7.9.1.2 ADV\_TDS.6.2D**

3049 The developer shall provide a mapping from the TSFI of the functional specification to the lowest level  
 3050 of decomposition available in the TOE design.

3051 **10.7.9.1.3 ADV\_TDS.6.3D**

3052 **The developer shall provide a formal specification of the TSF subsystems.**

3053 **10.7.9.1.4 ADV\_TDS.6.4D**

3054 **The developer shall provide a proof of correspondence between the formal specifications of the**  
 3055 **TSF subsystems and of the functional specification.**



- 3056 **10.7.9.2 Content and presentation elements**
- 3057 **10.7.9.2.1 ADV\_TDS.6.1C**
- 3058 The design shall describe the structure of the TOE in terms of subsystems.
- 3059 **10.7.9.2.2 ADV\_TDS.6.2C**
- 3060 The design shall describe the TSF in terms of modules, designating each module as SFR-enforcing, SFR-  
3061 supporting, or SFR-non-interfering.
- 3062 **10.7.9.2.3 ADV\_TDS.6.3C**
- 3063 The design shall identify all subsystems of the TSF.
- 3064 **10.7.9.2.4 ADV\_TDS.6.4C**
- 3065 The design shall provide a semiformal description of each subsystem of the TSF, supported by  
3066 informal, explanatory text where appropriate.
- 3067 **10.7.9.2.5 ADV\_TDS.6.5C**
- 3068 The design shall provide a description of the interactions among all subsystems of the TSF.
- 3069 **10.7.9.2.6 ADV\_TDS.6.6C**
- 3070 The design shall provide a mapping from the subsystems of the TSF to the modules of the TSF.
- 3071 **10.7.9.2.7 ADV\_TDS.6.7C**
- 3072 The design shall **describe** each module in **semiformal style** in terms of its purpose, interaction,  
3073 interfaces, return values from those interfaces, and called interfaces to other modules, supported by  
3074 informal, explanatory text where appropriate.
- 3075 **10.7.9.2.8 ADV\_TDS.6.8C**
- 3076 **The formal specification of the TSF subsystems shall describe the TSF using a formal style,**  
3077 **supported by informal, explanatory text where appropriate.**
- 3078 **10.7.9.2.9 ADV\_TDS.6.9C**
- 3079 The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE design that  
3080 they invoke.
- 3081 **10.7.9.2.10 ADV\_TDS.6.10C**
- 3082 **The proof of correspondence between the formal specifications of the TSF subsystems and of**  
3083 **the functional specification shall demonstrate that all behaviour described in the TOE design is**  
3084 **a correct and complete refinement of the TSFI that invoked it.**
- 3085 **10.7.9.3 Evaluator action elements**
- 3086 **10.7.9.3.1 ADV\_TDS.6.1E**
- 3087 The evaluator shall confirm that the information provided meets all requirements for content and  
3088 presentation of evidence.

### 3089 **10.7.9.3.2 ADV\_TDS.6.2E**

3090 The evaluator shall determine that the design is an accurate and complete instantiation of all security  
3091 functional requirements.

## 3092 **10.8 Composite design compliance (ADV\_COMP)**

### 3093 **10.8.1 Objectives**

3094 The aim of this activity is to determine whether the requirements on the application, imposed by the  
3095 underlying platform, are fulfilled in the composite product.

### 3096 **10.8.2 Component levelling**

3097 This family contains only one component.

### 3098 **10.8.3 Application notes**

3099 The requirements on the application, imposed by the underlying platform, can be formulated in the  
3100 relevant certification report (e.g. in form of constraints and recommendations), user guidance and  
3101 ETR\_COMP (in form of observations and recommendations) for the platform. The developer of the  
3102 composite product shall regard each of these sources, if available (cf. Table D2, chapter D.1.7), and  
3103 implement the composite product in such a way that the applicable requirements are fulfilled.

3104 The TSF of the composite product is represented at various levels of abstraction in the families of the  
3105 development class ADV. Experiential, the appropriate levels of design representation for examining,  
3106 whether the requirements of the platform are fulfilled by the composite product, are the TOE design  
3107 (ADV\_TDS), security architecture (ADV\_ARC) and the implementation (ADV\_IMP). In case, these design  
3108 representation levels are not available (e.g. due to the assurance package chosen is EAL1), the current  
3109 activity is not applicable (see the next paragraph for the reason).

3110 Due to the definition of the composite TOE (cf. ISO/IEC 15408-1) the interface between the underlying  
3111 platform and the application is the internal one, hence, a functional specification (ADV\_FSP) as  
3112 representation level is not appropriate for analysing the design compliance.

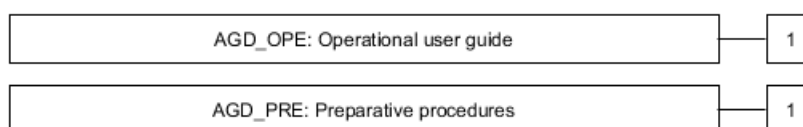
3113 Security architecture ADV\_ARC as assurance family is dedicated to ensure that integrative security  
3114 services like domain separation, self-protection and non-bypassability properly work. It is impossible  
3115 and not the sense of the composite evaluation to have an insight into the architectural internals of the  
3116 underlying platform (it is a matter of the platform evaluation). What the Composite Evaluator has to  
3117 do in the context of ADV\_ARC is

3118 i. to determine whether the application uses services of the underlying platform within its own  
3119 Composite-ST to provide domain separation, self-protection, non-bypassability and protected  
3120 start-up; if no, there is no further composite activities for ADV\_ARC; if yes, then

3121 ii. the evaluator has to determine, whether the application uses these platform-services in an  
3122 appropriate/secure way (please refer to the platform user guidance, cf. item #3 in Table D1,  
3123 chapter D.1.7).

3124 Since consistency of the composite product security policy has already been considered in the context  
3125 of the Security Target in the assurance family ASE\_COMP, there is no necessity to consider non-  
3126 contradictoriness of the security policy model (ADV\_SPM) of the composite TOE and the security  
3127 policy model of the underlying platform.

- 3128 **10.8.4 ADV\_COMP.1 Design compliance with the platform certification report, guidance and**  
 3129 **ETR\_COMP**
- 3130 Dependencies: No dependencies
- 3131 **10.8.4.1**
- 3132 **10.8.4.2 Developer action elements**
- 3133 **10.8.4.2.1 ADV\_COMP.1.1D**
- 3134 **The developer shall provide a design compliance justification; cf. item #6 as well as items #3,**  
 3135 **#4, #5 in Table D1, chapter D.1.7.**
- 3136 **10.8.4.3 Content and presentation elements**
- 3137 **10.8.4.3.1 ADV\_COMP.1.1C**
- 3138 **The design compliance justification shall provide a rationale for design compliance – on an**  
 3139 **appropriate representation level – of how the requirements on the application, imposed by the**  
 3140 **underlying platform, are fulfilled in the composite product.**
- 3141 **10.8.4.4 Evaluator action elements**
- 3142 **10.8.4.4.1 ADV\_COMP.1.1E**
- 3143 **The evaluator shall confirm that the rationale for design compliance is complete, coherent, and**  
 3144 **internally consistent.**
- 3145 **11 Class AGD: Guidance documents**
- 3146 **11.1 Introduction**
- 3147 The guidance documents class provides the requirements for guidance documentation for all user  
 3148 roles. For the secure preparation and operation of the TOE it is necessary to describe all relevant  
 3149 aspects for the secure handling of the TOE. The class also addresses the possibility of unintended  
 3150 incorrect configuration or handling of the TOE.
- 3151 In many cases it may be appropriate that guidance is provided in separate documents for preparation  
 3152 and operation of the TOE, or even separate for different user roles as end-users, administrators,  
 3153 application programmers using software or hardware interfaces, etc.
- 3154 The guidance documents class is subdivided into two families which are concerned with the  
 3155 preparative user guidance (what has to be done to transform the delivered TOE into its evaluated  
 3156 configuration in the operational environment as described in the ST) and with the operational user  
 3157 guidance (what has to be done during the operation of the TOE in its evaluated configuration).
- 3158 Figure 10 shows the families within this class, and the hierarchy of components within the families.



**Figure 10 — AGD: Guidance documents class decomposition**

## **11.2 Operational user guidance (AGD\_OPE)**

### **11.2.1 Objectives**

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

The operational user guidance provides a measure of confidence that non-malicious users, administrators, application providers and others exercising the external interfaces of the TOE will understand the secure operation of the TOE and will use it as intended. The evaluation of the user guidance includes investigating whether the TOE can be used in a manner that is insecure but that the user of the TOE would reasonably believe to be secure. The objective is to minimise the risk of human or other errors in operation that may deactivate, disable, or fail to activate security functionality, 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

- 3196 need not always be the case (e.g. the audit log fills up, an intrusion is detected). Furthermore, a  
 3197 security-relevant event may happen as a result of a specific chain of functions or, conversely, several  
 3198 security-relevant events may be triggered by one function.
- 3199 AGD\_OPE.1.7C requires that the user guidance is clear and reasonable. Misleading or unreasonable  
 3200 guidance may result in a user of the TOE believing that the TOE is secure when it is not.
- 3201 An example of misleading guidance would be the description of a single guidance instruction that  
 3202 could be parsed in more than one way, one of which may result in an insecure state.
- 3203 An example of unreasonable guidance would be a recommendation to follow a procedure that is so  
 3204 complicated that it cannot reasonably be expected that users will follow this guidance.
- 3205 **11.2.4 AGD\_OPE.1 Operational user guidance**
- 3206 Dependencies: ADV\_FSP.1 Basic functional specification
- 3207 **11.2.4.1 Developer action elements**
- 3208 **11.2.4.1.1 AGD\_OPE.1.1D**
- 3209 **The developer shall provide operational user guidance.**
- 3210 **11.2.4.2 Content and presentation elements**
- 3211 **11.2.4.2.1 AGD\_OPE.1.1C**
- 3212 **The operational user guidance shall describe, for each user role, the user-accessible functions**  
 3213 **and privileges that should be controlled in a secure processing environment, including**  
 3214 **appropriate warnings.**
- 3215 **11.2.4.2.2 AGD\_OPE.1.2C**
- 3216 **The operational user guidance shall describe, for each user role, how to use the available**  
 3217 **interfaces provided by the TOE in a secure manner.**
- 3218 **11.2.4.2.3 AGD\_OPE.1.3C**
- 3219 **The operational user guidance shall describe, for each user role, the available functions and**  
 3220 **interfaces, in particular all security parameters under the control of the user, indicating secure**  
 3221 **values as appropriate.**
- 3222 **11.2.4.2.4 AGD\_OPE.1.4C**
- 3223 **The operational user guidance shall, for each user role, clearly present each type of security-**  
 3224 **relevant event relative to the user-accessible functions that need to be performed, including**  
 3225 **changing the security characteristics of entities under the control of the TSF.**
- 3226 **11.2.4.2.5 AGD\_OPE.1.5C**
- 3227 **The operational user guidance shall identify all possible modes of operation of the TOE**  
 3228 **(including operation following failure or operational error), their consequences and**  
 3229 **implications for maintaining secure operation.**

3230 **11.2.4.2.6 AGD\_OPE.1.6C**

3231 **The operational user guidance shall, for each user role, describe the security controls to be**  
 3232 **followed in order to fulfil the security objectives for the operational environment as described**  
 3233 **in the ST.**

3234 **11.2.4.2.7 AGD\_OPE.1.7C**

3235 **The operational user guidance shall be clear and reasonable.**

3236 **11.2.4.3 Evaluator action elements**

3237 **11.2.4.3.1 AGD\_OPE.1.1E**

3238 **The evaluator shall confirm that the information provided meets all requirements for content**  
 3239 **and presentation of evidence.**

3240 **11.3 Preparative procedures (AGD\_PRE)**

3241 **11.3.1 Objectives**

3242 Preparative procedures are useful for ensuring that the TOE has been received and installed in a  
 3243 secure manner as intended by the developer. The requirements for preparation call for a secure  
 3244 transition from the delivered TOE to its initial operational environment. This includes investigating  
 3245 whether the TOE can be configured or installed in a manner that is insecure but that the user of the  
 3246 TOE would reasonably believe to be secure.

3247 **11.3.2 Component levelling**

3248 This family contains only one component.

3249 **11.3.3 Application notes**

3250 It is recognised that the application of these requirements will vary depending on aspects such as  
 3251 whether the TOE is delivered in an operational state, or whether it has to be installed at the TOE  
 3252 owner's site, etc.

3253 The first process covered by the preparative procedures is the consumer's secure acceptance of the  
 3254 received TOE in accordance with the developer's delivery procedures. If the developer has not defined  
 3255 delivery procedures, security of the acceptance has to be ensured otherwise.

3256 Installation of the TOE includes transforming its operational environment into a state that conforms to  
 3257 the security objectives for the operational environment provided in the ST.

3258 It might also be the case that no installation is necessary, for example a smart card. In this case it may  
 3259 be inappropriate to require and analyse installation procedures.

3260 The requirements in this assurance family are presented separately from those in the Operational user  
 3261 guidance (AGD\_OPE) family, due to the infrequent, possibly one-time use of the preparative  
 3262 procedures.

3263 **11.3.4 AGD\_PRE.1 Preparative procedures**

3264 Dependencies: No dependencies.

### 3265 11.3.4.1 Developer action elements

#### 3266 11.3.4.1.1 AGD\_PRE.1.1D

3267 The developer shall provide the TOE including its preparative procedures.

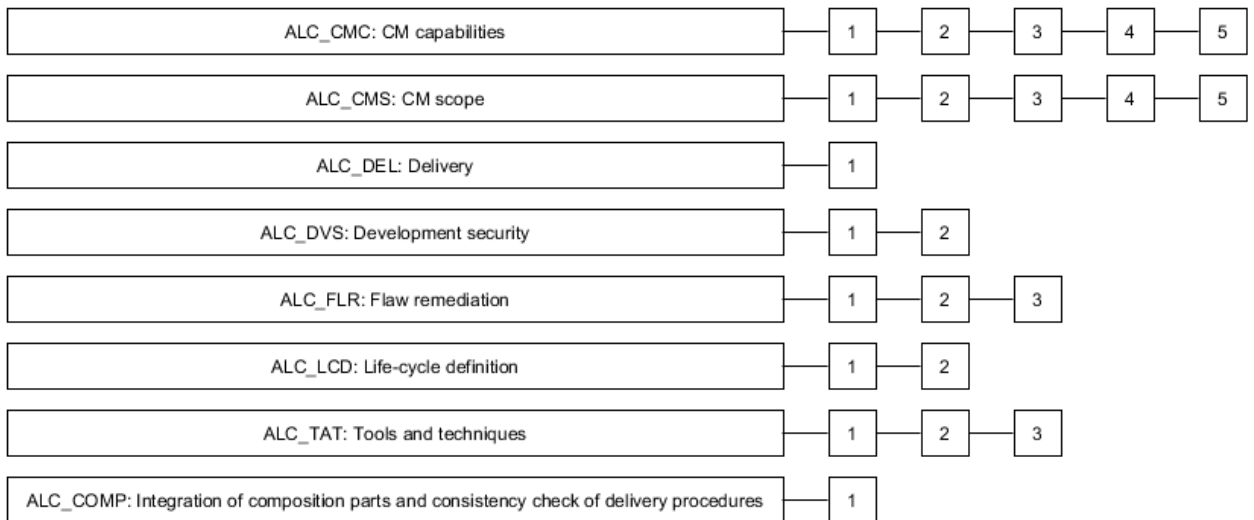
### 3268 11.3.4.2 Content and presentation elements

#### 3269 11.3.4.2.1 AGD\_PRE.1.1C

3270 The preparative procedures shall describe all the steps necessary for secure acceptance of the  
3271 delivered TOE in accordance with the developer's delivery procedures.

#### 3272 11.3.4.2.2 AGD\_PRE.1.2C

3273 The preparative procedures shall describe all the steps necessary for secure installation of the  
3274 TOE and for the secure preparation of the operational environment in accordance with the  
3275 secur



## 3283 12 C

### 3284 12.1

#### 3285 Introduction

3286 Life-cycle support is an aspect of establishing appropriate security controls in the development,  
3287 production, delivery and maintenance of the TOE. Confidence in the correspondence between the TOE  
3288 security requirements and the TOE is greater if security analysis and the production of the evidence  
3289 are done on a regular basis as an integral part of the development, production, delivery and  
3290 maintenance activities.

3291 During the life-cycle of the TOE it is distinguished whether the TOE is under the responsibility of the  
3292 TOE developer or the user rather than whether it is located in the development or the user  
3293 environment. The point of transition is when the TOE is accepted by the user. User in this context  
3294 relates to the end-user as well as product- and system integrators.

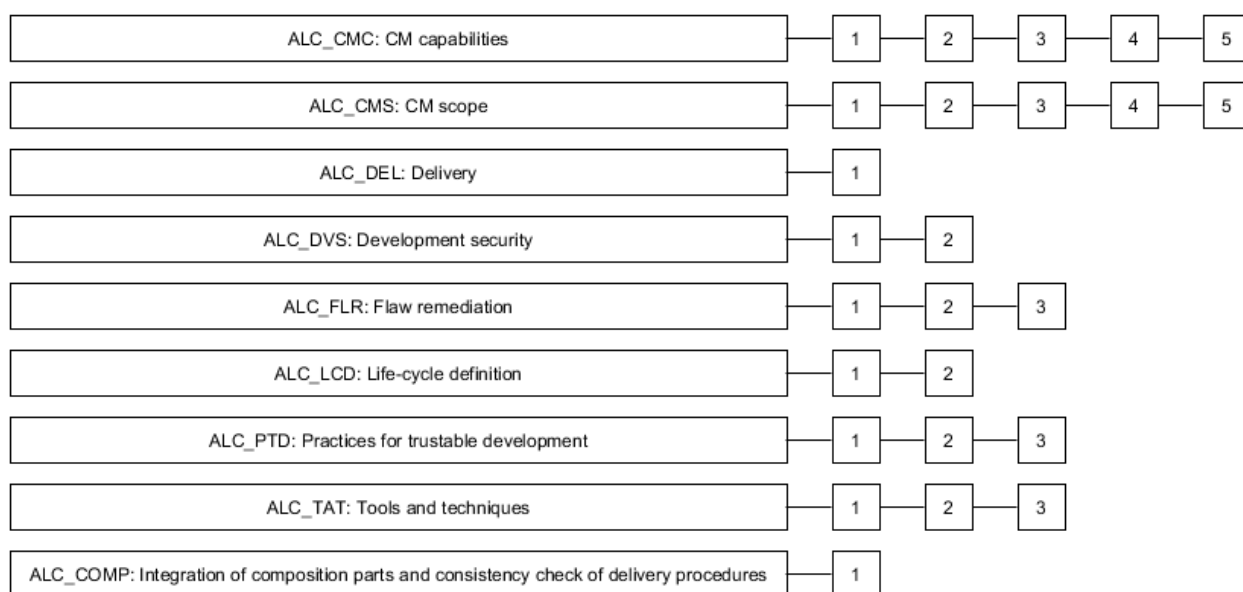
3295 The ALC class consists of seven families:

- 3296 • Development Life-cycle definition (ALC\_LCD) provides requirements for the developer's  
3297 description of the life-cycle model used in the development, production, delivery and maintenance  
3298 life-cycle of the TOE;
- 3299 • CM capabilities (ALC\_CMC) provides requirements for the management of the configuration items;

- 3300 • CM scope (ALC\_CMS) requires a minimum set of configuration items to be managed in the defined  
3301 way;
- 3302 • Developer environment security (ALC\_DVS) is concerned with the developer's physical, logical,  
3303 procedural, personnel, and other security controls;
- 3304 • Tools and techniques (ALC\_TAT) provides requirements for the development tools and  
3305 implementation standards used by the developer;
- 3306 • Flaw remediation (ALC\_FLR) provides requirements for the handling of security flaws.
- 3307 • Delivery (ALC\_DEL) provides requirements for the procedures used for the delivery of the TOE to  
3308 the downstream user. Delivery processes occurring during the development of the TOE are  
3309 denoted rather as transfers, and are handled in the context of integration and acceptance  
3310 procedures in other families of this class.

3311 Throughout this class, development and related terms (developer, develop) are meant in the more  
3312 general sense to comprise development and production, whereas production specifically means the  
3313 process of transforming the implementation representation into the final TOE.

3314 Figure 11 shows the families within this class, and the hierarchy of components within the families.



3315  
3316 **Figure 11 — ALC: Life-cycle support class decomposition**

## 3317 12.2 CM capabilities (ALC\_CMC)

### 3318 12.2.1 Objectives

3319 Configuration management (CM) techniques, properly defined as part of the development life-cycle  
3320 model, contribute to the assurance argument that the TOE meets the SFRs. A Configuration  
3321 Management (CM) system that is managed and operated correctly will help ensure the integrity of the  
3322 portions of the TOE that are controlled, by providing a method of tracking any changes to the TOE, and  
3323 to help ensure that all changes to the TOE are authorised.



3324 The objective of this family is to require the TOE developer's CM system to have certain capabilities.  
 3325 These capabilities are intended to reduce the likelihood that accidental or unauthorised modifications  
 3326 of the configuration items will occur. The CM system should support maintaining the integrity of the  
 3327 TOE throughout the part of the TOE's life-cycle that is under the control of the developer.

3328 The objective of introducing automated CM tools is to increase the effectiveness of the CM system.  
 3329 While both automated and manual CM systems can be bypassed, ignored, or proven insufficient to  
 3330 prevent unauthorised modification, automated systems are less susceptible to human error or  
 3331 negligence.

3332 The objectives of this family include the following:

- 3333 a) ensuring that the TOE is identifiable and complete before it is sent to the downstream user;
- 3334 b) ensuring that no configuration items are missed during evaluation;
- 3335 c) preventing unauthorised modification, addition, or deletion of TOE configuration items.

### 3336 **12.2.2 Component levelling**

3337 The components in this family are levelled on the basis of the CM system capabilities, the scope of the  
 3338 CM documentation and the evidence provided by the developer.

### 3339 **12.2.3 Application notes**

3340

3341 In the case where the TOE is a subset of a product, the requirements of this family apply only to the  
 3342 TOE configuration items, not to the product as a whole.

3343 For developer organizations that specify more than one CM application, or include different instances  
 3344 of a CM application within the scope of the TOEs design, development, production and maintenance, it  
 3345 is required to document all of them. For evaluation purposes, the set of CM applications should be  
 3346 regarded as parts of an overall CM system, applicable to the TOE, which is addressed in the criteria.

3347 The overall CM system should address any aspects of integration between component CM applications.

3348 Several elements of this family refer to configuration items. These elements identify CM requirements  
 3349 to be imposed on all items identified in the configuration list, but leave the contents of the list to the  
 3350 discretion of the developer. CM scope (ALC\_CMS) can be used to narrow this discretion by identifying  
 3351 specific items that must be included in the configuration list, and hence within the scope of the overall  
 3352 CM system.

3353 ALC\_CMC.2.3C introduces a requirement that the CM system uniquely identify all configuration items.  
 3354 This also requires that modifications to configuration items result in a new, unique identifier being  
 3355 assigned to the configuration item.

3356 ALC\_CMC.3.8C introduces the requirement that the evidence shall demonstrate that the CM system  
 3357 operates in accordance with the CM plan. Examples of such evidence might be documentation such as  
 3358 screen snapshots or audit trail output from the CM system, or a detailed demonstration of the CM  
 3359 system by the developer. The evaluator is responsible for determining that this evidence is sufficient  
 3360 to show that the CM system operates in accordance with the CM plan.

3361 ALC\_CMC.4.5C introduces a requirement that the CM system provide an automated means to support  
 3362 the production of the TOE. This requires that the CM system provide an automated means to assist in  
 3363 determining that the correct configuration items are used in generating the TOE.

3364 ALC\_CMC.5.10C introduces a requirement that the CM system provide an automated means to  
 3365 ascertain the changes between the TOE and its preceding version. If no previous version of the TOE  
 3366 exists, the developer still needs to provide an automated means to ascertain the changes between the  
 3367 TOE and a future version of the TOE.

#### 3368 **12.2.4 ALC\_CMC.1 Labelling of the TOE**

3369 Dependencies: ALC\_CMS.1 TOE CM coverage

##### 3370 **12.2.4.1 Objectives**

3371 A unique reference is required to ensure that there is no ambiguity in terms of which instance of the  
 3372 TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be  
 3373 aware of which instance of the TOE they are using.

##### 3374 **12.2.4.2 Developer action elements**

###### 3375 **12.2.4.2.1 ALC\_CMC.1.1D**

3376 **The developer shall provide the TOE and a unique reference for the TOE.**

##### 3377 **12.2.4.3 Content and presentation elements**

###### 3378 **12.2.4.3.1 ALC\_CMC.1.1C**

3379 **The TOE shall be labelled with its unique reference.**

##### 3380 **12.2.4.4 Evaluator action elements**

###### 3381 **12.2.4.4.1 ALC\_CMC.1.1E**

3382 **The evaluator shall confirm that the information provided meets all requirements for content**  
 3383 **and presentation of evidence.**

#### 3384 **12.2.5 ALC\_CMC.2 Use of the CM system**

3385 Dependencies: ALC\_CMS.1 TOE CM coverage

##### 3386 **12.2.5.1 Objectives**

3387 A unique reference is required to ensure that there is no ambiguity in terms of which instance of the  
 3388 TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be  
 3389 aware of which instance of the TOE they are using.

3390 Unique identification of the configuration items leads to a clearer understanding of the composition of  
 3391 the TOE, which in turn helps to determine those items which are subject to the evaluation  
 3392 requirements for the TOE.

3393 The use of a CM system increases assurance that the configuration items are maintained in a  
 3394 controlled manner.

3395 **12.2.5.2 Developer action elements**

3396 **12.2.5.2.1 ALC\_CMC.2.1D**

3397 The developer shall provide the TOE and a unique reference for the TOE.

3398 **12.2.5.2.2 ALC\_CMC.2.2D**

3399 **The developer shall provide the CM documentation.**

3400 **12.2.5.2.3 ALC\_CMC.2.3D**

3401 **The developer shall use a CM system.**

3402 **12.2.5.3 Content and presentation elements**

3403 **12.2.5.3.1 ALC\_CMC.2.1C**

3404 The TOE shall be labelled with its unique reference.

3405 **12.2.5.3.2 ALC\_CMC.2.2C**

3406 **The CM documentation shall describe the method used to uniquely identify the configuration**  
3407 **items.**

3408 **12.2.5.3.3 ALC\_CMC.2.3C**

3409 **The CM system shall uniquely identify all configuration items.**

3410 **12.2.5.4 Evaluator action elements**

3411 **12.2.5.4.1 ALC\_CMC.2.1E**

3412 The evaluator shall confirm that the information provided meets all requirements for content and  
3413 presentation of evidence.

3414 **12.2.6 ALC\_CMC.3 Authorisation controls**

3415 Dependencies: ALC\_CMS.1 TOE CM coverage

3416 ALC\_DVS.1 Identification of security measures

3417 ALC\_LCD.1 Developer defined life-cycle processes

3418 **12.2.6.1 Objectives**

3419 A unique reference is required to ensure that there is no ambiguity in terms of which instance of the  
3420 TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be  
3421 aware of which instance of the TOE they are using.

3422 Unique identification of the configuration items leads to a clearer understanding of the composition of  
3423 the TOE, which in turn helps to determine those items which are subject to the evaluation  
3424 requirements for the TOE.

3425 The use of a CM system increases assurance that the configuration items are maintained in a  
3426 controlled manner.

3427 Providing controls to ensure that unauthorised modifications are not made to the TOE (“CM access  
 3428 control”), and ensuring proper functionality and use of the CM system, helps to maintain the integrity  
 3429 of the TOE.

#### 3430 **12.2.6.2 Developer action elements**

##### 3431 **12.2.6.2.1 ALC\_CMC.3.1D**

3432 The developer shall provide the TOE and a unique reference for the TOE.

##### 3433 **12.2.6.2.2 ALC\_CMC.3.2D**

3434 The developer shall provide the CM documentation.

##### 3435 **12.2.6.2.3 ALC\_CMC.3.3D**

3436 The developer shall use a CM system.

#### 3437 **12.2.6.3 Content and presentation elements**

##### 3438 **12.2.6.3.1 ALC\_CMC.3.1C**

3439 The TOE shall be labelled with its unique reference.

##### 3440 **12.2.6.3.2 ALC\_CMC.3.2C**

3441 The CM documentation shall describe the method used to uniquely identify the configuration items.

##### 3442 **12.2.6.3.3 ALC\_CMC.3.3C**

3443 The CM system shall uniquely identify all configuration items.

##### 3444 **12.2.6.3.4 ALC\_CMC.3.4C**

3445 **The CM system shall provide controls such that only authorised changes are made to the**  
 3446 **configuration items.**

##### 3447 **12.2.6.3.5 ALC\_CMC.3.5C**

3448 **The CM documentation shall include a CM plan.**

##### 3449 **12.2.6.3.6 ALC\_CMC.3.6C**

3450 **The CM plan shall describe how the CM system is used for the development of the TOE.**

##### 3451 **12.2.6.3.7 ALC\_CMC.3.7C**

3452 **The evidence shall demonstrate that all configuration items are being maintained under the CM**  
 3453 **system.**

##### 3454 **12.2.6.3.8 ALC\_CMC.3.8C**

3455 **The evidence shall demonstrate that the CM system is being operated in accordance with the**  
 3456 **CM plan.**

3457 **12.2.6.4 Evaluator action elements**

3458 **12.2.6.4.1 ALC\_CMC.3.1E**

3459 The evaluator shall confirm that the information provided meets all requirements for content and  
3460 presentation of evidence.

3461 **12.2.7 ALC\_CMC.4 Production support, acceptance procedures and automation**

3462 Dependencies: ALC\_CMS.1 TOE CM coverage

3463 ALC\_DVS.1 Identification of security measures

3464 ALC\_LCD.1 Developer defined life-cycle processes

3465 **12.2.7.1 Objectives**

3466 A unique reference is required to ensure that there is no ambiguity in terms of which instance of the  
3467 TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be  
3468 aware of which instance of the TOE they are using.

3469 Unique identification of the configuration items leads to a clearer understanding of the composition of  
3470 the TOE, which in turn helps to determine those items which are subject to the evaluation  
3471 requirements for the TOE.

3472 The use of a CM system increases assurance that the configuration items are maintained in a  
3473 controlled manner.

3474 Providing access controls to help ensure that unauthorised modifications are not made to the TOE  
3475 ("CM access control"), and ensuring proper functionality and use of the CM system, helps to maintain  
3476 the integrity of the TOE.

3477 The purpose of the acceptance procedures is to ensure that the parts of the TOE are of adequate  
3478 quality and to confirm that any creation or modification of configuration items is authorised.  
3479 Acceptance procedures are an essential element in integration processes and in the life-cycle  
3480 management of the TOE.

3481 In a CM system where the quantity and organization of configuration items is complex, it is difficult to  
3482 control changes without the support of automated tools. In particular, these automated tools need to  
3483 be able to support the numerous changes that occur during development and ensure that those  
3484 changes are authorised. It is an objective of this component to ensure that the configuration items are  
3485 controlled through automated means. In the case where the overall CM system includes more than one  
3486 CM application then automated tools can also support integration between the CM applications and of  
3487 the TOE.

3488 Production support procedures help to ensure that the generation of the TOE from a managed set of  
3489 configuration items is correctly performed in an authorised manner, particularly in the case when  
3490 different developers are involved and integration processes have to be carried out.

3491 **12.2.7.2 Developer action elements**

3492 **12.2.7.2.1 ALC\_CMC.4.1D**

3493 The developer shall provide the TOE and a unique reference for the TOE.

- 3494 **12.2.7.2.2 ALC\_CMC.4.2D**
- 3495 The developer shall provide the CM documentation.
- 3496 **12.2.7.2.3 ALC\_CMC.4.3D**
- 3497 The developer shall use a CM system.
- 3498 **12.2.7.3 Content and presentation elements**
- 3499 **12.2.7.3.1 ALC\_CMC.4.1C**
- 3500 The TOE shall be labelled with its unique reference.
- 3501 **12.2.7.3.2 ALC\_CMC.4.2C**
- 3502 The CM documentation shall describe the method or methods used to uniquely identify the  
3503 configuration items.
- 3504 **12.2.7.3.3 ALC\_CMC.4.3C**
- 3505 The CM system shall uniquely identify all configuration items.
- 3506 **12.2.7.3.4 ALC\_CMC.4.4C**
- 3507 The CM system shall provide **automated** controls such that only authorised changes are made to the  
3508 configuration items.
- 3509 **12.2.7.3.5 ALC\_CMC.4.5C**
- 3510 **The CM system shall support the production of the TOE by automated means.**
- 3511 **12.2.7.3.6 ALC\_CMC.4.6C**
- 3512 The CM documentation shall include a CM plan.
- 3513 **12.2.7.3.7 ALC\_CMC.4.7C**
- 3514 The CM plan shall describe how the CM system is used for the development of the TOE.
- 3515 **12.2.7.3.8 ALC\_CMC.4.8C**
- 3516 **The CM plan shall describe the procedures used to accept modified or newly created**  
3517 **configuration items as part of the TOE.**
- 3518 **12.2.7.3.9 ALC\_CMC.4.9C**
- 3519 The evidence shall demonstrate that all configuration items are being maintained under the CM  
3520 system.
- 3521 **12.2.7.3.10 ALC\_CMC.4.10C**
- 3522 The evidence shall demonstrate that the CM system is being operated in accordance with the CM plan.

3523 **12.2.7.4 Evaluator action elements**

3524 **12.2.7.4.1 ALC\_CMC.4.1E**

3525 The evaluator shall confirm that the information provided meets all requirements for content and  
3526 presentation of evidence.

3527 **12.2.8 ALC\_CMC.5 Advanced support**

3528 Dependencies: ALC\_CMS.1 TOE CM coverage

3529 ALC\_DVS.2 Sufficiency of security measures

3530 ALC\_LCD.1 Developer defined life-cycle processes

3531 **12.2.8.1 Objectives**

3532 A unique reference is required to ensure that there is no ambiguity in terms of which instance of the  
3533 TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be  
3534 aware of which instance of the TOE they are using.

3535 Unique identification of the configuration items leads to a clearer understanding of the composition of  
3536 the TOE, which in turn helps to determine those items which are subject to the evaluation  
3537 requirements for the TOE.

3538 The use of a CM system increases assurance that the configuration items are maintained in a  
3539 controlled manner.

3540 Providing controls to ensure that unauthorised modifications are not made to the TOE ("CM access  
3541 control"), and ensuring proper functionality and use of the CM system, helps to maintain the integrity  
3542 of the TOE.

3543 The purpose of the acceptance procedures is to ensure that the parts of the TOE meet defined criteria  
3544 in regard to the integrity of the TOE. Acceptance criteria may include code review, checking for  
3545 vulnerabilities, authenticity checking, and functional testing to confirm that any creation or  
3546 modification of configuration items is authorised. Acceptance procedures are an essential element in  
3547 integration processes and in the life-cycle management of the TOE.

3548 In development environments where the configuration items are complex, it is difficult to control  
3549 changes without the support of automated tools. In particular, these automated tools need to be able  
3550 to support the numerous changes that occur during development and ensure that those changes are  
3551 authorised. It is an objective of this component to ensure that the configuration items are controlled  
3552 through automated means. If the TOE is developed by multiple developers, i.e. integration has to take  
3553 place, the use of automatic tools is adequate.

3554 Production support procedures help to ensure that the generation of the TOE from a managed set of  
3555 configuration items is correctly performed in an authorised manner, particularly in the case when  
3556 different developers are involved and integration processes have to be carried out.

3557 Requiring that the CM system be able to identify the version of the implementation representation  
3558 from which the TOE is generated helps to ensure that the integrity of this material is preserved by the  
3559 appropriate technical, physical and procedural safeguards.

3560 Providing an automated means of ascertaining changes between versions of the TOE and identifying  
 3561 which configuration items are affected by modifications to other configuration items assists in  
 3562 determining the impact of the changes between successive versions of the TOE. This in turn can  
 3563 provide valuable information in determining whether changes to the TOE result in all configuration  
 3564 items being consistent with one another.

## 3565 **12.2.8.2 Developer action elements**

### 3566 **12.2.8.2.1 ALC\_CMC.5.1D**

3567 The developer shall provide the TOE and a unique reference for the TOE.

### 3568 **12.2.8.2.2 ALC\_CMC.5.2D**

3569 The developer shall provide the CM documentation.

### 3570 **12.2.8.2.3 ALC\_CMC.5.3D**

3571 The developer shall use a CM system.

## 3572 **12.2.8.3 Content and presentation elements**

### 3573 **12.2.8.3.1 ALC\_CMC.5.1C**

3574 The TOE shall be labelled with its unique reference.

### 3575 **12.2.8.3.2 ALC\_CMC.5.2C**

3576 The CM documentation shall describe the method used to uniquely identify the configuration items.

### 3577 **12.2.8.3.3 ALC\_CMC.5.3C**

3578 **The CM documentation shall justify that the acceptance procedures provide for an adequate**  
 3579 **and appropriate review of changes to all configuration items.**

### 3580 **12.2.8.3.4 ALC\_CMC.5.4C**

3581 The CM system shall uniquely identify all configuration items.

### 3582 **12.2.8.3.5 ALC\_CMC.5.5C**

3583 The CM system shall provide automated controls such that only authorised changes are made to the  
 3584 configuration items.

### 3585 **12.2.8.3.6 ALC\_CMC.5.6C**

3586 The CM system shall support the production of the TOE by automated means.

### 3587 **12.2.8.3.7 ALC\_CMC.5.7C**

3588 **The CM system shall ensure that the person responsible for accepting a configuration item into**  
 3589 **CM is not the person who developed it.**

### 3590 **12.2.8.3.8 ALC\_CMC.5.8C**

3591 **The CM system shall identify the configuration items that comprise the TSF.**



3592 **12.2.8.3.9 ALC\_CMC.5.9C**

3593 **The CM system shall support the audit of all changes to the TOE by automated means, including**  
 3594 **the originator, date, and time in the audit trail.**

3595 **12.2.8.3.10 ALC\_CMC.5.10C**

3596 **The CM system shall provide an automated means to identify all other configuration items that**  
 3597 **are affected by the change of a given configuration item.**

3598 **12.2.8.3.11 ALC\_CMC.5.11C**

3599 **The CM system shall be able to identify the version of the implementation representation from**  
 3600 **which the TOE is generated.**

3601 **12.2.8.3.12 ALC\_CMC.5.12C**

3602 The CM documentation shall include a CM plan.

3603 **12.2.8.3.13 ALC\_CMC.5.13C**

3604 The CM plan shall describe how the CM system is used for the development of the TOE.

3605 **12.2.8.3.14 ALC\_CMC.5.14C**

3606 The CM plan shall describe the procedures used to accept modified or newly created configuration  
 3607 items as part of the TOE.

3608 **12.2.8.3.15 ALC\_CMC.5.15C**

3609 The evidence shall demonstrate that all configuration items are being maintained under the CM  
 3610 system.

3611 **12.2.8.3.16 ALC\_CMC.5.16C**

3612 The evidence shall demonstrate that the CM system is being operated in accordance with the CM plan.

3613 **12.2.8.4 Evaluator action elements**

3614 **12.2.8.4.1 ALC\_CMC.5.1E**

3615 The evaluator shall confirm that the information provided meets all requirements for content and  
 3616 presentation of evidence.

3617 **12.2.8.4.2 ALC\_CMC.5.2E**

3618 **The evaluator shall determine that the application of the production support procedures**  
 3619 **results in a TOE as provided by the developer for testing activities.**

### 3620 **12.3 CM scope (ALC\_CMS)**

#### 3621 **12.3.1 Objectives**

3622 The objective of this family is to identify items to be included as configuration items and hence placed  
 3623 under the CM requirements of CM capabilities (ALC\_CMC). Applying configuration management to  
 3624 these additional items provides additional assurance that the integrity of TOE is maintained.

#### 3625 **12.3.2 Component levelling**

3626 The components in this family are levelled on the basis of which of the following are required to be  
 3627 included as configuration items: the TOE and the evaluation evidence required by the SARs; the parts  
 3628 of the TOE; the implementation representation; security flaws; and development tools and related  
 3629 information.

#### 3630 **12.3.3 Application notes**

3631 While CM scope (ALC\_CMS) mandates a list of configuration items and that each item on this list be  
 3632 under CM, CM capabilities (ALC\_CMC) leaves the contents of the configuration list to the discretion of  
 3633 the developer. CM scope (ALC\_CMS) narrows this discretion by identifying items that must be included  
 3634 in the configuration list, and hence come under the CM requirements of CM capabilities (ALC\_CMC).

#### 3635 **12.3.4 ALC\_CMS.1 TOE CM coverage**

3636 Dependencies: No dependencies.

##### 3637 **12.3.4.1 Objectives**

3638 A CM system can control changes only to those items that have been placed under CM (i.e., the  
 3639 configuration items identified in the configuration list). Placing the TOE itself and the evaluation  
 3640 evidence required by the other SARs in the ST under CM provides assurance that they have been  
 3641 modified in a controlled manner with proper authorisations.

##### 3642 **12.3.4.2 Application notes**

3643 ALC\_CMS.1.1C introduces the requirement that the TOE itself and the evaluation evidence required by  
 3644 the other SARs in the ST be included in the configuration list and hence be subject to the CM  
 3645 requirements of CM capabilities (ALC\_CMC).

##### 3646 **12.3.4.3 Developer action elements**

###### 3647 **12.3.4.3.1 ALC\_CMS.1.1D**

3648 **The developer shall provide a configuration list for the TOE.**

##### 3649 **12.3.4.4 Content and presentation elements**

###### 3650 **12.3.4.4.1 ALC\_CMS.1.1C**

3651 **The configuration list shall include the following: the TOE itself; and the evaluation evidence**  
 3652 **required by the SARs.**

###### 3653 **12.3.4.4.2 ALC\_CMS.1.2C**

3654 **The configuration list shall uniquely identify the configuration items.**

3655 **12.3.4.5 Evaluator action elements**

3656 **12.3.4.5.1 ALC\_CMS.1.1E**

3657 **The evaluator shall confirm that the information provided meets all requirements for content**  
3658 **and presentation of evidence.**

3659 **12.3.5 ALC\_CMS.2 Parts of the TOE CM coverage**

3660 Dependencies: No dependencies.

3661 **12.3.5.1 Objectives**

3662 A CM system can control changes only to those items that have been placed under CM (i.e., the  
3663 configuration items identified in the configuration list). Placing the TOE itself, the parts that comprise  
3664 the TOE, and the evaluation evidence required by the other SARs under CM provides assurance that  
3665 they have been modified in a controlled manner with proper authorisations.

3666 **12.3.5.2 Application notes**

3667 ALC\_CMS.2.1C introduces the requirement that the parts that comprise the TOE (all parts that are  
3668 delivered to the consumer, for example hardware parts or executable files) be included in the  
3669 configuration list and hence be subject to the CM requirements of CM capabilities (ALC\_CMC).

3670 ALC\_CMS.2.3C introduces the requirement that the configuration list indicate the developer of each  
3671 TSF relevant configuration item. "Developer" here refers to an individual person.

3672 **12.3.5.3 Developer action elements**

3673 **12.3.5.3.1 ALC\_CMS.2.1D**

3674 The developer shall provide a configuration list for the TOE.

3675 **12.3.5.4 Content and presentation elements**

3676 **12.3.5.4.1 ALC\_CMS.2.1C**

3677 The configuration list shall include the following: the TOE itself; the evaluation evidence required by  
3678 the SARs; **and the parts that comprise the TOE.**

3679 **12.3.5.4.2 ALC\_CMS.2.2C**

3680 The configuration list shall uniquely identify the configuration items.

3681 **12.3.5.4.3 ALC\_CMS.2.3C**

3682 **For each TSF relevant configuration item, the configuration list shall indicate the developer of**  
3683 **the item.**

3684 **12.3.5.5 Evaluator action elements**

3685 **12.3.5.5.1 ALC\_CMS.2.1E**

3686 The evaluator shall confirm that the information provided meets all requirements for content and  
3687 presentation of evidence.

3688 **12.3.6 ALC\_CMS.3 Implementation representation CM coverage**

3689 Dependencies: No dependencies.

3690 **12.3.6.1 Objectives**

3691 A CM system can control changes only to those items that have been placed under CM (i.e., the  
3692 configuration items identified in the configuration list). Placing the TOE itself, the parts that comprise  
3693 the TOE, the TOE implementation representation and the evaluation evidence required by the other  
3694 SARs under CM provides assurance that they have been modified in a controlled manner with proper  
3695 authorisations.

3696 **12.3.6.2 Application notes**

3697 ALC\_CMS.3.1C introduces the requirement that the TOE implementation representation be included in  
3698 the list of configuration items and hence be subject to the CM requirements of CM capabilities  
3699 (ALC\_CMC).

3700 **12.3.6.3 Developer action elements**

3701 **12.3.6.3.1 ALC\_CMS.3.1D**

3702 The developer shall provide a configuration list for the TOE.

3703 **12.3.6.4 Content and presentation elements**

3704 **12.3.6.4.1 ALC\_CMS.3.1C**

3705 The configuration list shall include the following: the TOE itself; the evaluation evidence required by  
3706 the SARs; the parts that comprise the TOE; **and the implementation representation.**

3707 **12.3.6.4.2 ALC\_CMS.3.2C**

3708 The configuration list shall uniquely identify the configuration items.

3709 **12.3.6.4.3 ALC\_CMS.3.3C**

3710 For each TSF relevant configuration item, the configuration list shall indicate the developer of the item.

3711 **12.3.6.5 Evaluator action elements**

3712 **12.3.6.5.1 ALC\_CMS.3.1E**

3713 The evaluator shall confirm that the information provided meets all requirements for content and  
3714 presentation of evidence.

3715 **12.3.7 ALC\_CMS.4 Problem tracking CM coverage**

3716 Dependencies: No dependencies.

3717 **12.3.7.1 Objectives**

3718 A CM system can control changes only to those items that have been placed under CM (i.e., the  
3719 configuration items identified in the configuration list). Placing the TOE itself, the parts that comprise  
3720 the TOE, the TOE implementation representation and the evaluation evidence required by the other

3721 SARs under CM provides assurance that they have been modified in a controlled manner with proper  
3722 authorisations.

3723 Placing security flaw reports under CM ensures that the integrity of the reports is maintained and that  
3724 access to them is managed, further, it may support developers in tracking security flaws to their  
3725 resolution.

### 3726 **12.3.7.2 Application notes**

3727 ALC\_CMS.4.1C introduces the requirement that reports of identified security flaws be included in the  
3728 configuration list and hence be subject to the CM requirements of CM capabilities (ALC\_CMC). This  
3729 requires that information regarding previously identified security flaw reports and their resolution be  
3730 maintained.

### 3731 **12.3.7.3 Developer action elements**

#### 3732 **12.3.7.3.1 ALC\_CMS.4.1D**

3733 The developer shall provide a configuration list for the TOE.

### 3734 **12.3.7.4 Content and presentation elements**

#### 3735 **12.3.7.4.1 ALC\_CMS.4.1C**

3736 The configuration list shall include the following: the TOE itself; the evaluation evidence required by  
3737 the SARs; the parts that comprise the TOE; the implementation representation; **and security flaw**  
3738 **reports and resolution status.**

#### 3739 **12.3.7.4.2 ALC\_CMS.4.2C**

3740 The configuration list shall uniquely identify the configuration items.

#### 3741 **12.3.7.4.3 ALC\_CMS.4.3C**

3742 For each TSF relevant configuration item, the configuration list shall indicate the developer of the item.

### 3743 **12.3.7.5 Evaluator action elements**

#### 3744 **12.3.7.5.1 ALC\_CMS.4.1E**

3745 The evaluator shall confirm that the information provided meets all requirements for content and  
3746 presentation of evidence.

### 3747 **12.3.8 ALC\_CMS.5 Development tools CM coverage**

3748 Dependencies: No dependencies.

#### 3749 **12.3.8.1 Objectives**

3750 A CM system can control changes only to those items that have been placed under CM (i.e., the  
3751 configuration items identified in the configuration list). Placing the TOE itself, the parts that comprise  
3752 the TOE, the TOE implementation representation and the evaluation evidence required by the other  
3753 SARs under CM provides assurance that they have been modified in a controlled manner with proper  
3754 authorisations.

3755 Placing security flaw reports under CM ensures that the integrity of the reports is maintained and that  
 3756 access to them is managed, further, it may support developers in tracking security flaws to their  
 3757 resolution.

3758 Development tools play an important role in ensuring the production of a quality version of the TOE.  
 3759 Therefore, it is important to control modifications to these tools.

### 3760 **12.3.8.2 Application notes**

3761 ALC\_CMS.5.1C introduces the requirement that development tools and other related information be  
 3762 included in the list of configuration items and hence be subject to the CM requirements of CM  
 3763 capabilities (ALC\_CMC). Examples of development tools are programming languages and compilers.  
 3764 Information pertaining to TOE generation items (such as compiler options, generation options, and  
 3765 build options) is an example of information relating to development tools.

### 3766 **12.3.8.3 Developer action elements**

#### 3767 **12.3.8.3.1 ALC\_CMS.5.1D**

3768 The developer shall provide a configuration list for the TOE.

### 3769 **12.3.8.4 Content and presentation elements**

#### 3770 **12.3.8.4.1 ALC\_CMS.5.1C**

3771 The configuration list shall include the following: the TOE itself; the evaluation evidence required by  
 3772 the SARs; the parts that comprise the TOE; the implementation representation; security flaw reports  
 3773 and resolution status; **and development tools and related information.**

#### 3774 **12.3.8.4.2 ALC\_CMS.5.2C**

3775 The configuration list shall uniquely identify the configuration items.

#### 3776 **12.3.8.4.3 ALC\_CMS.5.3C**

3777 For each TSF relevant configuration item, the configuration list shall indicate the developer of the item.

### 3778 **12.3.8.5 Evaluator action elements**

#### 3779 **12.3.8.5.1 ALC\_CMS.5.1E**

3780 The evaluator shall confirm that the information provided meets all requirements for content and  
 3781 presentation of evidence.

## 3782 **12.4 Delivery (ALC\_DEL)**

### 3783 **12.4.1 Objectives**

3784 The concern of this family is the secure transfer of the finished TOE from the development  
 3785 environment into the responsibility of the user.

3786 The requirements for delivery call for system control and distribution facilities and procedures that  
 3787 detail the controls necessary to provide assurance that the security of the TOE is maintained during  
 3788 distribution of the TOE to the user. For a valid distribution of the TOE, the procedures used for the

3789 distribution of the TOE address the implied or identified objectives identified in the PP/ST relating to  
3790 the security of the TOE during delivery.

#### 3791 **12.4.2 Component levelling**

3792 This family contains only one component. An increasing level of protection for the TOE is established  
3793 by requiring that the delivery procedures are commensurate with the assumed attack potential in the  
3794 family Vulnerability analysis (AVA\_VAN) specified in the ST.

#### 3795 **12.4.3 Application notes**

3796 Transfers from subcontractors to the developer or between different development sites are not  
3797 considered here, but in the family Developer environment security (ALC\_DVS).

3798 The end of the delivery phase is marked by the acceptance of the transfer of the TOE into the  
3799 responsibility of the downstream user.

3800 NOTE: This does not necessarily coincide with the arrival of the TOE at the downstream user's  
3801 location.

3802 The delivery procedures should consider, if applicable, issues such as:

- 3803 a) ensuring that the TOE received by the consumer corresponds precisely to the evaluated version of  
3804 the TOE;
- 3805 b) avoiding or detecting any tampering with the actual version of the TOE;
- 3806 c) preventing submission of a counterfeit version of the TOE;
- 3807 d) avoiding unwanted knowledge of distribution of the TOE to the consumer: there might be cases  
3808 where potential attackers should not know when and how it is delivered;
- 3809 e) avoiding or detecting the TOE being intercepted during delivery; and
- 3810 f) avoiding the TOE being delayed or stopped during distribution.

3811 The delivery procedures should include the recipient's actions implied by these issues. The consistent  
3812 description of these implied actions is examined in the Preparative procedures (AGD\_PRE) family, if  
3813 present.

#### 3814 **12.4.4 ALC\_DEL.1 Delivery procedures**

3815 Dependencies: No dependencies.

##### 3816 **12.4.4.1 Developer action elements**

###### 3817 **12.4.4.1.1 ALC\_DEL.1.1D**

3818 **The developer shall document and provide procedures for delivery of the TOE or parts of it to**  
3819 **the consumer.**

###### 3820 **12.4.4.1.2 ALC\_DEL.1.2D**

3821 **The developer shall use the delivery procedures.**

3822 **12.4.4.2 Content and presentation elements**

3823 **12.4.4.2.1 ALC\_DEL.1.1C**

3824 **The delivery documentation shall describe all procedures that are necessary to maintain**  
 3825 **security when distributing versions of the TOE to the consumer.**

3826 **12.4.4.3 Evaluator action elements**

3827 **12.4.4.3.1 ALC\_DEL.1.1E**

3828 **The evaluator shall confirm that the information provided meets all requirements for content**  
 3829 **and presentation of evidence.**

3830 **12.5 Developer environment security (ALC\_DVS)**

3831 **12.5.1 Objectives**

3832 Development security is concerned with the determination and specification of security controls  
 3833 relating to the developer provided environment.

3834 NOTE: Such controls include coverage of security relevant aspects of asset management, human  
 3835 resources security, physical and environmental security, communications and operations  
 3836 management, access control, information systems acquisition, development and maintenance,  
 3837 information security incident management, and business continuity management.

3838 **12.5.2 Component levelling**

3839 The components in this family are levelled on the basis of whether justification of the sufficiency of the  
 3840 security controls is required.

3841 **12.5.3 Application notes**

3842 This family deals with controls to remove or reduce threads and security risks existing at the  
 3843 developer's site.

3844 The evaluator should visit the site(s) in order to assess evidence for development security. This may  
 3845 include sites of subcontractors involved in the TOE development and production. Any decision not to  
 3846 visit shall be agreed with the evaluation authority.

3847 Although development security deals with the maintenance of the TOE and hence with aspects  
 3848 becoming relevant after the completion of the evaluation, the Developer environment security  
 3849 (ALC\_DVS) requirements specify only that the development security controls be in place at the time of  
 3850 evaluation. Furthermore, Developer environment security (ALC\_DVS) does not contain any  
 3851 requirements related to the sponsor's intention to apply the development security controls in the  
 3852 future, after completion of the evaluation.

3853 It is recognised that confidentiality may not always be an issue for the protection of the TOE in its  
 3854 development environment. The use of the word "necessary" allows for the selection of appropriate  
 3855 safeguards.

3856 **12.5.4 ALC\_DVS.1 Identification of security controls**

3857 Dependencies: No dependencies.



3858 **12.5.4.1 Developer action elements**

3859 **12.5.4.1.1 ALC\_DVS.1.1D**

3860 **The developer shall produce and provide development security documentation.**

3861 **12.5.4.2 Content and presentation elements**

3862 **12.5.4.2.1 ALC\_DVS.1.1C**

3863 **The development security documentation shall describe all the physical, logical, procedural,**  
 3864 **personnel, and other security controls that are necessary to protect the confidentiality and**  
 3865 **integrity of the TOE design and implementation in its development environment.**

3866 **12.5.4.3 Evaluator action elements**

3867 **12.5.4.3.1 ALC\_DVS.1.1E**

3868 **The evaluator shall confirm that the information provided meets all requirements for content**  
 3869 **and presentation of evidence.**

3870 **12.5.4.3.2 ALC\_DVS.1.2E**

3871 **The evaluator shall confirm that the security controls are being applied.**

3872 **12.5.5 ALC\_DVS.2 Sufficiency of security controls**

3873 Dependencies: No dependencies.

3874 **12.5.5.1 Developer action elements**

3875 **12.5.5.1.1 ALC\_DVS.2.1D**

3876 **The developer shall produce and provide development security documentation.**

3877 **12.5.5.2 Content and presentation elements**

3878 **12.5.5.2.1 ALC\_DVS.2.1C**

3879 **The development security documentation shall describe all the physical, procedural, personnel, and**  
 3880 **other security controls that are necessary to protect the confidentiality and integrity of the TOE design**  
 3881 **and implementation in its development environment.**

3882 **12.5.5.2.2 ALC\_DVS.2.2C**

3883 **The development security documentation shall justify that the security controls provide the**  
 3884 **necessary level of protection to maintain the confidentiality and integrity of the TOE.**

3885 **12.5.5.3 Evaluator action elements**

3886 **12.5.5.3.1 ALC\_DVS.2.1E**

3887 **The evaluator shall confirm that the information provided meets all requirements for content and**  
 3888 **presentation of evidence.**

3889 **12.5.5.3.2 ALC\_DVS.2.2E**

3890 The evaluator shall confirm that the security controls are being applied.

3891 **12.6 Flaw remediation (ALC\_FLR)**

3892 **12.6.1 Objectives**

3893 Flaw remediation requires that discovered security flaws be tracked and corrected by the developer.  
 3894 Although future compliance with flaw remediation procedures cannot be determined at the time of the  
 3895 TOE evaluation, it is possible to evaluate the policies and procedures that a developer has in place to  
 3896 track and correct flaws, and to distribute the flaw information and corrections.

3897 **12.6.2 Component levelling**

3898 The components in this family are levelled on the basis of the increasing extent in scope of the flaw  
 3899 remediation procedures and the rigour of the flaw remediation policies.

3900 **12.6.3 Application notes**

3901 This family provides assurance that the TOE will be maintained and supported in the future, requiring  
 3902 the TOE developer to track and correct flaws in the TOE. Additionally, requirements are included for  
 3903 the distribution of flaw corrections. However, this family does not impose evaluation requirements  
 3904 beyond the current evaluation.

3905 The TOE user is considered to be the focal point in the user organisation that is responsible for  
 3906 receiving and implementing fixes to security flaws. This is not necessarily an individual user, but may  
 3907 be an organisational representative who is responsible for the handling of security flaws. The use of  
 3908 the term TOE user recognises that different organisations have different procedures for handling flaw  
 3909 reporting, which may be done either by an individual user, or by a central administrative body.

3910 The flaw remediation procedures should describe the methods for dealing with all types of flaws  
 3911 encountered. These flaws may be reported by the developer, by users of the TOE, or by other parties  
 3912 with familiarity with the TOE. Some flaws may not be reparable immediately. There may be some  
 3913 occasions where a flaw cannot be fixed and other (e.g. procedural) controls must be taken. The  
 3914 documentation provided should cover the procedures for providing the operational sites with fixes,  
 3915 and providing information on flaws where fixes are delayed (and what to do in the interim) or when  
 3916 fixes are not possible.

3917 Changes applied to a TOE after its release render it unevaluated; although some information from the  
 3918 original evaluation may still apply. The phrase "release of the TOE" used in this family therefore refers  
 3919 to a version of a product that is a release of a certified TOE, to which changes have been applied.

3920 **12.6.4 ALC\_FLR.1 Basic flaw remediation**

3921 Dependencies: No dependencies.

3922 **12.6.4.1 Developer action elements**

3923 **12.6.4.1.1 ALC\_FLR.1.1D**

3924 **The developer shall document and provide flaw remediation procedures addressed to TOE**  
 3925 **developers.**

- 3926 **12.6.4.2 Content and presentation elements**
- 3927 **12.6.4.2.1 ALC\_FLR.1.1C**
- 3928 **The flaw remediation procedures documentation shall describe the procedures used to track**  
 3929 **all reported security flaws in each release of the TOE.**
- 3930 **12.6.4.2.2 ALC\_FLR.1.2C**
- 3931 **The flaw remediation procedures shall require that a description of the nature and effect of**  
 3932 **each security flaw be provided, as well as the status of finding a correction to that flaw.**
- 3933 **12.6.4.2.3 ALC\_FLR.1.3C**
- 3934 **The flaw remediation procedures shall require that corrective actions be identified for each of**  
 3935 **the security flaws.**
- 3936 **12.6.4.2.4 ALC\_FLR.1.4C**
- 3937 **The flaw remediation procedures documentation shall describe the methods used to provide**  
 3938 **flaw information, corrections and guidance on corrective actions to TOE users.**
- 3939 **12.6.4.3 Evaluator action elements**
- 3940 **12.6.4.3.1 ALC\_FLR.1.1E**
- 3941 **The evaluator shall confirm that the information provided meets all requirements for content**  
 3942 **and presentation of evidence.**
- 3943 **12.6.5 ALC\_FLR.2 Flaw reporting procedures**
- 3944 Dependencies: No dependencies.
- 3945 **12.6.5.1 Objectives**
- 3946 In order for the developer to be able to act appropriately upon security flaw reports from TOE users,  
 3947 and to know to whom to send corrective fixes, TOE users need to understand how to submit security  
 3948 flaw reports to the developer. Flaw remediation guidance from the developer to the TOE user ensures  
 3949 that TOE users are aware of this important information.
- 3950 **12.6.5.2 Developer action elements**
- 3951 **12.6.5.2.1 ALC\_FLR.2.1D**
- 3952 The developer shall document and provide flaw remediation procedures addressed to TOE developers.
- 3953 **12.6.5.2.2 ALC\_FLR.2.2D**
- 3954 **The developer shall establish a procedure for accepting and acting upon all reports of security**  
 3955 **flaws and requests for corrections to those flaws.**
- 3956 **12.6.5.2.3 ALC\_FLR.2.3D**
- 3957 **The developer shall provide flaw remediation guidance addressed to TOE users.**

- 3958 **12.6.5.3 Content and presentation elements**
- 3959 **12.6.5.3.1 ALC\_FLR.2.1C**
- 3960 The flaw remediation procedures documentation shall describe the procedures used to track all  
3961 reported security flaws in each release of the TOE.
- 3962 **12.6.5.3.2 ALC\_FLR.2.2C**
- 3963 The flaw remediation procedures shall require that a description of the nature and effect of each  
3964 security flaw be provided, as well as the status of finding a correction to that flaw.
- 3965 **12.6.5.3.3 ALC\_FLR.2.3C**
- 3966 The flaw remediation procedures shall require that corrective actions be identified for each of the  
3967 security flaws.
- 3968 **12.6.5.3.4 ALC\_FLR.2.4C**
- 3969 The flaw remediation procedures documentation shall describe the methods used to provide flaw  
3970 information, corrections and guidance on corrective actions to TOE users.
- 3971 **12.6.5.3.5 ALC\_FLR.2.5C**
- 3972 **The flaw remediation procedures shall describe a means by which the developer receives from**  
3973 **TOE users reports and enquiries of suspected security flaws in the TOE.**
- 3974 **12.6.5.3.6 ALC\_FLR.2.6C**
- 3975 **The procedures for processing reported security flaws shall ensure that any reported flaws are**  
3976 **remediated and the remediation procedures issued to TOE users.**
- 3977 **12.6.5.3.7 ALC\_FLR.2.7C**
- 3978 **The procedures for processing reported security flaws shall provide safeguards that any**  
3979 **corrections to these security flaws do not introduce any new flaws.**
- 3980 **12.6.5.3.8 ALC\_FLR.2.8C**
- 3981 **The flaw remediation guidance shall describe a means by which TOE users report to the**  
3982 **developer any suspected security flaws in the TOE.**
- 3983 **12.6.5.4 Evaluator action elements**
- 3984 **12.6.5.4.1 ALC\_FLR.2.1E**
- 3985 The evaluator shall confirm that the information provided meets all requirements for content and  
3986 presentation of evidence.
- 3987 **12.6.6 ALC\_FLR.3 Systematic flaw remediation**
- 3988 Dependencies: No dependencies.

### 3989 **12.6.6.1 Objectives**

3990 In order for the developer to be able to act appropriately upon security flaw reports from TOE users,  
 3991 and to know to whom to send corrective fixes, TOE users need to understand how to submit security  
 3992 flaw reports to the developer, and how to register themselves with the developer so that they may  
 3993 receive these corrective fixes. Flaw remediation guidance from the developer to the TOE user ensures  
 3994 that TOE users are aware of this important information.

### 3995 **12.6.6.2 Developer action elements**

#### 3996 **12.6.6.2.1 ALC\_FLR.3.1D**

3997 The developer shall document and provide flaw remediation procedures addressed to TOE developers.

#### 3998 **12.6.6.2.2 ALC\_FLR.3.2D**

3999 The developer shall establish a procedure for accepting and acting upon all reports of security flaws  
 4000 and requests for corrections to those flaws.

#### 4001 **12.6.6.2.3 ALC\_FLR.3.3D**

4002 The developer shall provide flaw remediation guidance addressed to TOE users.

### 4003 **12.6.6.3 Content and presentation elements**

#### 4004 **12.6.6.3.1 ALC\_FLR.3.1C**

4005 The flaw remediation procedures documentation shall describe the procedures used to track all  
 4006 reported security flaws in each release of the TOE.

#### 4007 **12.6.6.3.2 ALC\_FLR.3.2C**

4008 The flaw remediation procedures shall require that a description of the nature and effect of each  
 4009 security flaw be provided, as well as the status of finding a correction to that flaw.

#### 4010 **12.6.6.3.3 ALC\_FLR.3.3C**

4011 The flaw remediation procedures shall require that corrective actions be identified for each of the  
 4012 security flaws.

#### 4013 **12.6.6.3.4 ALC\_FLR.3.4C**

4014 The flaw remediation procedures documentation shall describe the methods used to provide flaw  
 4015 information, corrections and guidance on corrective actions to TOE users.

#### 4016 **12.6.6.3.5 ALC\_FLR.3.5C**

4017 The flaw remediation procedures shall describe a means by which the developer receives from TOE  
 4018 users reports and enquiries of suspected security flaws in the TOE.

#### 4019 **12.6.6.3.6 ALC\_FLR.3.6C**

4020 **The flaw remediation procedures shall include a procedure requiring timely response and the**  
 4021 **automatic distribution of security flaw reports and the associated corrections to registered**  
 4022 **users who might be affected by the security flaw.**

4023 **12.6.6.3.7 ALC\_FLR.3.7C**

4024 The procedures for processing reported security flaws shall ensure that any reported flaws are  
4025 remediated and the remediation procedures issued to TOE users.

4026 **12.6.6.3.8 ALC\_FLR.3.8C**

4027 The procedures for processing reported security flaws shall provide safeguards that any corrections to  
4028 these security flaws do not introduce any new flaws.

4029 **12.6.6.3.9 ALC\_FLR.3.9C**

4030 The flaw remediation guidance shall describe a means by which TOE users report to the developer any  
4031 suspected security flaws in the TOE.

4032 **12.6.6.3.10 ALC\_FLR.3.10C**

4033 **The flaw remediation guidance shall describe a means by which TOE users may register with**  
4034 **the developer, to be eligible to receive security flaw reports and corrections.**

4035 **12.6.6.3.11 ALC\_FLR.3.11C**

4036 **The flaw remediation guidance shall identify the specific points of contact for all reports and**  
4037 **enquiries about security issues involving the TOE.**

4038 **12.6.6.4 Evaluator action elements**

4039 **12.6.6.4.1 ALC\_FLR.3.1E**

4040 The evaluator shall confirm that the information provided meets all requirements for content and  
4041 presentation of evidence.

4042 **12.7 Development Life-cycle definition (ALC\_LCD)**

4043 **12.7.1 Objectives**

4044 Poorly defined or uncontrolled processes applied during the development, production and  
4045 maintenance of the TOE can result in a TOE that does not meet all of its security objectives. Therefore,  
4046 it is important that well defined and controlled processes be established as early as possible in the  
4047 TOE's life-cycle.

4048 Defining and implementing such processes does not guarantee that the TOE meets all of its SFRs. It is  
4049 possible that the processes will be insufficient or inadequate.

4050 Adopting a life-cycle model, or models that meets the needs of the developer's organization will  
4051 improve the likelihood that the development, production and maintenance processes applied to TOE  
4052 support the correct design and implementation of a TOE that meets the specified SFRs.

4053 The determination of appropriate process controls in order to support process improvement is a long  
4054 established best practice.

4055 **12.7.2 Component levelling**

4056 The components in this family are levelled on the basis of increasing requirements for measurability of  
4057 the life-cycle model, and for compliance with that model.

### 4058 12.7.3 Application notes

4059 A life-cycle model encompasses the procedures, tools and techniques used to develop and maintain the  
 4060 TOE. Aspects of the process that may be covered by such a model include design methods, review  
 4061 procedures, project management controls, change control procedures, test methods and acceptance  
 4062 procedures. An effective life-cycle model will address these aspects of the development and  
 4063 maintenance process within an overall management structure that assigns responsibilities and  
 4064 monitors progress.

4065 There are different types of acceptance situations that are dealt with at different locations in this  
 4066 document:

- 4067 • acceptance of parts delivered by upstream developers for inclusion in the TOE are addressed in  
 4068 this family;
- 4069 • acceptance of parts due to internal transfers are addressed in Development security (ALC\_DVS);
- 4070 • acceptance of configuration items into the CM system is addressed in CM capabilities (ALC\_CMC);
- 4071 • acceptance of the TOE delivered to the downstream users is addressed in Delivery (ALC\_DEL); and
- 4072 • acceptance of security flaw reports is addressed in Flaw remediation (ALC\_FLR).

4073 The first three types of acceptance situations may overlap.

4074 Although development life-cycle definition deals with the maintenance of the TOE and hence with  
 4075 aspects becoming relevant after the completion of the evaluation, its evaluation adds assurance for  
 4076 processes invoked after the completion of the evaluation by providing an analysis of the life-cycle  
 4077 information for the TOE provided at the time of the evaluation.

4078 A life-cycle model provides a framework for the processes that control the development, production  
 4079 and maintenance of the TOE which mitigate the risk that the TOE will not meet its stated security  
 4080 requirements.

4081 EXAMPLE: Life cycle models include Waterfall “V”, Incremental, Agile, RAD, Spiral and Prototype.

4082 Measurement of both the product, and the development processes is a critical activity for successful  
 4083 software development. Process measurements specify some quantitative valuation of (arithmetic  
 4084 parameters and/or metrics) of the effectiveness of that process. Other measurements may be specified  
 4085 that are pertinent to the properties of the TOE itself.

4086 EXAMPLE Process measurements include, adherence to coding rules, development security training,  
 4087 the rate of discovery of vulnerabilities and security flaws.

4088 EXAMPLE Product source code complexity metrics, defect density (errors per size of code) or mean  
 4089 time to failure.

4090 For a security evaluation, the definition of such measurements by the developer is of relevance, since if  
 4091 used to promote process improvements will decrease the probability of faults and thereby in turn  
 4092 increase assurance in the security of the TOE.

### 4093 12.7.4 ALC\_LCD.1 Developer defined life-cycle processes

4094 Dependencies: No dependencies.

4095 **12.7.4.1 Developer action elements**

4096 **12.7.4.1.1 ALC\_LCD.1.1D**

4097 **The developer shall establish the life-cycle processes to be used in the development and**  
4098 **maintenance of the TOE.**

4099 **12.7.4.1.2 ALC\_LCD.1.2D**

4100 **The developer shall provide life-cycle definition documentation.**

4101 **12.7.4.2 Content and presentation elements**

4102 **12.7.4.2.1 ALC\_LCD.1.1C**

4103 **The life-cycle definition documentation shall describe the processes used to develop and**  
4104 **maintain the TOE.**

4105 **12.7.4.2.2 ALC\_LCD.1.2C**

4106 **The life-cycle processes documentation shall provide for the necessary control over the**  
4107 **development and maintenance of the TOE.**

4108 **12.7.4.3 Evaluator action elements**

4109 **12.7.4.3.1 ALC\_LCD.1.1E**

4110 **The evaluator shall confirm that the information provided meets all requirements for content**  
4111 **and presentation of evidence.**

4112 **Call for contribution**

4113 **As agreed during the Wuhan-meeting (DE/PL07), the editor requests contribution to clarify whether the**  
4114 **following aspect ALC\_LCD should remain in this standard.**

4115 **12.7.5 ALC\_LCD.2 Measurable life-cycle model**

4116 Dependencies: No dependencies.

4117 **12.7.5.1 Developer action elements**

4118 **12.7.5.1.1 ALC\_LCD.2.1D**

4119 **The developer shall establish the life-cycle processes to be used in the development and maintenance**  
4120 **of the TOE and specify appropriate measurements.**

4121 **12.7.5.1.2 ALC\_LCD.2.2D**

4122 **The developer shall provide life-cycle definition documentation.**

4123 **12.7.5.1.3 ALC\_LCD.2.3D**

4124 **The developer shall control the TOE development processes and security relevant properties of**  
4125 **the TOE using the defined measurements.**



4126 **12.7.5.1.4 ALC\_LCD.2.4D**

4127 **The developer shall provide life-cycle output documentation.**

4128 **12.7.5.1.5 ALC\_LCD.2.5D**

4129 **The developer shall use the measurements of the TOE development processes and security**  
 4130 **relevant properties of the TOE to support improvements in the development processes and/or**  
 4131 **the TOE itself.**

4132 **12.7.5.2 Content and presentation elements**

4133 **12.7.5.2.1 ALC\_LCD.2.1C**

4134 The life-cycle definition documentation shall describe the processes used to develop and maintain the  
 4135 TOE **including the details of its arithmetic parameters and/or metrics used to control the**  
 4136 **security relevant properties of the TOE and its development processes.**

4137 **12.7.5.2.2 ALC\_LCD.2.2C**

4138 The life-cycle processes documentation shall provide for the necessary control over the development  
 4139 and maintenance of the TOE.

4140 **12.7.5.2.3 ALC\_LCD.2.3C**

4141 **The life-cycle process output documentation shall include the results of the measurements of**  
 4142 **the security relevant properties of the TOE and its development processes.**

4143 **12.7.5.2.4 ALC\_LCD.2.4C**

4144 **The life-cycle process output documentation shall include records of improvements made in**  
 4145 **the security relevant properties of the TOE and its development processes.**

4146 **12.7.5.3 Evaluator action elements**

4147 **12.7.5.3.1 ALC\_LCD.2.1E**

4148 The evaluator shall confirm that the information provided meets all requirements for content and  
 4149 presentation of evidence.

4150 **12.7.5.3.2 ALC\_LCD.2.2E**

4151 **The evaluator shall confirm that the measurements of the TOE development processes and**  
 4152 **security relevant properties of the TOE support improvements in the development processes**  
 4153 **and/or the TOE itself.**

4154 **12.8 Practices for trustable development (ALC\_PTD)**

4155 **12.8.1 Objectives**

4156 The concept of this family aims to add trust to the development process or a development (for  
 4157 brevity). It focuses on the generation of certain artifacts in the development process. These artifacts  
 4158 are used at a later point in time to assess the degree to which the development process or the  
 4159 development is trustable. This trust is realized through the validation of the generated artifacts for  
 4160 confirming them as sufficient evidence for trustable development.

4161 This family introduces developer practices within the development process to generate the required  
 4162 artifacts for realizing trustable development. Unless explicitly stated in the requirements, the  
 4163 developer is free to undertake a specific practice manually, or using some integrated automation in the  
 4164 development process, or using a hybrid method of both. It is expected that the degree of trust in the  
 4165 development process is proportional to the degree of automation adoption to implement the  
 4166 corresponding practice in the development process.

4167 This family also has a closer relationship with ALC\_TAT. As ALC\_TAT focuses on the tools and  
 4168 techniques aspect for developing, analysing, and implementing the TOE, it provides the necessary  
 4169 context when describing the practices of this family being introduced into the development process.

## 4170 **12.8.2 Component levelling**

4171 The components in this family are levelled on the basis of increasing cross-checking for consistency  
 4172 with relevant evidence from components of other families of other security assurance classes.

## 4173 **12.8.3 Application notes**

4174 The requirements in ALC\_PTD.1 are essential to provide a degree of trust in the developer's ability to  
 4175 identify the set of implementation representation which actually has been used during the TOE  
 4176 generation time. This degree of trust helps to positively answer the question "is that really the source  
 4177 code for this software" or "is that really the set of implementation representation for this TOE", which  
 4178 is potentially relevant in an evaluation. Such degree of trust is built on

- 4179 a) the timing of when the set of implementation representation identifiers is recorded or logged,
- 4180 b) the integrity and authenticity of the record of implementation representation identifiers, and
- 4181 c) the traceability of implementation representation identifiers from the TOE.

4182 In the case where some implementation representation elements are also covered in the configuration  
 4183 list due to ALC\_CMS.3, the requirements in ALC\_PTD.2 make sure that these implementation  
 4184 representation elements actually are identifiable through the use of the implementation  
 4185 representation identifiers of ALC\_PTD.1.

4186 With the accurate recording or logging of the actual implementation representation being used by the  
 4187 development tools under the scope of ALC\_TAT, it provides an additional evidence to convince a third  
 4188 party that a regeneration of the TOE is functionally equivalent to the original TOE.

4189 The requirements in ALC\_PTD.3 provide the developer an opportunity to testify the absence of  
 4190 functional differences between the two possibly visibly different TOEs which have been independently  
 4191 generated from the identical set of implementation representation.

## 4192 **12.8.4 ALC\_PTD.1 Uniquely identifying implementation representation**

4193 Dependencies: No dependencies.

### 4194 **12.8.4.1 Developer action elements**

#### 4195 **12.8.4.1.1 ALC\_PTD.1.1D**

4196 **The developer shall identify individual elements of the TOE implementation representation to**  
 4197 **record the list of unique TOE implementation representation identifiers, as the development**  
 4198 **tool generates the TOE.**

4199 **12.8.4.1.2 ALC\_PTD.1.2D**

4200 **The developer shall use the current date and time to timestamp the list of unique TOE**  
 4201 **implementation representation identifiers as recorded during the TOE generation time.**

4202 **12.8.4.1.3 ALC\_PTD.1.3D**

4203 **The developer shall maintain the integrity of the list of unique TOE implementation**  
 4204 **representation identifiers as recorded during the TOE generation time.**

4205 **12.8.4.1.4 ALC\_PTD.1.4D**

4206 **The developer shall ensure the authenticity of the list of unique TOE implementation**  
 4207 **representation identifiers as recorded during the TOE generation time, with the maintenance**  
 4208 **of the (author) origination information.**

4209 **12.8.4.1.5 ALC\_PTD.1.5D**

4210 **The developer shall be able to trace from the TOE to the list of unique TOE implementation**  
 4211 **representation identifiers as recorded during the TOE generation time.**

4212 **12.8.4.1.6 ALC\_PTD.1.6D**

4213 **The developer shall produce and provide documentation describing**

4214 **a) the developer's creation of the list of unique TOE implementation representation**  
 4215 **identifiers as recorded during the TOE generation time;**

4216 **b) the developer's timestamp being applied to the list of unique TOE implementation**  
 4217 **representation identifiers as recorded during the TOE generation time;**

4218 **c) the maintenance of the (author) origination information of the list of unique TOE**  
 4219 **implementation representation identifiers as recorded during the TOE generation time;**

4220 **d) the maintenance of the integrity of the list of unique TOE implementation**  
 4221 **representation identifiers as recorded during the TOE generation time and its**  
 4222 **associated timestamp and (author) origination information;**

4223 **e) the developer's mechanism to trace from the TOE to the list of unique TOE**  
 4224 **implementation representation identifiers as recorded during the TOE generation time.**

4225 **12.8.4.2 Content and presentation elements**

4226 **12.8.4.2.1 ALC\_PTD.1.1C**

4227 **The list of unique TOE implementation representation identifiers as recorded during the TOE**  
 4228 **generation time shall demonstrate the correspondence between the TOE implementation**  
 4229 **representation element identifiers and the TOE implementation representation element**  
 4230 **names.**

4231 **12.8.4.2.2 ALC\_PTD.1.2C**

4232 **The TOE implementation representation element names shall be in the same form as used or**  
 4233 **referenced by the development tool to generate the TOE.**

4234 **12.8.4.2.3 ALC\_PTD.1.3C**

4235 **The timestamp of the list of unique TOE implementation representation identifiers as recorded**  
 4236 **during the TOE generation time shall be consistent with the creation time of the TOE.**

4237 **12.8.4.2.4 ALC\_PTD.1.4C**

4238 **The (author) origination information of the list of unique TOE implementation representation**  
 4239 **identifiers as recorded during the TOE generation time shall be consistent with the (author)**  
 4240 **origination information of the TOE.**

4241 **12.8.4.3 Evaluator action elements**

4242 **12.8.4.3.1 ALC\_PTD.1.1E**

4243 **The evaluator shall confirm that the information provided meets all requirements for content**  
 4244 **and presentation of evidence.**

4245 **12.8.4.3.2 ALC\_PTD.1.2E**

4246 **The evaluator shall confirm that the development tool for generating the TOE is capable to use**  
 4247 **or reference the implementation representation element names.**

4248 **12.8.4.3.3 ALC\_PTD.1.3E**

4249 **The evaluator shall confirm that the list of unique TOE implementation representation**  
 4250 **identifiers as recorded during the TOE generation time is consistent with the creation time of**  
 4251 **the TOE.**

4252 **12.8.4.3.4 ALC\_PTD.1.4E**

4253 **The evaluator shall confirm that the (author) origination information of the list of unique TOE**  
 4254 **implementation representation identifiers as recorded during the TOE generation time is**  
 4255 **consistent with the (author) origination information of the TOE.**

4256 **12.8.4.3.5 ALC\_PTD.1.5E**

4257 **The evaluator shall check the integrity of the list of unique TOE implementation representation**  
 4258 **identifiers as recorded during the TOE generation time and its associated timestamp and**  
 4259 **(author) origination information.**

4260 **12.8.4.3.6 ALC\_PTD.1.6E**

4261 **The evaluator shall confirm the developer's ability to trace from the TOE to the list of unique**  
 4262 **TOE implementation representation identifiers as recorded during the TOE generation time.**

4263 **12.8.5 ALC\_PTD.2 Matching CMS scope of implementation representation**

4264 Dependencies: ALC\_CMS.3 Implementation representation CM coverage

4265 **12.8.5.1 Developer action elements**

4266 **12.8.5.1.1 ALC\_PTD.2.1D**

4267 The developer shall identify individual elements of the TOE implementation representation to record  
 4268 the list of unique TOE implementation representation identifiers, as the development tool generates  
 4269 the TOE.

4270 **12.8.5.1.2 ALC\_PTD.2.2D**

4271 The developer shall use the current date and time to timestamp the list of unique TOE implementation  
 4272 representation identifiers as recorded during the TOE generation time.

4273 **12.8.5.1.3 ALC\_PTD.2.3D**

4274 The developer shall maintain the integrity of the list of unique TOE implementation representation  
 4275 identifiers as recorded during the TOE generation time.

4276 **12.8.5.1.4 ALC\_PTD.2.4D**

4277 The developer shall ensure the authenticity of the list of unique TOE implementation representation  
 4278 identifiers as recorded during the TOE generation time, with the maintenance of the (author)  
 4279 origination information.

4280 **12.8.5.1.5 ALC\_PTD.2.5D**

4281 The developer shall be able to trace from the TOE to the list of unique TOE implementation  
 4282 representation identifiers as recorded during the TOE generation time.

4283 **12.8.5.1.6 ALC\_PTD.2.6D**

4284 The developer shall produce and provide documentation describing

- 4285 a) the developer's creation of the list of unique TOE implementation representation identifiers as  
 4286 recorded during the TOE generation time;
- 4287 b) the developer's timestamp being applied to the list of unique TOE implementation  
 4288 representation identifiers as recorded during the TOE generation time;
- 4289 c) the maintenance of the (author) origination information of the list of unique TOE  
 4290 implementation representation identifiers as recorded during the TOE generation time;
- 4291 d) the maintenance of the integrity of the list of unique TOE implementation representation  
 4292 identifiers as recorded during the TOE generation time and its associated timestamp and  
 4293 (author) origination information;
- 4294 e) the developer's mechanism to trace from the TOE to the list of unique TOE implementation  
 4295 representation identifiers as recorded during the TOE generation time.

4296 **12.8.5.1.7 ALC\_PTD.2.7D**

4297 **The developer shall provide evidence that the elements of implementation representation**  
 4298 **under the configuration scope of ALC\_CMS.3 are identified by the list of unique TOE**  
 4299 **implementation representation identifiers as recorded during the TOE generation time.**

4300 **12.8.5.2 Content and presentation elements**

4301 **12.8.5.2.1 ALC\_PTD.2.1C**

4302 The list of unique TOE implementation representation identifiers as recorded during the TOE  
4303 generation time shall demonstrate the correspondence between the TOE implementation  
4304 representation element identifiers and the TOE implementation representation element names.

4305 **12.8.5.2.2 ALC\_PTD.2.2C**

4306 The TOE implementation representation element names shall be in the same form as used or  
4307 referenced by the development tool to generate the TOE.

4308 **12.8.5.2.3 ALC\_PTD.2.3C**

4309 The timestamp of the list of unique TOE implementation representation identifiers as recorded during  
4310 the TOE generation time shall be consistent with the creation time of the TOE.

4311 **12.8.5.2.4 ALC\_PTD.2.4C**

4312 The (author) origination information of the list of unique TOE implementation representation  
4313 identifiers as recorded during the TOE generation time shall be consistent with the (author)  
4314 origination information of the TOE.

4315 **12.8.5.2.5 ALC\_PTD.2.5C**

4316 **The list of identifiers of the elements of implementation representation under the**  
4317 **configuration scope of ALC\_CMS.3 shall match with the list of unique TOE implementation**  
4318 **representation identifiers as recorded during the TOE generation time.**

4319 **12.8.5.3 Evaluator action elements**

4320 **12.8.5.3.1 ALC\_PTD.2.1E**

4321 The evaluator shall confirm that the information provided meets all requirements for content and  
4322 presentation of evidence.

4323 **12.8.5.3.2 ALC\_PTD.2.2E**

4324 The evaluator shall confirm that the development tool for generating the TOE is capable to use or  
4325 reference the implementation representation element names.

4326 **12.8.5.3.3 ALC\_PTD.2.3E**

4327 The evaluator shall confirm that the list of unique TOE implementation representation identifiers as  
4328 recorded during the TOE generation time is consistent with the creation time of the TOE.

4329 **12.8.5.3.4 ALC\_PTD.2.4E**

4330 The evaluator shall confirm that the (author) origination information of the list of unique TOE  
4331 implementation representation identifiers as recorded during the TOE generation time is consistent  
4332 with the (author) origination information of the TOE.

4333 **12.8.5.3.5 ALC\_PTD.2.5E**

4334 The evaluator shall check the integrity of the list of unique TOE implementation representation  
4335 identifiers as recorded during the TOE generation time and its associated timestamp and (author)  
4336 origination information.

4337 **12.8.5.3.6 ALC\_PTD.2.6E**

4338 The evaluator shall confirm the developer's ability to trace from the TOE to the list of unique TOE  
4339 implementation representation identifiers as recorded during the TOE generation time.

4340 **12.8.5.3.7 ALC\_PTD.2.7E**

4341 **The evaluator shall confirm that the list of identifiers of the elements of implementation**  
4342 **representation under the configuration scope of ALC\_CMS.3 matches with the list of unique TOE**  
4343 **implementation representation identifiers as recorded during the TOE generation time.**

4344 **12.8.6 ALC\_PTD.3 Regenerate TOE with well-defined development tools**

4345 Dependencies: ALC\_TAT.1 Well-defined development tools and

4346 ADV\_IMP.1 Implementation representation of the TSF

4347 **12.8.6.1 Developer action elements**

4348 **12.8.6.1.1 ALC\_PTD.3.1D**

4349 The developer shall identify individual elements of the TOE implementation representation to record  
4350 the list of unique TOE implementation representation identifiers, as the development tool generates  
4351 the TOE.

4352 **12.8.6.1.2 ALC\_PTD.3.2D**

4353 The developer shall use the current date and time to timestamp the list of unique TOE implementation  
4354 representation identifiers as recorded during the TOE generation time.

4355 **12.8.6.1.3 ALC\_PTD.3.3D**

4356 The developer shall maintain the integrity of the list of unique TOE implementation representation  
4357 identifiers as recorded during the TOE generation time.

4358 **12.8.6.1.4 ALC\_PTD.3.4D**

4359 The developer shall ensure the authenticity of the list of unique TOE implementation representation  
4360 identifiers as recorded during the TOE generation time, with the maintenance of the (author)  
4361 origination information.

4362 **12.8.6.1.5 ALC\_PTD.3.5D**

4363 The developer shall be able to trace from the TOE to the list of unique TOE implementation  
4364 representation identifiers as recorded during the TOE generation time.

4365 **12.8.6.1.6 ALC\_PTD.3.6D**

4366 The developer shall produce and provide documentation describing

- a) the developer's creation of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time;
- b) the developer's timestamp being applied to the list of unique TOE implementation representation identifiers as recorded during the TOE generation time;
- c) the maintenance of the (author) origination information of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time;
- d) the maintenance of the integrity of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time and its associated timestamp and (author) origination information;
- e) the developer's mechanism to trace from the TOE to the list of unique TOE implementation representation identifiers as recorded during the TOE generation time.

#### **12.8.6.1.7 ALC\_PTD.3.7D**

The developer shall provide evidence that the elements of implementation representation under the configuration scope of ALC\_CMS.3 are identified by the list of unique TOE implementation representation identifiers as recorded during the TOE generation time.

#### **12.8.6.1.8 ALC\_PTD.3.8D**

**After applying the development tools to another copy of the TOE implementation representation according to the list of unique TOE implementation representation identifiers to regenerate a TOE copy, the developer shall explain the functional differences, if any, between the TOE copy and the original TOE.**

#### **12.8.6.1.9 ALC\_PTD.3.2D**

**The developer shall produce and provide documentation explaining the functional differences, if any, between the regenerated TOE copy and the original TOE.**

### **12.8.6.2 Content and presentation elements**

#### **12.8.6.2.1 ALC\_PTD.3.1C**

The list of unique TOE implementation representation identifiers as recorded during the TOE generation time shall demonstrate the correspondence between the TOE implementation representation element identifiers and the TOE implementation representation element names.

#### **12.8.6.2.2 ALC\_PTD.3.2C**

The TOE implementation representation element names shall be in the same form as used or referenced by the development tool to generate the TOE.

#### **12.8.6.2.3 ALC\_PTD.3.3C**

The timestamp of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time shall be consistent with the creation time of the TOE.

#### **12.8.6.2.4 ALC\_PTD.3.4C**

The (author) origination information of the list of unique TOE implementation representation identifiers as recorded during the TOE generation time shall be consistent with the (author) origination information of the TOE.



4405 **12.8.6.2.5 ALC\_PTD.3.5C**

4406 The list of identifiers of the elements of implementation representation under the configuration scope  
4407 of ALC\_CMS.3 shall match with the list of unique TOE implementation representation identifiers as  
4408 recorded during the TOE generation time.

4409 **12.8.6.2.6 ALC\_PTD.3.6C**

4410 **The developer's explanation of the functional differences, if any, between the regenerated TOE**  
4411 **copy and the original TOE shall take into account all visible differences, if any, between the**  
4412 **regenerated TOE copy and the original TOE.**

4413 **12.8.6.3 Evaluator action elements**

4414 **12.8.6.3.1 ALC\_PTD.3.1E**

4415 The evaluator shall confirm that the information provided meets all requirements for content and  
4416 presentation of evidence.

4417 **12.8.6.3.2 ALC\_PTD.3.2E**

4418 The evaluator shall confirm that the development tool for generating the TOE is capable to use or  
4419 reference the implementation representation element names.

4420 **12.8.6.3.3 ALC\_PTD.3.3E**

4421 The evaluator shall confirm that the list of unique TOE implementation representation identifiers as  
4422 recorded during the TOE generation time is consistent with the creation time of the TOE.

4423 **12.8.6.3.4 ALC\_PTD.3.4E**

4424 The evaluator shall confirm that the (author) origination information of the list of unique TOE  
4425 implementation representation identifiers as recorded during the TOE generation time is consistent  
4426 with the (author) origination information of the TOE.

4427 **12.8.6.3.5 ALC\_PTD.3.5E**

4428 The evaluator shall check the integrity of the list of unique TOE implementation representation  
4429 identifiers as recorded during the TOE generation time and its associated timestamp and (author)  
4430 origination information.

4431 **12.8.6.3.6 ALC\_PTD.3.6E**

4432 The evaluator shall confirm the developer's ability to trace from the TOE to the list of unique TOE  
4433 implementation representation identifiers as recorded during the TOE generation time.

4434 **12.8.6.3.7 ALC\_PTD.3.7E**

4435 The evaluator shall confirm that the list of identifiers of the elements of implementation  
4436 representation under the configuration scope of ALC\_CMS.3 matches with the list of unique TOE  
4437 implementation representation identifiers as recorded during the TOE generation time.

4438 **12.8.6.3.8 ALC\_PTD.3.8E**

4439 **The evaluator shall check that the developer's explanation of the functional differences, if any,**  
 4440 **between the regenerated TOE copy and the original TOE takes into account all visible**  
 4441 **differences, if any, between the regenerated TOE copy and the original TOE.**

4442 **12.9 Tools and techniques (ALC\_TAT)**4443 **12.9.1 Objectives**

4444 Tools and techniques is an aspect of selecting tools that are used to develop, analyse and implement  
 4445 the TOE. It includes requirements to prevent ill-defined, inconsistent or incorrect development tools  
 4446 from being used to develop the TOE. This includes, but is not limited to, programming languages,  
 4447 documentation, implementation standards, and other parts of the TOE such as supporting runtime  
 4448 libraries.

4449 **12.9.2 Component levelling**

4450 The components in this family are levelled on the basis of increasing requirements on the description  
 4451 and scope of the implementation standards and the documentation of implementation-dependent  
 4452 options.

4453 **12.9.3 Application notes**

4454 There is a requirement for well-defined development tools. These are tools that are clearly and  
 4455 completely described. For example, programming languages and computer aided design (CAD)  
 4456 systems that are based on a standard published by standards bodies are considered to be well-defined.  
 4457 Self-made tools would need further investigation to clarify whether they are well-defined.

4458 The requirement in ALC\_TAT.1.2C is especially applicable to programming languages so as to ensure  
 4459 that all statements in the source code have an unambiguous meaning.

4460 In ALC\_TAT.2 and ALC\_TAT.3, implementation guidelines may be accepted as an implementation  
 4461 standard if they have been approved by some group of experts (e.g. academic experts, standards  
 4462 bodies). Implementation standards are normally public, well accepted and common practise in a  
 4463 specific industry, but developer-specific implementation guidelines may also be accepted as a  
 4464 standard; the emphasis is on the expertise.

4465 Tools and techniques distinguishes between the implementation standards applied by the developer  
 4466 (ALC\_TAT.2.3D) and the implementation standards for "all parts of the TOE" (ALC\_TAT.3.3D) which  
 4467 include third party software, hardware, or firmware. The configuration list introduced in CM scope  
 4468 (ALC\_CMS) requires that for each TSF relevant configuration item to indicate if it has been generated  
 4469 by the TOE developer or by third party developers

4470 **12.9.4 ALC\_TAT.1 Well-defined development tools**

4471 Dependencies: ADV\_IMP.1 Implementation representation of the TSF

4472 **12.9.4.1 Developer action elements**4473 **12.9.4.1.1 ALC\_TAT.1.1D**

4474 **The developer shall provide the documentation identifying each development tool being used**  
 4475 **for the TOE.**

4476 **12.9.4.1.2 ALC\_TAT.1.2D**

4477 **The developer shall document and provide the selected implementation-dependent options of**  
 4478 **each development tool.**

4479 **12.9.4.2 Content and presentation elements**

4480 **12.9.4.2.1 ALC\_TAT.1.1C**

4481 **Each development tool used for implementation shall be well-defined.**

4482 **12.9.4.2.2 ALC\_TAT.1.2C**

4483 **The documentation of each development tool shall unambiguously define the meaning of all**  
 4484 **statements as well as all conventions and directives used in the implementation.**

4485 **12.9.4.2.3 ALC\_TAT.1.3C**

4486 **The documentation of each development tool shall unambiguously define the meaning of all**  
 4487 **implementation-dependent options.**

4488 **12.9.4.3 Evaluator action elements**

4489 **12.9.4.3.1 ALC\_TAT.1.1E**

4490 **The evaluator shall confirm that the information provided meets all requirements for content**  
 4491 **and presentation of evidence.**

4492 **12.9.5 ALC\_TAT.2 Compliance with implementation standards**

4493 Dependencies: ADV\_IMP.1 Implementation representation of the TSF

4494 **12.9.5.1 Developer action elements**

4495 **12.9.5.1.1 ALC\_TAT.2.1D**

4496 **The developer shall provide the documentation identifying each development tool being used for the**  
 4497 **TOE.**

4498 **12.9.5.1.2 ALC\_TAT.2.2D**

4499 **The developer shall document and provide the selected implementation-dependent options of each**  
 4500 **development tool.**

4501 **12.9.5.1.3 ALC\_TAT.2.3D**

4502 **The developer shall describe and provide the implementation standards that are being applied**  
 4503 **by the developer.**

4504 **12.9.5.2 Content and presentation elements**

4505 **12.9.5.2.1 ALC\_TAT.2.1C**

4506 **Each development tool used for implementation shall be well-defined.**

4507 **12.9.5.2.2 ALC\_TAT.2.2C**

4508 The documentation of each development tool shall unambiguously define the meaning of all  
4509 statements as well as all conventions and directives used in the implementation.

4510 **12.9.5.2.3 ALC\_TAT.2.3C**

4511 The documentation of each development tool shall unambiguously define the meaning of all  
4512 implementation-dependent options.

4513 **12.9.5.3 Evaluator action elements**

4514 **12.9.5.3.1 ALC\_TAT.2.1E**

4515 The evaluator shall confirm that the information provided meets all requirements for content and  
4516 presentation of evidence.

4517 **12.9.5.3.2 ALC\_TAT.2.2E**

4518 **The evaluator shall confirm that the implementation standards have been applied.**

4519 **12.9.6 ALC\_TAT.3 Compliance with implementation standards - all parts**

4520 Dependencies: ADV\_IMP.1 Implementation representation of the TSF

4521 **12.9.6.1 Developer action elements**

4522 **12.9.6.1.1 ALC\_TAT.3.1D**

4523 The developer shall provide the documentation identifying each development tool being used for the  
4524 TOE.

4525 **12.9.6.1.2 ALC\_TAT.3.2D**

4526 The developer shall document and provide the selected implementation-dependent options of each  
4527 development tool.

4528 **12.9.6.1.3 ALC\_TAT.3.3D**

4529 The developer shall describe and provide the implementation standards that are being applied by the  
4530 developer **and by any third-party providers for all parts of the TOE.**

4531 **12.9.6.2 Content and presentation elements**

4532 **12.9.6.2.1 ALC\_TAT.3.1C**

4533 Each development tool used for implementation shall be well-defined.

4534 **12.9.6.2.2 ALC\_TAT.3.2C**

4535 The documentation of each development tool shall unambiguously define the meaning of all  
4536 statements as well as all conventions and directives used in the implementation.

4537 **12.9.6.2.3 ALC\_TAT.3.3C**

4538 The documentation of each development tool shall unambiguously define the meaning of all  
4539 implementation-dependent options.

4540 **12.9.6.3 Evaluator action elements**

4541 **12.9.6.3.1 ALC\_TAT.3.1E**

4542 The evaluator shall confirm that the information provided meets all requirements for content and  
4543 presentation of evidence.

4544 **12.9.6.3.2 ALC\_TAT.3.2E**

4545 **The evaluator shall confirm that the implementation standards have been applied.**

4546 **12.10 Integration of composition parts and consistency check of delivery procedures**  
4547 **(ALC\_COMP)**

4548 **12.10.1 Objectives**

4549 The aims of this activity are to determine whether

- 4550 - the correct version of the application is installed onto/into the correct version of the  
4551 underlying platform, and
- 4552 - the preparative guidance procedures of Platform and Application Developers are compatible  
4553 with the acceptance procedure of the Composite Product Integrator.

4554 **12.10.2 Component levelling**

4555 This family contains only one component.

4556 **12.10.3 ALC\_COMP.1 Integration of the application into the underlying platform and**  
4557 **Consistency check for delivery and acceptance procedures**

4558 Dependencies: No dependencies

4559 **12.10.3.1 Developer action elements**

4560 **12.10.3.1.1 ALC\_COMP.1.1D**

4561 The developer shall provide components configuration evidence; cf. item #7, item #8 and item #3 in  
4562 Table D1, section D.1.7.

4563 **12.10.3.2 Content and presentation elements**

4564 **12.10.3.2.1 ALC\_COMP.1.1C**

4565 The components configuration evidence shall show that the evaluated version of the application has  
4566 been installed onto / embedded into the certified version of the underlying platform.

4567 **12.10.3.2.2 ALC\_COMP.1.2C**

4568 The components configuration evidence shall show that:

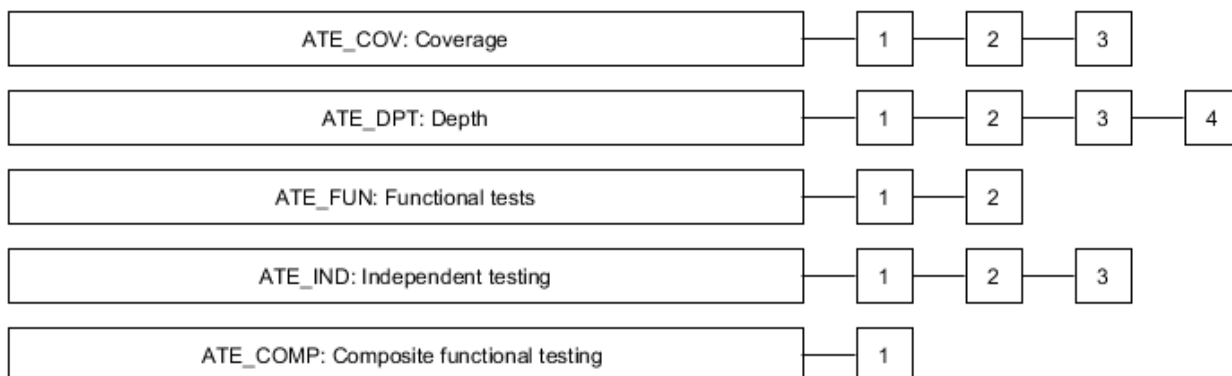
i. The evidence for delivery and acceptance compatibility shall show that the delivery procedures of the Platform and Application Developers are compatible with the acceptance procedure of the Composite Product Integrator.

ii. the evidence shall show that preparative guidance procedures prescribed by the Platform and Application Developers are either actually being used by the Composite Product Integrator or compatible with the Composite Product Integrator guidance and do not contradict each other.

### 12.10.3.3 Evaluator action elements

## 12.10

The e  
prese



## 12.10

The e  
interr

## 13 C

### 13.1

#### Introduction

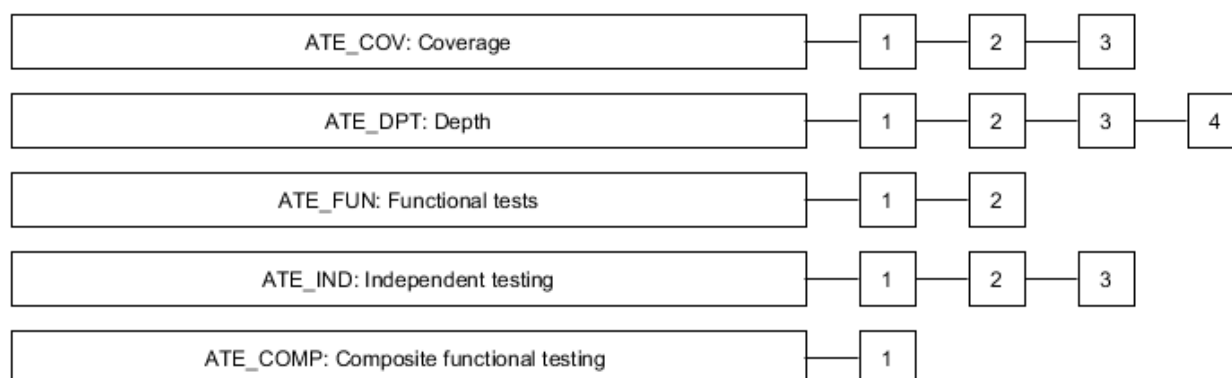
The class “Tests” encompasses five families: Coverage (ATE\_COV), Depth (ATE\_DPT), Independent testing (ATE\_IND) (i.e. functional testing performed by evaluators), Functional tests (ATE\_FUN) and Composite functional testing (ATE\_COMP). Testing provides assurance that the TSF behaves as described (in the functional specification, TOE design, implementation representation, and allows straightforward traceability of SFR in test scenario).

The emphasis in this class is on confirmation that the TSF operates according to its design descriptions. This class does not address penetration testing, which is based upon an analysis of the TSF that specifically seeks to identify vulnerabilities in the design and implementation of the TSF. Penetration testing is addressed separately as an aspect of vulnerability assessment in the AVA: Vulnerability assessment class.

The ATE: Tests class separates testing into developer testing and evaluator testing. The Coverage (ATE\_COV), and Depth (ATE\_DPT) families address the completeness of developer testing. Coverage (ATE\_COV) addresses the rigour with which the functional specification is tested; Depth (ATE\_DPT) addresses whether testing against other design descriptions (security architecture, TOE design, and implementation representation) is required.

Functional tests (ATE\_FUN) addresses the performing of the tests by the developer and how this testing should be documented. Finally, Independent testing (ATE\_IND) then addresses evaluator testing: whether the evaluator should repeat part or all of the developer testing and how much independent testing the evaluator should do.

Figure 12 shows the families within this class, and the hierarchy of components within the families.



**Figure 12 — ATE: Tests class decomposition**

## **13.2 Coverage (ATE\_COV)**

### **13.2.1 Objectives**

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

### **13.2.2 Component levelling**

The components in this family are levelled on the basis of specification.

### **13.2.3 Application notes**

#### **13.2.4 ATE\_COV.1 Evidence of coverage**

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

ATE\_FUN.1 Functional testing

##### **13.2.4.1 Objectives**

The objective of this component is to establish that some of the TSFIs have been tested.

##### **13.2.4.2 Application notes**

In this component the developer shows how tests in the test documentation correspond to TSFIs in the functional specification. This can be achieved by a statement of correspondence, perhaps using a table.

##### **13.2.4.3 Developer action elements**

###### **13.2.4.3.1 ATE\_COV.1.1D**

**The developer shall provide evidence of the test coverage.**

4625 **13.2.4.4 Content and presentation elements**

4626 **13.2.4.4.1 ATE\_COV.1.1C**

4627 **The evidence of the test coverage shall show the correspondence between the tests in the test**  
4628 **documentation and the TSFIs in the functional specification.**

4629 **13.2.4.5 Evaluator action elements**

4630 **13.2.4.5.1 ATE\_COV.1.1E**

4631 **The evaluator shall confirm that the information provided meets all requirements for content**  
4632 **and presentation of evidence.**

4633 **13.2.5 ATE\_COV.2 Analysis of coverage**

4634 Dependencies: ADV\_FSP.2 Security-enforcing functional specification

4635 ATE\_FUN.1 Functional testing

4636 **13.2.5.1 Objectives**

4637 The objective of this component is to confirm that all of the TSFIs have been tested.

4638 **13.2.5.2 Application notes**

4639 In this component the developer confirms that tests in the test documentation correspond to all of the  
4640 TSFIs in the functional specification. This can be achieved by a statement of correspondence, perhaps  
4641 using a table, but the developer also provides an analysis of the test coverage.

4642 **13.2.5.3 Developer action elements**

4643 **13.2.5.3.1 ATE\_COV.2.1D**

4644 The developer shall provide **an analysis** of the test coverage.

4645 **13.2.5.4 Content and presentation elements**

4646 **13.2.5.4.1 ATE\_COV.2.1C**

4647 The **analysis** of the test coverage shall **demonstrate** the correspondence between the tests in the test  
4648 documentation and the TSFIs in the functional specification.

4649 **13.2.5.4.2 ATE\_COV.2.2C**

4650 **The analysis of the test coverage shall demonstrate that all TSFIs in the functional specification**  
4651 **have been tested.**

4652 **13.2.5.5 Evaluator action elements**

4653 **13.2.5.5.1 ATE\_COV.2.1E**

4654 The evaluator shall confirm that the information provided meets all requirements for content and  
4655 presentation of evidence.



4656 **13.2.6 ATE\_COV.3 Rigorous analysis of coverage**

4657 Dependencies: ADV\_FSP.2 Security-enforcing functional specification

4658 ATE\_FUN.1 Functional testing

4659 **13.2.6.1 Objectives**

4660 In this component, the objective is to confirm that the developer performed exhaustive tests of all  
4661 interfaces in the functional specification.

4662 The objective of this component is to confirm that all parameters of all of the TSFIs have been tested.

4663 **13.2.6.2 Application notes**

4664 In this component the developer is required to show how tests in the test documentation correspond  
4665 to all of the TSFIs in the functional specification. This can be achieved by a statement of  
4666 correspondence, perhaps using a table, but in addition the developer is required to demonstrate that  
4667 the tests exercise all of the parameters of all TSFIs. This additional requirement includes bounds  
4668 testing (i.e. verifying that errors are generated when stated limits are exceeded) and negative testing  
4669 (e.g. when access is given to User A, verifying not only that User A now has access, but also that User B  
4670 did not suddenly gain access). This kind of testing is not, strictly speaking, *exhaustive* because not  
4671 every possible value of the parameters is expected to be checked.

4672 **13.2.6.3 Developer action elements**

4673 **13.2.6.3.1 ATE\_COV.3.1D**

4674 The developer shall provide an analysis of the test coverage.

4675 **13.2.6.4 Content and presentation elements**

4676 **13.2.6.4.1 ATE\_COV.3.1C**

4677 The analysis of the test coverage shall demonstrate the correspondence between the tests in the test  
4678 documentation and the TSFIs in the functional specification.

4679 **13.2.6.4.2 ATE\_COV.3.2C**

4680 The analysis of the test coverage shall demonstrate that all TSFIs in the functional specification have  
4681 been **completely** tested.

4682 **13.2.6.5 Evaluator action elements**

4683 **13.2.6.5.1 ATE\_COV.3.1E**

4684 The evaluator shall confirm that the information provided meets all requirements for content and  
4685 presentation of evidence.

4686 **13.3 Depth (ATE\_DPT)**

4687 **13.3.1 Objectives**

4688 The components in this family deal with the level of detail to which the TSF is tested by the developer.  
4689 Testing of the TSF is based upon increasing depth of information derived from additional design

4690 representations and descriptions (TOE design, implementation representation, and security  
4691 architecture description).

4692 The objective is to counter the risk of missing an error in the development of the TOE. Testing that  
4693 exercises specific internal interfaces can provide assurance not only that the TSF exhibits the desired  
4694 external security behaviour, but also that this behaviour stems from correctly operating internal  
4695 functionality.

### 4696 **13.3.2 Component levelling**

4697 The components in this family are levelled on the basis of increasing detail provided in the TSF  
4698 representations, from the TOE design to the implementation representation. This levelling reflects the  
4699 TSF representations presented in the ADV class.

### 4700 **13.3.3 Application notes**

4701 The TOE design describes the internal components (e.g. subsystems) and, perhaps, modules of the TSF,  
4702 together with a description of the interfaces among these components and modules. Evidence of  
4703 testing of this TOE design must show that the internal interfaces have been exercised and seen to  
4704 behave as described. This may be achieved through testing via the external interfaces of the TSF, or by  
4705 testing of the TOE subsystem or module interfaces in isolation, perhaps employing a test harness. In  
4706 cases where some aspects of an internal interface cannot be tested via the external interfaces, there  
4707 should either be justification that these aspects need not be tested, or the internal interface needs to  
4708 be tested directly. In the latter case the TOE design needs to be sufficiently detailed in order to  
4709 facilitate direct testing.

4710 In cases where the description of the TSF's architectural soundness (in Security Architecture  
4711 (ADV\_ARC)) cites specific mechanisms, the tests performed by the developer must show that the  
4712 mechanisms have been exercised and seen to behave as described.

4713 At the highest component of this family, the testing is performed not only against the TOE design, but  
4714 also against the implementation representation.

### 4715 **13.3.4 ATE\_DPT.1 Testing: basic design**

4716 Dependencies: ADV\_ARC.1 Security architecture description

4717 ADV\_TDS.2 Architectural design

4718 ATE\_FUN.1 Functional testing

#### 4719 **13.3.4.1 Objectives**

4720 The subsystem descriptions of the TSF provide a high-level description of the internal workings of the  
4721 TSF. Testing at the level of the TOE subsystems provides assurance that the TSF subsystems behave  
4722 and interact as described in the TOE design and the security architecture description.

#### 4723 **13.3.4.2 Developer action elements**

##### 4724 **13.3.4.2.1 ATE\_DPT.1.1D**

4725 **The developer shall provide the analysis of the depth of testing.**

4726 **13.3.4.3 Content and presentation elements**

4727 **13.3.4.3.1 ATE\_DPT.1.1C**

4728 **The analysis of the depth of testing shall demonstrate the correspondence between the tests in**  
 4729 **the test documentation and the TSF subsystems in the TOE design.**

4730 **13.3.4.3.2 ATE\_DPT.1.2C**

4731 **The analysis of the depth of testing shall demonstrate that all TSF subsystems in the TOE design**  
 4732 **have been tested.**

4733 **13.3.4.4 Evaluator action elements**

4734 **13.3.4.4.1 ATE\_DPT.1.1E**

4735 **The evaluator shall confirm that the information provided meets all requirements for content**  
 4736 **and presentation of evidence.**

4737 **13.3.5 ATE\_DPT.2 Testing: security enforcing modules**

4738 Dependencies: ADV\_ARC.1 Security architecture description

4739 ADV\_TDS.3 Basic modular design

4740 ATE\_FUN.1 Functional testing

4741 **13.3.5.1 Objectives**

4742 The subsystem and module descriptions of the TSF provide a high-level description of the internal  
 4743 workings, and a description of the interfaces of the SFR-enforcing modules, of the TSF. Testing at this  
 4744 level of TOE description provides assurance that the TSF subsystems and SFR-enforcing modules  
 4745 behave and interact as described in the TOE design and the security architecture description.

4746 **13.3.5.2 Developer action elements**

4747 **13.3.5.2.1 ATE\_DPT.2.1D**

4748 The developer shall provide the analysis of the depth of testing.

4749 **13.3.5.3 Content and presentation elements**

4750 **13.3.5.3.1 ATE\_DPT.2.1C**

4751 The analysis of the depth of testing shall demonstrate the correspondence between the tests in the test  
 4752 documentation and the TSF subsystems **and SFR-enforcing modules** in the TOE design.

4753 **13.3.5.3.2 ATE\_DPT.2.2C**

4754 The analysis of the depth of testing shall demonstrate that all TSF subsystems in the TOE design have  
 4755 been tested.

4756 **13.3.5.3.3 ATE\_DPT.2.3C**

4757 **The analysis of the depth of testing shall demonstrate that the SFR-enforcing modules in the**  
 4758 **TOE design have been tested.**

4759 **13.3.5.4 Evaluator action elements**

4760 **13.3.5.4.1 ATE\_DPT.2.1E**

4761 The evaluator shall confirm that the information provided meets all requirements for content and  
4762 presentation of evidence.

4763 **13.3.6 ATE\_DPT.3 Testing: modular design**

4764 Dependencies: ADV\_ARC.1 Security architecture description

4765 ADV\_TDS.4 Semiformal modular design

4766 ATE\_FUN.1 Functional testing

4767 **13.3.6.1 Objectives**

4768 The subsystem and module descriptions of the TSF provide a high-level description of the internal  
4769 workings, and a description of the interfaces of the modules, of the TSF. Testing at this level of TOE  
4770 description provides assurance that the TSF subsystems and modules behave and interact as  
4771 described in the TOE design and the security architecture description.

4772 **13.3.6.2 Developer action elements**

4773 **13.3.6.2.1 ATE\_DPT.3.1D**

4774 The developer shall provide the analysis of the depth of testing.

4775 **13.3.6.3 Content and presentation elements**

4776 **13.3.6.3.1 ATE\_DPT.3.1C**

4777 The analysis of the depth of testing shall demonstrate the correspondence between the tests in the test  
4778 documentation and the TSF subsystems and modules in the TOE design.

4779 **13.3.6.3.2 ATE\_DPT.3.2C**

4780 The analysis of the depth of testing shall demonstrate that all TSF subsystems in the TOE design have  
4781 been tested.

4782 **13.3.6.3.3 ATE\_DPT.3.3C**

4783 The analysis of the depth of testing shall demonstrate that **all TSF** modules in the TOE design have  
4784 been tested.

4785 **13.3.6.4 Evaluator action elements**

4786 **13.3.6.4.1 ATE\_DPT.3.1E**

4787 The evaluator shall confirm that the information provided meets all requirements for content and  
4788 presentation of evidence.

4789 **13.3.7 ATE\_DPT.4 Testing: implementation representation**

4790 Dependencies: ADV\_ARC.1 Security architecture description

4791 ADV\_TDS.4 Semiformal modular design

4792 ADV\_IMP.1 Implementation representation of the TSF

4793 ATE\_FUN.1 Functional testing

#### 4794 **13.3.7.1 Objectives**

4795 The subsystem and module descriptions of the TSF provide a high-level description of the internal  
4796 workings, and a description of the interfaces of the modules, of the TSF. Testing at this level of TOE  
4797 description provides assurance that the TSF subsystems and modules behave and interact as  
4798 described in the TOE design and the security architecture description, and in accordance with the  
4799 implementation representation.

#### 4800 **13.3.7.2 Developer action elements**

##### 4801 **13.3.7.2.1 ATE\_DPT.4.1D**

4802 The developer shall provide the analysis of the depth of testing.

#### 4803 **13.3.7.3 Content and presentation elements**

##### 4804 **13.3.7.3.1 ATE\_DPT.4.1C**

4805 The analysis of the depth of testing shall demonstrate the correspondence between the tests in the test  
4806 documentation and the TSF subsystems and modules in the TOE design.

##### 4807 **13.3.7.3.2 ATE\_DPT.4.2C**

4808 The analysis of the depth of testing shall demonstrate that all TSF subsystems in the TOE design have  
4809 been tested.

##### 4810 **13.3.7.3.3 ATE\_DPT.4.3C**

4811 The analysis of the depth of testing shall demonstrate that all modules in the TOE design have been  
4812 tested.

##### 4813 **13.3.7.3.4 ATE\_DPT.4.4C**

4814 **The analysis of the depth of testing shall demonstrate that the TSF operates in accordance with**  
4815 **its implementation representation.**

#### 4816 **13.3.7.4 Evaluator action elements**

##### 4817 **13.3.7.4.1 ATE\_DPT.4.1E**

4818 The evaluator shall confirm that the information provided meets all requirements for content and  
4819 presentation of evidence.

### 4820 **13.4 Functional tests (ATE\_FUN)**

#### 4821 **13.4.1 Objectives**

4822 Functional testing performed by the developer provides assurance that the tests in the test  
4823 documentation are performed and documented correctly. The correspondence of these tests to the

4824 design descriptions of the TSF is achieved through the Coverage (ATE\_COV) and Depth (ATE\_DPT)  
4825 families.

4826 This family contributes to providing assurance that the likelihood of undiscovered flaws is relatively  
4827 small.

4828 The families Coverage (ATE\_COV), Depth (ATE\_DPT) and Functional tests (ATE\_FUN) are used in  
4829 combination to define the evidence of testing to be supplied by a developer. Independent functional  
4830 testing by the evaluator is specified by Independent testing (ATE\_IND).

#### 4831 **13.4.2 Component levelling**

4832 This family contains two components, the higher requiring that ordering dependencies are analysed.

#### 4833 **13.4.3 Application notes**

4834 Procedures for performing tests are expected to provide instructions for using test programs and test  
4835 suites, including the test environment, test conditions, test data parameters and values. The test  
4836 procedures should also show how the test results are derived from the test inputs.

4837 Ordering dependencies are relevant when the successful execution of a particular test depends upon  
4838 the existence of a particular state. For example, this might require that test A be executed immediately  
4839 before test B, since the state resulting from the successful execution of test A is a prerequisite for the  
4840 successful execution of test B. Thus, failure of test B could be related to a problem with the ordering  
4841 dependencies. In the above example, test B could fail because test C (rather than test A) was executed  
4842 immediately before it, or the failure of test B could be related to a failure of test A.

#### 4843 **13.4.4 ATE\_FUN.1 Functional testing**

4844 Dependencies: ATE\_COV.1 Evidence of coverage

##### 4845 **13.4.4.1 Objectives**

4846 The objective is for the developer to demonstrate that the tests in the test documentation are  
4847 performed and documented correctly.

##### 4848 **13.4.4.2 Developer action elements**

###### 4849 **13.4.4.2.1 ATE\_FUN.1.1D**

4850 **The developer shall test the TSF and document the results.**

###### 4851 **13.4.4.2.2 ATE\_FUN.1.2D**

4852 **The developer shall provide test documentation.**

##### 4853 **13.4.4.3 Content and presentation elements**

###### 4854 **13.4.4.3.1 ATE\_FUN.1.1C**

4855 **The test documentation shall consist of test plans, expected test results and actual test results.**

4856 **13.4.4.3.2 ATE\_FUN.1.2C**

4857 **The test plans shall identify the tests to be performed and describe the scenarios for**  
 4858 **performing each test. These scenarios shall include any ordering dependencies on the results**  
 4859 **of other tests.**

4860 **13.4.4.3.3 ATE\_FUN.1.3C**

4861 **The expected test results shall show the anticipated outputs from a successful execution of the**  
 4862 **tests.**

4863 **13.4.4.3.4 ATE\_FUN.1.4C**

4864 **The actual test results shall be consistent with the expected test results.**

4865 **13.4.4.4 Evaluator action elements**

4866 **13.4.4.4.1 ATE\_FUN.1.1E**

4867 **The evaluator shall confirm that the information provided meets all requirements for content**  
 4868 **and presentation of evidence.**

4869 **13.4.5 ATE\_FUN.2 Ordered functional testing**

4870 Dependencies: ATE\_COV.1 Evidence of coverage

4871 **13.4.5.1 Objectives**

4872 The objectives are for the developer to demonstrate that the tests in the test documentation are  
 4873 performed and documented correctly, and to ensure that testing is structured such as to avoid circular  
 4874 arguments about the correctness of the interfaces being tested.

4875 **13.4.5.2 Application notes**

4876 Although the test procedures may state pre-requisite initial test conditions in terms of ordering of  
 4877 tests, they may not provide a rationale for the ordering. An analysis of test ordering is an important  
 4878 factor in determining the adequacy of testing, as there is a possibility of faults being concealed by the  
 4879 ordering of tests.

4880 **13.4.5.3 Developer action elements**

4881 **13.4.5.3.1 ATE\_FUN.2.1D**

4882 The developer shall test the TSF and document the results.

4883 **13.4.5.3.2 ATE\_FUN.2.2D**

4884 The developer shall provide test documentation.

4885 **13.4.5.4 Content and presentation elements**

4886 **13.4.5.4.1 ATE\_FUN.2.1C**

4887 The test documentation shall consist of test plans, expected test results and actual test results.

4888 **13.4.5.4.2 ATE\_FUN.2.2C**

4889 The test plans shall identify the tests to be performed and describe the scenarios for performing each  
4890 test. These scenarios shall include any ordering dependencies on the results of other tests.

4891 **13.4.5.4.3 ATE\_FUN.2.3C**

4892 The expected test results shall show the anticipated outputs from a successful execution of the tests.

4893 **13.4.5.4.4 ATE\_FUN.2.4C**

4894 The actual test results shall be consistent with the expected test results.

4895 **13.4.5.4.5 ATE\_FUN.2.5C**

4896 **The test documentation shall include an analysis of the test procedure ordering dependencies.**

4897 **13.4.5.5 Evaluator action elements**

4898 **13.4.5.5.1 ATE\_FUN.2.1E**

4899 The evaluator shall confirm that the information provided meets all requirements for content and  
4900 presentation of evidence.

4901 **13.5 Independent testing (ATE\_IND)**

4902 **13.5.1 Objectives**

4903 The objectives of this family are built upon the assurances achieved in the ATE\_FUN, ATE\_COV, and  
4904 ATE\_DPT families by verifying the developer testing and performing additional tests by the evaluator.

4905 **13.5.2 Component levelling**

4906 Levelling is based upon the amount of developer test documentation and test support and the amount  
4907 of evaluator testing.

4908 **13.5.3 Application notes**

4909 This family deals with the degree to which there is independent functional testing of the TSF.  
4910 Independent functional testing may take the form of repeating the developer's functional tests (in  
4911 whole or in part) or of extending the scope or the depth of the developer's tests. These activities are  
4912 complementary, and an appropriate mix must be planned for each TOE, which takes into account the  
4913 availability and coverage of test results, and the functional complexity of the TSF.

4914 Sampling of developer tests is intended to provide confirmation that the developer has carried out his  
4915 planned test programme on the TSF, and has correctly recorded the results. The size of sample  
4916 selected will be influenced by the detail and quality of the developer's functional test results. The  
4917 evaluator will also need to consider the scope for devising additional tests, and the relative benefit that  
4918 may be gained from effort in these two areas. It is recognised that repetition of all developer tests may  
4919 be feasible and desirable in some cases, but may be very arduous and less productive in others. The  
4920 highest component in this family should therefore be used with caution. Sampling will address the  
4921 whole range of test results available, including those supplied to meet the requirements of both  
4922 Coverage (ATE\_COV) and Depth (ATE\_DPT).



4923 There is also a need to consider the different configurations of the TOE that are included within the  
 4924 evaluation. The evaluator will need to assess the applicability of the results provided, and to plan his  
 4925 own testing accordingly.

4926 The suitability of the TOE for testing is based on the access to the TOE, and the supporting  
 4927 documentation and information required (including any test software or tools) to run tests. The need  
 4928 for such support is addressed by the dependencies to other assurance families.

4929 Additionally, suitability of the TOE for testing may be based on other considerations. For example, the  
 4930 version of the TOE submitted by the developer may not be the final version.

4931 The term *interfaces* refers to interfaces described in the functional specification and TOE design, and  
 4932 parameters passed through invocations identified in the implementation representation. The exact set  
 4933 of interfaces to be used is selected through Coverage (ATE\_COV) and the Depth (ATE\_DPT)  
 4934 components.

4935 References to a subset of the interfaces are intended to allow the evaluator to design an appropriate  
 4936 set of tests which is consistent with the objectives of the evaluation being conducted.

#### 4937 **13.5.4 ATE\_IND.1 Independent testing - conformance**

4938 Dependencies: ADV\_FSP.1 Basic functional specification

4939 AGD\_OPE.1 Operational user guidance

4940 AGD\_PRE.1 Preparative procedures

##### 4941 **13.5.4.1 Objectives**

4942 In this component, the objective is to demonstrate that the TOE operates in accordance with its design  
 4943 representations and guidance documents.

##### 4944 **13.5.4.2 Application notes**

4945 This component does not address the use of developer test results. It is applicable where such results  
 4946 are not available, and also in cases where the developer's testing is accepted without validation. The  
 4947 evaluator is required to devise and conduct tests with the objective of confirming that the TOE  
 4948 operates in accordance with its design representations, including but not limited to the functional  
 4949 specification. The approach is to gain confidence in correct operation through representative testing,  
 4950 rather than to conduct every possible test. The extent of testing to be planned for this purpose is a  
 4951 methodology issue, and needs to be considered in the context of a particular TOE and the balance of  
 4952 other evaluation activities.

##### 4953 **13.5.4.3 Developer action elements**

###### 4954 **13.5.4.3.1 ATE\_IND.1.1D**

4955 **The developer shall provide the TOE for testing.**

##### 4956 **13.5.4.4 Content and presentation elements**

###### 4957 **13.5.4.4.1 ATE\_IND.1.1C**

4958 **The TOE shall be suitable for testing.**

4959 **13.5.4.5 Evaluator action elements**

4960 **13.5.4.5.1 ATE\_IND.1.1E**

4961 **The evaluator shall confirm that the information provided meets all requirements for content**  
 4962 **and presentation of evidence.**

4963 **13.5.4.5.2 ATE\_IND.1.2E**

4964 **The evaluator shall test a subset of the TSF to confirm that the TSF operates as specified.**

4965 **13.5.5 ATE\_IND.2 Independent testing - sample**

4966 Dependencies: ADV\_FSP.2 Security-enforcing functional specification

4967 AGD\_OPE.1 Operational user guidance

4968 AGD\_PRE.1 Preparative procedures

4969 ATE\_COV.1 Evidence of coverage

4970 ATE\_FUN.1 Functional testing

4971 **13.5.5.1 Objectives**

4972 In this component, the objective is to demonstrate that the TOE operates in accordance with its design  
 4973 representations and guidance documents. Evaluator testing confirms that the developer performed  
 4974 some tests of some interfaces in the functional specification.

4975 **13.5.5.2 Application notes**

4976 The intent is that the developer should provide the evaluator with materials necessary for the efficient  
 4977 reproduction of developer tests. This may include such things as machine-readable test  
 4978 documentation, test programs, etc.

4979 This component contains a requirement that the evaluator has available test results from the  
 4980 developer to supplement the programme of testing. The evaluator will repeat a sample of the  
 4981 developer's tests to gain confidence in the results obtained. Having established such confidence the  
 4982 evaluator will build upon the developer's testing by conducting additional tests that exercise the TOE  
 4983 in a different manner. By using a platform of validated developer test results the evaluator is able to  
 4984 gain confidence that the TOE operates correctly in a wider range of conditions than would be possible  
 4985 purely using the developer's own efforts, given a fixed level of resource. Having gained confidence that  
 4986 the developer has tested the TOE, the evaluator will also have more freedom, where appropriate, to  
 4987 concentrate testing in areas where examination of documentation or specialist knowledge has raised  
 4988 particular concerns.

4989 **13.5.5.3 Developer action elements**

4990 **13.5.5.3.1 ATE\_IND.2.1D**

4991 The developer shall provide the TOE for testing.

- 4992 **13.5.5.4 Content and presentation elements**
- 4993 **13.5.5.4.1 ATE\_IND.2.1C**
- 4994 The TOE shall be suitable for testing.
- 4995 **13.5.5.4.2 ATE\_IND.2.2C**
- 4996 **The developer shall provide an equivalent set of resources to those that were used in the**  
 4997 **developer's functional testing of the TSF.**
- 4998 **13.5.5.5 Evaluator action elements**
- 4999 **13.5.5.5.1 ATE\_IND.2.1E**
- 5000 The evaluator shall confirm that the information provided meets all requirements for content and  
 5001 presentation of evidence.
- 5002 **13.5.5.5.2 ATE\_IND.2.2E**
- 5003 **The evaluator shall execute a sample of tests in the test documentation to verify the developer**  
 5004 **test results.**
- 5005 **13.5.5.5.3 ATE\_IND.2.3E**
- 5006 The evaluator shall test a subset of the TSF to confirm that the TSF operates as specified.
- 5007 **13.5.6 ATE\_IND.3 Independent testing - complete**
- 5008 Dependencies: ADV\_FSP.4 Complete functional specification
- 5009 AGD\_OPE.1 Operational user guidance
- 5010 AGD\_PRE.1 Preparative procedures
- 5011 ATE\_COV.1 Evidence of coverage
- 5012 ATE\_FUN.1 Functional testing
- 5013 **13.5.6.1 Objectives**
- 5014 In this component, the objective is to demonstrate that the TOE operates in accordance with its design  
 5015 representations and guidance documents. Evaluator testing includes repeating all of the developer  
 5016 tests.
- 5017 **13.5.6.2 Application notes**
- 5018 The intent is that the developer should provide the evaluator with materials necessary for the efficient  
 5019 reproduction of developer tests. This may include such things as machine-readable test  
 5020 documentation, test programs, etc.

5021 In this component the evaluator must repeat all of the developer's tests as part of the programme of  
 5022 testing. As in the previous component the evaluator will also conduct tests that aim to exercise the TSF  
 5023 in a different manner from that achieved by the developer. In cases where developer testing has been  
 5024 exhaustive, there may remain little scope for this.

### 5025 **13.5.6.3 Developer action elements**

#### 5026 **13.5.6.3.1 ATE\_IND.3.1D**

5027 The developer shall provide the TOE for testing.

### 5028 **13.5.6.4 Content and presentation elements**

#### 5029 **13.5.6.4.1 ATE\_IND.3.1C**

5030 The TOE shall be suitable for testing.

#### 5031 **13.5.6.4.2 ATE\_IND.3.2C**

5032 The developer shall provide an equivalent set of resources to those that were used in the developer's  
 5033 functional testing of the TSF.

### 5034 **13.5.6.5 Evaluator action elements**

#### 5035 **13.5.6.5.1 ATE\_IND.3.1E**

5036 The evaluator shall confirm that the information provided meets all requirements for content and  
 5037 presentation of evidence.

#### 5038 **13.5.6.5.2 ATE\_IND.3.2E**

5039 The evaluator shall execute **all** tests in the test documentation to verify the developer test results.

#### 5040 **13.5.6.5.3 ATE\_IND.3.3E**

5041 The evaluator shall test the TSF to confirm that the **entire** TSF operates as specified.

## 5042 **13.6 Composite functional testing (ATE\_COMP)**

### 5043 **13.6.1 Objectives**

5044 The objective of this family is to determine whether composite product as a whole exhibits the  
 5045 properties necessary to satisfy the functional requirements of its Security Target.

### 5046 **13.6.2 Application notes**

5047 A composite product can be tested by testing the components of it separately and by testing the  
 5048 integrated product. Separate testing means that the platform and the application are being tested  
 5049 independent of each other. A lot of tests of the platform may have been performed within the scope of  
 5050 its accomplished evaluation. The application may be tested on a simulator or an emulator, which  
 5051 represent a virtual machine.

5052 Integration testing means that the composite product is being tested as it is: the application is running  
 5053 on the platform.

5054 Behaviour of implementation of some SFRs can depend on properties of the underlying platform as  
 5055 well as of the application (e.g. correctness of the measures of the composite product to withstand a  
 5056 side channel attack or correctness of the implementation of tamper resistance against physical  
 5057 attacks). In such a case the SFR implementation shall be tested on the final composite product, but not  
 5058 on a simulator or an emulator.

5059 This activity focuses exclusively on testing of the composite product as a whole and represents merely  
 5060 partial efforts within the general test approach being covered by the assurance ATE. These integration  
 5061 tests shall be specified and performed, whereby the approach of the standard assurance families of  
 5062 the class ATE shall be applied.

5063 - A correct behaviour of the Platform-TSF being relevant for the Composite-ST (corresponding to the  
 5064 group *RP-SFR-SERV* and *RP-SFR-MECH* in the work unit ADV\_COMP.1-1 above), and- absence of  
 5065 exploitable vulnerabilities (sufficient effectiveness) in the context of the Platform-ST are confirmed by  
 5066 the valid Platform Certificate, cf. chapter D.3.

### 5067 **13.6.3 ATE\_COMP.1 Composite product functional testing**

5068 Dependencies: No dependencies

#### 5069 **13.6.3.1 Developer action elements**

##### 5070 **13.6.3.1.1 ATE\_COMP.1.1D**

5071 **The developer shall provide a set of tests as required by the assurance package chosen.**

##### 5072 **13.6.3.1.2 ATE\_COMP.1.2D**

5073 **The developer shall provide the composite TOE for testing.**

#### 5074 **13.6.3.2 Content and presentation elements**

##### 5075 **13.6.3.2.1 ATE\_COMP.1.1C**

5076 **Content and presentation of the specification and documentation of the *integration* tests shall**  
 5077 **correspond to the standard<sup>6</sup> requirements of the assurance families ATE\_FUN and ATE\_COV.**

##### 5078 **13.6.3.2.2 ATE\_COMP.1.2C**

5079 **The composite TOE provided shall be suitable for testing.**

#### 5080 **13.6.3.3 Evaluator action elements**

##### 5081 **13.6.3.3.1 ATE\_COMP.1.1E**

5082 **The evaluator shall confirm that the information provided meets all requirements for content**  
 5083 **and presentation of evidence.**

---

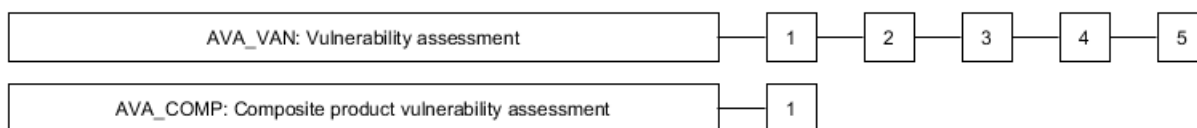
<sup>6</sup> i.e. as defined by CEM

## 5084 14 Class AVA: Vulnerability assessment

### 5085 14.1 Introduction

5086 The AVA: Vulnerability assessment class addresses the possibility of exploitable vulnerabilities  
5087 introduced in the development or the operation of the TOE.

5088 Figure 13 shows the families within this class, and the hierarchy of components within the families.



5089

5090 **Figure 13 — AVA: Vulnerability assessment class decomposition**

### 5091 14.2 Application notes

5092 Generally, the vulnerability assessment activity covers various vulnerabilities in the development and  
5093 operation of the TOE. Development vulnerabilities take advantage of some property of the TOE, or the  
5094 product where the TOE resides, which was introduced during its development, e.g. defeating the TSF  
5095 self-protection through tampering, direct attack or monitoring of the TSF, defeating the TSF domain  
5096 separation through monitoring or direct attack the TSF, or defeating non-bypassability through  
5097 circumventing (bypassing) the TSF. Explicit dependencies of the TOE on IT systems in the  
5098 environment must also be considered. Operational vulnerabilities take advantage of weaknesses in  
5099 non-technical countermeasures to violate the TOE SFRs, e.g. misuse or incorrect configuration. Misuse  
5100 investigates whether the TOE can be configured or used in a manner that is insecure, but that an  
5101 administrator or user of the TOE would reasonably believe to be secure.

5102 Assessment of development vulnerabilities is covered by the assurance family AVA\_VAN. Basically, all  
5103 development vulnerabilities can be considered in the context of AVA\_VAN due to the fact, that this  
5104 family allows application of a wide range of assessment methodologies being unspecific to the kind of  
5105 an attack scenario. These unspecific assessment methodologies comprise, among other, also the  
5106 specific methodologies for those TSF where covert channels are to be considered (a channel capacity  
5107 estimation can be done using informal engineering measurements, as well as actual test  
5108 measurements) or can be overcome by the use of sufficient resources in the form of a direct attack  
5109 (underlying technical concept of those TSF is based on probabilistic or permutational mechanisms; a  
5110 qualification of their security behaviour and the effort required to overcome them can be made using a  
5111 quantitative or statistical analysis).

5112 If there are security objectives specified in the ST to either to prevent one user of the TOE from  
5113 observing activity associated with another user of the TOE, or to ensure that information flows cannot  
5114 be used to achieve enforced illicit data signals, covert channel analysis should be considered during  
5115 the conduct of the vulnerability analysis. This is often reflected by the inclusion of Unobservability  
5116 (FPR\_UNO) and multilevel access control policies specified through Access control policy (FDP\_ACC)  
5117 and/or Information flow control policy (FDP\_IFC) requirements in the ST.

## 5118 **14.3 Vulnerability analysis (AVA\_VAN)**

### 5119 **14.3.1 Objectives**

5120 Vulnerability analysis is an assessment to determine whether potential vulnerabilities identified,  
5121 during the evaluation of the development and anticipated operation of the TOE or by other methods  
5122 (e.g. by flaw hypotheses or quantitative or statistical analysis of the security behaviour of the  
5123 underlying security mechanisms), could allow attackers to violate the SFRs.

5124 Vulnerability analysis deals with the threats that an attacker will be able to discover flaws that will  
5125 allow unauthorised access to data and functionality, allow the ability to interfere with or alter the TSF,  
5126 or interfere with the authorised capabilities of other users.

### 5127 **14.3.2 Component levelling**

5128 Levelling is based on an increasing rigour of vulnerability analysis by the evaluator and increased  
5129 levels of attack potential required by an attacker to identify and exploit the potential vulnerabilities.

### 5130 **14.3.3 AVA\_VAN.1 Vulnerability survey**

5131 Dependencies: ADV\_FSP.1 Basic functional specification

5132 AGD\_OPE.1 Operational user guidance

5133 AGD\_PRE.1 Preparative procedures

#### 5134 **14.3.3.1 Objectives**

5135 A vulnerability survey of information available in the public domain is performed by the evaluator to  
5136 ascertain potential vulnerabilities that may be easily found by an attacker.

5137 The evaluator performs penetration testing, to confirm that the potential vulnerabilities cannot be  
5138 exploited in the operational environment for the TOE. Penetration testing is performed by the  
5139 evaluator assuming an attack potential of Basic.

#### 5140 **14.3.3.2 Developer action elements**

##### 5141 **14.3.3.2.1 AVA\_VAN.1.1D**

5142 **The developer shall provide the TOE for testing.**

#### 5143 **14.3.3.3 Content and presentation elements**

##### 5144 **14.3.3.3.1 AVA\_VAN.1.1C**

5145 **The TOE shall be suitable for testing.**

#### 5146 **14.3.3.4 Evaluator action elements**

##### 5147 **14.3.3.4.1 AVA\_VAN.1.1E**

5148 **The evaluator shall confirm that the information provided meets all requirements for content**  
5149 **and presentation of evidence.**

5150 **14.3.3.4.2 AVA\_VAN.1.2E**

5151 **The evaluator shall perform a search of public domain sources to identify potential**  
 5152 **vulnerabilities in the TOE.**

5153 **14.3.3.4.3 AVA\_VAN.1.3E**

5154 **The evaluator shall conduct penetration testing, based on the identified potential**  
 5155 **vulnerabilities, to determine that the TOE is resistant to attacks performed by an attacker**  
 5156 **possessing Basic attack potential.**

5157 **14.3.4 AVA\_VAN.2 Vulnerability analysis**

5158 Dependencies: ADV\_ARC.1 Security architecture description

5159 ADV\_FSP.2 Security-enforcing functional specification

5160 ADV\_TDS.1 Basic design

5161 AGD\_OPE.1 Operational user guidance

5162 AGD\_PRE.1 Preparative procedures

5163 **14.3.4.1 Objectives**

5164 A vulnerability analysis is performed by the evaluator to ascertain the presence of potential  
 5165 vulnerabilities.

5166 The evaluator performs penetration testing, to confirm that the potential vulnerabilities cannot be  
 5167 exploited in the operational environment for the TOE. Penetration testing is performed by the  
 5168 evaluator assuming an attack potential of Basic.

5169 **14.3.4.2 Developer action elements**

5170 **14.3.4.2.1 AVA\_VAN.2.1D**

5171 The developer shall provide the TOE for testing.

5172 **14.3.4.2.2 AVA\_VAN.2.2D**

5173 **The developer shall provide a list of third party components included in the TOE and the TOE**  
 5174 **delivery.**

5175 **14.3.4.3 Content and presentation elements**

5176 **14.3.4.3.1 AVA\_VAN.2.1C**

5177 The TOE shall be suitable for testing.

5178 **14.3.4.3.2 AVA\_VAN.2.2C**

5179 **The list of third party components shall include components provided by third parties, and that**  
 5180 **are part of the TOE or otherwise part of the TOE delivery.**



5181 **14.3.4.4 Evaluator action elements**

5182 **14.3.4.4.1 AVA\_VAN.2.1E**

5183 The evaluator shall confirm that the information provided meets all requirements for content and  
5184 presentation of evidence.

5185 **14.3.4.4.2 AVA\_VAN.2.2E**

5186 The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in  
5187 the TOE **the components in the list of third party components, and specific IT products in the**  
5188 **environment that the TOE depends on.**

5189 **14.3.4.4.3 AVA\_VAN.2.3E**

5190 **The evaluator shall perform an independent vulnerability analysis of the TOE using the**  
5191 **guidance documentation, functional specification, TOE design and security architecture**  
5192 **description to identify potential vulnerabilities in the TOE.**

5193 **14.3.4.4.4 AVA\_VAN.2.4E**

5194 The evaluator shall conduct penetration testing, based on the identified potential vulnerabilities, to  
5195 determine that the TOE is resistant to attacks performed by an attacker possessing Basic attack  
5196 potential.

5197 **14.3.5 AVA\_VAN.3 Focused vulnerability analysis**

5198 Dependencies: ADV\_ARC.1 Security architecture description

5199 ADV\_FSP.4 Complete functional specification

5200 ADV\_TDS.3 Basic modular design

5201 ADV\_IMP.1 Implementation representation of the TSF

5202 AGD\_OPE.1 Operational user guidance

5203 AGD\_PRE.1 Preparative procedures

5204 ATE\_DPT.1 Testing: basic design

5205 **14.3.5.1 Objectives**

5206 A vulnerability analysis is performed by the evaluator to ascertain the presence of potential  
5207 vulnerabilities.

5208 The evaluator performs penetration testing, to confirm that the potential vulnerabilities cannot be  
5209 exploited in the operational environment for the TOE. Penetration testing is performed by the  
5210 evaluator assuming an attack potential of Enhanced-Basic.

5211 **14.3.5.2 Developer action elements**

5212 **14.3.5.2.1 AVA\_VAN.3.1D**

5213 The developer shall provide the TOE for testing.

5214 **14.3.5.2.2 AVA\_VAN.3.2D**

5215 The developer shall provide a list of third party components included in the TOE and the TOE delivery.

5216 **14.3.5.3 Content and presentation elements**

5217 **14.3.5.3.1 AVA\_VAN.3.1C**

5218 The TOE shall be suitable for testing.

5219 **14.3.5.3.2 AVA\_VAN.3.2C**

5220 The list of third party components shall include components provided by third parties, and that are  
5221 part of the TOE or otherwise part of the TOE delivery.

5222 **14.3.5.4 Evaluator action elements**

5223 **14.3.5.4.1 AVA\_VAN.3.1E**

5224 The evaluator shall confirm that the information provided meets all requirements for content and  
5225 presentation of evidence.

5226 **14.3.5.4.2 AVA\_VAN.3.2E**

5227 The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in  
5228 the TOE the components in the list of third party components, and specific IT products in the  
5229 environment that the TOE depends on.

5230 **14.3.5.4.3 AVA\_VAN.3.3E**

5231 The evaluator shall perform an independent, **focused** vulnerability analysis of the TOE using the  
5232 guidance documentation, functional specification, TOE design, security architecture description **and**  
5233 **implementation representation** to identify potential vulnerabilities in the TOE.

5234 **14.3.5.4.4 AVA\_VAN.3.4E**

5235 The evaluator shall conduct penetration testing, based on the identified potential vulnerabilities, to  
5236 determine that the TOE is resistant to attacks performed by an attacker possessing **Enhanced-Basic**  
5237 attack potential.

5238 **14.3.6 AVA\_VAN.4 Methodical vulnerability analysis**

5239 Dependencies: ADV\_ARC.1 Security architecture description

5240 ADV\_FSP.4 Complete functional specification

5241 ADV\_TDS.3 Basic modular design

5242 ADV\_IMP.1 Implementation representation of the TSF

5243 AGD\_OPE.1 Operational user guidance

5244 AGD\_PRE.1 Preparative procedures

5245 ATE\_DPT.1 Testing: basic design

5246 **14.3.6.1 Objectives**

5247 A methodical vulnerability analysis is performed by the evaluator to ascertain the presence of  
5248 potential vulnerabilities.

5249 The evaluator performs penetration testing, to confirm that the potential vulnerabilities cannot be  
5250 exploited in the operational environment for the TOE. Penetration testing is performed by the  
5251 evaluator assuming an attack potential of Moderate.

5252 **14.3.6.2 Developer action elements**

5253 **14.3.6.2.1 AVA\_VAN.4.1D**

5254 The developer shall provide the TOE for testing.

5255 **14.3.6.2.2 AVA\_VAN.4.2D**

5256 The developer shall provide a list of third party components included in the TOE and the TOE delivery.

5257 **14.3.6.3 Content and presentation elements**

5258 **14.3.6.3.1 AVA\_VAN.4.1C**

5259 The TOE shall be suitable for testing.

5260 **14.3.6.3.2 AVA\_VAN.4.2C**

5261 The list of third party components shall include components provided by third parties, and that are  
5262 part of the TOE or otherwise part of the TOE delivery.

5263 **14.3.6.4 Evaluator action elements**

5264 **14.3.6.4.1 AVA\_VAN.4.1E**

5265 The evaluator shall confirm that the information provided meets all requirements for content and  
5266 presentation of evidence.

5267 **14.3.6.4.2 AVA\_VAN.4.2E**

5268 The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in  
5269 the TOE the components in the list of third party components, and specific IT products in the  
5270 environment that the TOE depends on.

5271 **14.3.6.4.3 AVA\_VAN.4.3E**

5272 The evaluator shall perform an independent, **methodical** vulnerability analysis of the TOE using the  
5273 guidance documentation, functional specification, TOE design, security architecture description and  
5274 implementation representation to identify potential vulnerabilities in the TOE.

5275 **14.3.6.4.4 AVA\_VAN.4.4E**

5276 The evaluator shall conduct penetration testing based on the identified potential vulnerabilities to  
5277 determine that the TOE is resistant to attacks performed by an attacker possessing **Moderate** attack  
5278 potential.

5279 **14.3.7 AVA\_VAN.5 Advanced methodical vulnerability analysis**

5280 Dependencies: ADV\_ARC.1 Security architecture description

5281 ADV\_FSP.4 Complete functional specification

5282 ADV\_TDS.3 Basic modular design

5283 ADV\_IMP.1 Implementation representation of the TSF

5284 AGD\_OPE.1 Operational user guidance

5285 AGD\_PRE.1 Preparative procedures

5286 ATE\_DPT.1 Testing: basic design

5287 **14.3.7.1 Objectives**

5288 A methodical vulnerability analysis is performed by the evaluator to ascertain the presence of  
5289 potential vulnerabilities.

5290 The evaluator performs penetration testing, to confirm that the potential vulnerabilities cannot be  
5291 exploited in the operational environment for the TOE. Penetration testing is performed by the  
5292 evaluator assuming an attack potential of High.

5293 **14.3.7.2 Developer action elements**

5294 **14.3.7.2.1 AVA\_VAN.5.1D**

5295 The developer shall provide the TOE for testing.

5296 **14.3.7.2.2 AVA\_VAN.5.2D**

5297 The developer shall provide a list of third party components included in the TOE and the TOE delivery.

5298 **14.3.7.3 Content and presentation elements**

5299 **14.3.7.3.1 AVA\_VAN.5.1C**

5300 The TOE shall be suitable for testing.

5301 **14.3.7.3.2 AVA\_VAN.2.2C**

5302 The list of third party components shall include components provided by third parties, and that are  
5303 part of the TOE or otherwise part of the TOE delivery.

5304 **14.3.7.4 Evaluator action elements**

5305 **14.3.7.4.1 AVA\_VAN.5.1E**

5306 The evaluator shall confirm that the information provided meets all requirements for content and  
5307 presentation of evidence.

5308 **14.3.7.4.2 AVA\_VAN.5.2E**

5309 The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in  
 5310 the TOE the components in the list of third party components, and specific IT products in the  
 5311 environment that the TOE depends on.

5312 **14.3.7.4.3 AVA\_VAN.5.3E**

5313 The evaluator shall perform an independent, methodical vulnerability analysis of the TOE using the  
 5314 guidance documentation, functional specification, TOE design, security architecture description and  
 5315 implementation representation to identify potential vulnerabilities in the TOE.

5316 **14.3.7.4.4 AVA\_VAN.5.4E**

5317 The evaluator shall conduct penetration testing based on the identified potential vulnerabilities to  
 5318 determine that the TOE is resistant to attacks performed by an attacker possessing **High** attack  
 5319 potential.

5320 **14.4 Composite vulnerability assessment (AVA\_COMP)**

5321 **14.4.1 Objectives**

5322 The aim of this activity is to determine the exploitability of flaws or weaknesses in the composite TOE  
 5323 as a whole in the intended environment.

5324 **14.4.2 AVA\_COMP.1 Composite product vulnerability assessment**

5325 Dependencies: No dependencies

5326 **14.4.2.1 Application notes**

5327 This activity focuses exclusively on vulnerability assessment of the composite product *as a whole* and  
 5328 represents merely *partial efforts* within the general approach being covered by the standard<sup>7</sup>  
 5329 assurance family of the class AVA: AVA\_VAN.

5330 The results of the vulnerability assessment for the underlying platform represented in the ETR\_COMP  
 5331 can be reused under the following conditions: they are up to date and all composite activities for  
 5332 correctness – ASE\_COMP.1, ALC\_COMP.1, ADV\_COMP.1 and ATE\_COMP.1 – are finalised with the  
 5333 verdict PASS.

5334 Due to composing of the platform and the application a new quality arises, which can cause additional  
 5335 vulnerabilities of the platform which might be not mentioned in the ETR\_COMP. In these  
 5336 circumstances [R44] in chapter D.3 applies.

5337 **14.4.2.2 Developer action elements**

5338 **14.4.2.2.1 AVA\_COMP.1.1D**

5339 The developer shall provide the composite TOE for penetrating testing.

---

<sup>7</sup> i.e. as defined by CEM

5340 **14.4.2.3 Content and presentation elements**

5341 **14.4.2.3.1 AVA\_COMP.1.1C**

5342 The composite TOE provided shall be suitable for testing as a whole.

5343 **14.4.2.4 Evaluator action elements**

5344 **14.4.2.4.1 AVA\_COMP.1.1E**

5345 The evaluator shall conduct penetration testing of the composite product *as a whole* building on  
 5346 evaluator's own vulnerability analysis, to ensure that the vulnerabilities being relevant for the  
 5347 Composite-ST are not exploitable.

5348 **15 Class ACO: Composition**

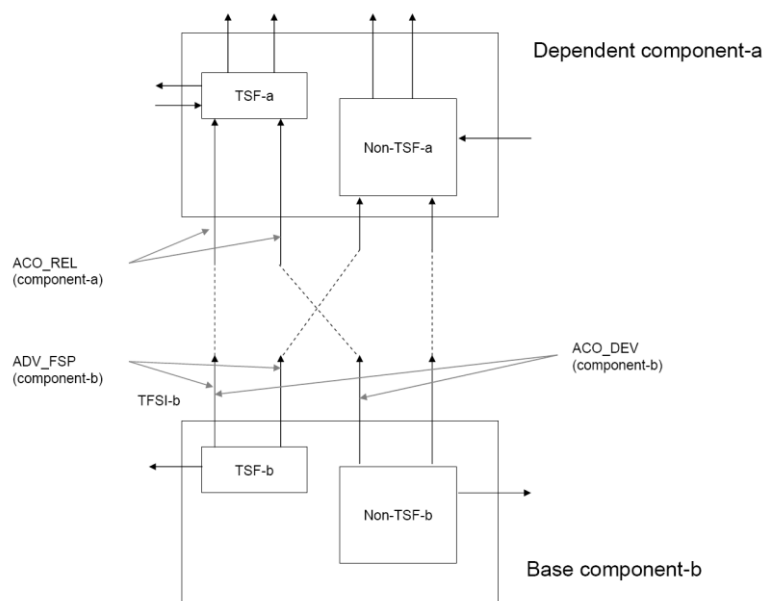
5349 **15.1 Introduction**

5350 The class ACO: Composition encompasses five families. These families specify assurance requirements  
 5351 that are designed to provide confidence that a composed TOE will operate securely when relying upon  
 5352 security functionality provided by previously evaluated software, firmware or hardware components.

5353 Composition involves taking two or more IT entities successfully evaluated against ISO/IEC 15408  
 5354 security assurance requirements packages (base components and dependent components, see  
 5355 Annex B) and combining them for use, with no further development of either IT entity. The  
 5356 development of additional IT entities is not included (entities that have not previously been the  
 5357 subject of a component evaluation). The composed TOE forms a new product that can be installed and  
 5358 integrated into any specific environment instance that meets the objectives for the environment.

5359 This approach does not provide an alternative approach for the evaluation of components.  
 5360 Composition under ACO provides a composed TOE integrator a method, which can be used as an  
 5361 alternative to other assurance levels specified in ISO/IEC 15408, to gain confidence in a TOE that is the  
 5362 combination of two or more successfully evaluated components without having to re-evaluate the  
 5363 composite TSF. (The composed TOE integrator is referred to as "developer" throughout the ACO class,  
 5364 with any references to the developer of the base or dependent components clarified as such.)

5365 Composed Assurance Packages, as defined in part 5 provide an assurance scale for composed TOEs.  
 5366 This assurance scale is required in addition to other assurance packages, for example the EALs,  
 5367 because to combine components evaluated against another assurance package and gain equivalent  
 5368 assurance in the resulting composed TOE, all SARs have to be applied to the composed TOE. Although  
 5369 reuse can be made of the component TOE evaluation results, there are often additional aspects of the  
 5370 components that have to be considered in the composed TOE, as described in Annex B.3. Due to the  
 5371 different parties involved in a composed TOE evaluation activity it is generally not possible to gain all  
 5372 necessary evidence about these additional aspects of the components to apply the appropriate EAL.  
 5373 Hence, CAPs have been defined to address the issue of combining evaluated components and gaining a  
 5374 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.

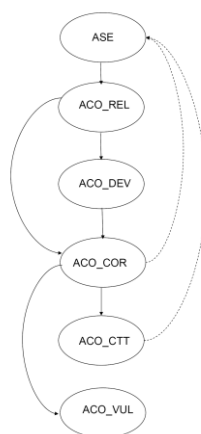
The families within the ACO class interact in a similar manner to the ADV, ATE and AVA classes in a component TOE evaluation and hence leverage from the specification of requirements from those classes where applicable. There are however a few items specific to composed TOE evaluations. To determine 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 configuration and identifies where any specifications are incomplete, which are then identified as inputs into testing (ACO\_CTT) and vulnerability analysis (ACO\_VUL) activities of the composed TOE.

5404 Testing of the composed TOE is performed to determine that the composed TOE exhibits the expected  
 5405 behaviour as determined by the composed TOE SFRs, and at higher levels demonstrates the  
 5406 compatibility of the interfaces between the components of the composed TOE.

5407 The vulnerability analysis of the composed TOE leverages from the outputs of the vulnerability  
 5408 analysis of the component evaluations. The composed TOE vulnerability analysis considers any  
 5409 residual vulnerabilities from the component evaluations to determine that the residual vulnerabilities  
 5410 are not applicable to the composed TOE. A search of publicly available information relating to the  
 5411 components is also performed to identify any issues reported in the components since the completion  
 5412 of the respective evaluations.

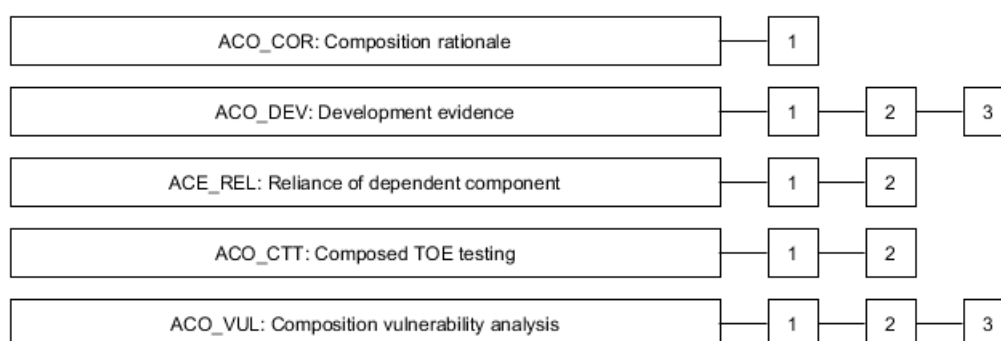
5413 The interaction between the ACO families is depicted in Figure 15 below. This shows by solid arrowed  
 5414 lines where the evidence and understanding gained in one family feeds into the next activity and the  
 5415 dashed arrows identify where an activity explicitly traces back to the composed TOE SFRs, as  
 5416 described above.



5417  
 5418 **Figure 15 — Relationship between ACO families**

5419 Further discussion of the definition and interactions within composed TOEs is provided in Annex B.

5420 Figure 16 shows the families within this class, and the hierarchy of components within the families.



5421  
 5422 **Figure 16 — ACO: Composition class decomposition**



5423 **15.2 Composition rationale (ACO\_COR)**

5424 **15.2.1 Objectives**

5425 This family addresses the requirement to demonstrate that the base component can provide an  
5426 appropriate level of assurance for use in composition.

5427 **15.2.2 Component levelling**

5428 There is only a single component in this family.

5429 **15.2.3 ACO\_COR.1 Composition rationale**

5430 Dependencies: ACO\_DEV.1 Functional Description

5431 ALC\_CMC.1 Labelling of the TOE

5432 ACO\_REL.1 Basic reliance information

5433 **15.2.3.1 Developer action elements**

5434 **15.2.3.1.1 ACO\_COR.1.1D**

5435 **The developer shall provide composition rationale for the base component.**

5436 **15.2.3.2 Content and presentation elements**

5437 **15.2.3.2.1 ACO\_COR.1.1C**

5438 **The composition rationale shall demonstrate that a level of assurance at least as high as that of**  
5439 **the dependent component has been obtained for the support functionality of the base**  
5440 **component, when the base component is configured as required to support the TSF of the**  
5441 **dependent component.**

5442 **15.2.3.3 Evaluator action elements**

5443 **15.2.3.3.1 ACO\_COR.1.1E**

5444 **The evaluator shall confirm that the information meets all requirements for content and**  
5445 **presentation of evidence.**

5446 **15.3 Development evidence (ACO\_DEV)**

5447 **15.3.1 Objectives**

5448 This family sets out requirements for a specification of the base component in increasing levels of  
5449 detail. Such information is required to gain confidence that the appropriate security functionality is  
5450 provided to support the requirements of the dependent component (as identified in the reliance  
5451 information).

5452 **15.3.2 Component levelling**

5453 The components are levelled on the basis of increasing amounts of detail about the interfaces  
5454 provided, and how they are implemented.

### 5455 **15.3.3 Application notes**

5456 The TSF of the base component is often defined without knowledge of the dependencies of the  
5457 possible applications with which it may be composed. The TSF of this base component is defined to  
5458 include all parts of the base component that have to be relied upon for enforcement of the base  
5459 component SFRs. This will include all parts of the base component required to implement the base  
5460 component SFRs.

5461 The functional specification of the base component will describe the TSFI in terms of the interfaces the  
5462 base component provides to allow an external entity to invoke operations of the TSF. This includes  
5463 interfaces to the human user to permit interaction with the operation of the TSF invoking SFRs and  
5464 also interfaces allowing an external IT entity to make calls into the TSF.

5465 The functional specification only provides a description of what the TSF provides at its interface and  
5466 the means by which that TSF functionality are invoked. Therefore, the functional specification does not  
5467 necessarily provide a complete interface specification of all possible interfaces available between an  
5468 external entity and the base component. It does not include what the TSF expects/requires from the  
5469 operational environment. The description of what a dependent component TSF relies upon of a base  
5470 component is considered in Reliance of dependent component (ACO\_REL) and the development  
5471 information evidence provides a response to the interfaces specified.

5472 The development information evidence includes a specification of the base component. This may be  
5473 the evidence used during evaluation of the base component to satisfy the ADV requirements, or may  
5474 be another form of evidence produced by either the base component developer or the composed TOE  
5475 developer. This specification of the base component is used during Development evidence (ACO\_DEV)  
5476 to gain confidence that the appropriate security functionality is provided to support the requirements  
5477 of the dependent component. The level of detail required of this evidence increases to reflect the level  
5478 of required assurance in the composed TOE. This is expected to broadly reflect the increasing  
5479 confidence gained from the application of the assurance packages to the components. The evaluator  
5480 determines that this description of the base component is consistent with the reliance information  
5481 provided for the dependent component.

### 5482 **15.3.4 ACO\_DEV.1 Functional Description**

5483 Dependencies: ACO\_REL.1 Basic reliance information

#### 5484 **15.3.4.1 Objectives**

5485 A description of the interfaces in the base component, on which the dependent component relies, is  
5486 required. This is examined to determine whether or not it is consistent with the description of  
5487 interfaces on which the dependent component relies, as provided in the reliance information.

#### 5488 **15.3.4.2 Developer action elements**

##### 5489 **15.3.4.2.1 ACO\_DEV.1.1D**

5490 **The developer shall provide development information for the base component.**

#### 5491 **15.3.4.3 Content and presentation elements**

##### 5492 **15.3.4.3.1 ACO\_DEV.1.1C**

5493 **The development information shall describe the purpose of each interface of the base**  
5494 **component used in the composed TOE.**

5495 **15.3.4.3.2 ACO\_DEV.1.2C**

5496 **The development information shall show correspondence between the interfaces, used in the**  
 5497 **composed TOE, of the base component and the dependent component to support the TSF of the**  
 5498 **dependent component.**

5499 **15.3.4.4 Evaluator action elements**

5500 **15.3.4.4.1 ACO\_DEV.1.1E**

5501 **The evaluator shall confirm that the information meets all requirements for content and**  
 5502 **presentation of evidence.**

5503 **15.3.4.4.2 ACO\_DEV.1.2E**

5504 **The evaluator shall determine that the interface description provided is consistent with the**  
 5505 **reliance information provided for the dependent component.**

5506 **15.3.5 ACO\_DEV.2 Basic evidence of design**

5507 Dependencies: ACO\_REL.1 Basic reliance information

5508 **15.3.5.1 Objectives**

5509 A description of the interfaces in the base component, on which the dependent component relies, is  
 5510 required. This is examined to determine whether or not it is consistent with the description of  
 5511 interfaces on which the dependent component relies, as provided in the reliance information.

5512 In addition, the security behaviour of the base component that supports the dependent component  
 5513 TSF is described.

5514 **15.3.5.2 Developer action elements**

5515 **15.3.5.2.1 ACO\_DEV.2.1D**

5516 The developer shall provide development information for the base component.

5517 **15.3.5.3 Content and presentation elements**

5518 **15.3.5.3.1 ACO\_DEV.2.1C**

5519 The development information shall describe the purpose **and method of use of** each interface of the  
 5520 base component used in the composed TOE.

5521 **15.3.5.3.2 ACO\_DEV.2.2C**

5522 **The development information shall provide a high-level description of the behaviour of the**  
 5523 **base component, which supports the enforcement of the dependent component SFRs.**

5524 **15.3.5.3.3 ACO\_DEV.2.3C**

5525 The development information shall show correspondence between the interfaces, used in the  
 5526 composed TOE, of the base component and the dependent component to support the TSF of the  
 5527 dependent component.

5528 **15.3.5.4 Evaluator action elements**

5529 **15.3.5.4.1 ACO\_DEV.2.1E**

5530 The evaluator shall confirm that the information meets all requirements for content and presentation  
5531 of evidence.

5532 **15.3.5.4.2 ACO\_DEV.2.2E**

5533 The evaluator shall determine that the interface description provided is consistent with the reliance  
5534 information provided for the dependent component.

5535 **15.3.6 ACO\_DEV.3 Detailed evidence of design**

5536 Dependencies: ACO\_REL.2 Reliance information

5537 **15.3.6.1 Objectives**

5538 A description of the interfaces in the base component, on which the dependent component relies, is  
5539 required. This is examined to determine whether or not it is consistent with the description of  
5540 interfaces on which the dependent component relies, as provided in the reliance information.

5541 The interface description of the architecture of the base component is provided to enable the evaluator  
5542 to determine whether or not that interface formed part of the TSF of the base component.

5543 **15.3.6.2 Developer action elements**

5544 **15.3.6.2.1 ACO\_DEV.3.1D**

5545 The developer shall provide development information for the base component.

5546 **15.3.6.3 Content and presentation elements**

5547 **15.3.6.3.1 ACO\_DEV.3.1C**

5548 The development information shall describe the purpose and method of use of each interface of the  
5549 base component used in the composed TOE.

5550 **15.3.6.3.2 ACO\_DEV.3.2C**

5551 **The development information shall identify the subsystems of the base component that**  
5552 **provide interfaces of the base component used in the composed TOE.**

5553 **15.3.6.3.3 ACO\_DEV.3.3C**

5554 The development information shall provide a high-level description of the behaviour of the base  
5555 component **subsystems**, which **support** the enforcement of the dependent component SFRs.

5556 **15.3.6.3.4 ACO\_DEV.3.4C**

5557 **The development information shall provide a mapping from the interfaces to the subsystems of**  
5558 **the base component.**

### 5559 **15.3.6.3.5 ACO\_DEV.3.5C**

5560 The development information shall show correspondence between the interfaces, used in the  
5561 composed TOE, of the base component and the dependent component to support the TSF of the  
5562 dependent component.

### 5563 **15.3.6.4 Evaluator action elements**

#### 5564 **15.3.6.4.1 ACO\_DEV.3.1E**

5565 The evaluator shall confirm that the information meets all requirements for content and presentation  
5566 of evidence.

#### 5567 **15.3.6.4.2 ACO\_DEV.3.2E**

5568 The evaluator shall determine that the interface description provided is consistent with the reliance  
5569 information provided for the dependent component.

## 5570 **15.4 Reliance of dependent component (ACO\_REL)**

### 5571 **15.4.1 Objectives**

5572 The purpose of this family is to provide evidence that describes the reliance that a dependent  
5573 component has upon the base component. This information is useful to persons responsible for  
5574 integrating the component with other evaluated IT components to form the composed TOE, and for  
5575 providing insight into the security properties of the resulting composition.

5576 This provides a description of the interface between the dependent and base components of the  
5577 composed TOE that may not have been analysed during evaluation of the individual components, as  
5578 the interfaces were not TSFIs of the individual component TOEs.

### 5579 **15.4.2 Component levelling**

5580 The components in this family are levelled according to the amount of detail provided in the  
5581 description of the reliance by the dependent component upon the base component.

### 5582 **15.4.3 Application notes**

5583 The Reliance of dependent component (ACO\_REL) family considers the interactions between the  
5584 components where the dependent component relies upon a service from the base component to  
5585 support the operation of security functionality of the dependent component. The interfaces into these  
5586 services of the base component may not have been considered during evaluation of the base  
5587 component because the service in the base component was not considered security-relevant in the  
5588 component evaluation, either because of the inherent purpose of the service (e.g., adjust type font) or  
5589 because associated ISO/IEC 15408 SFRs are not being claimed in the base component's ST (e.g. the  
5590 login interface when no FIA: Identification and authentication SFRs are claimed). These interfaces into  
5591 the base component are often viewed as functional interfaces in the evaluation of the base component,  
5592 and are in addition to the security interfaces (TSFI) considered in the functional specification.

5593 In summary, the TSFIs described in the functional specification only include the calls made into a TSF  
5594 by external entities and responses to those calls. Calls made by a TSF, which were not explicitly  
5595 considered during evaluation of the components, are described by the reliance information provided  
5596 to satisfy Reliance of dependent component (ACO\_REL).

- 5597 **15.4.4 ACO\_REL.1 Basic reliance information**
- 5598 Dependencies: No dependencies.
- 5599 **15.4.4.1 Developer action elements**
- 5600 **15.4.4.1.1 ACO\_REL.1.1D**
- 5601 **The developer shall provide reliance information of the dependent component.**
- 5602 **15.4.4.2 Content and presentation elements**
- 5603 **15.4.4.2.1 ACO\_REL.1.1C**
- 5604 **The reliance information shall describe the functionality of the base component hardware,**  
5605 **firmware and/or software that is relied upon by the dependent component TSF.**
- 5606 **15.4.4.2.2 ACO\_REL.1.2C**
- 5607 **The reliance information shall describe all interactions through which the dependent**  
5608 **component TSF requests services from the base component.**
- 5609 **15.4.4.2.3 ACO\_REL.1.3C**
- 5610 **The reliance information shall describe how the dependent TSF protects itself from**  
5611 **interference and tampering by the base component.**
- 5612 **15.4.4.3 Evaluator action elements**
- 5613 **15.4.4.3.1 ACO\_REL.1.1E**
- 5614 **The evaluator shall confirm that the information provided meets all requirements for content**  
5615 **and presentation of evidence.**
- 5616 **15.4.5 ACO\_REL.2 Reliance information**
- 5617 Dependencies: No dependencies.
- 5618 **15.4.5.1 Developer action elements**
- 5619 **15.4.5.1.1 ACO\_REL.2.1D**
- 5620 **The developer shall provide reliance information of the dependent component.**
- 5621 **15.4.5.2 Content and presentation elements**
- 5622 **15.4.5.2.1 ACO\_REL.2.1C**
- 5623 **The reliance information shall describe the functionality of the base component hardware, firmware**  
5624 **and/or software that is relied upon by the dependent component TSF.**
- 5625 **15.4.5.2.2 ACO\_REL.2.2C**
- 5626 **The reliance information shall describe all interactions through which the dependent component TSF**  
5627 **requests services from the base component.**

5628 **15.4.5.2.3 ACO\_REL.2.3C**

5629 **The reliance information shall describe each interaction in terms of the interface used and the**  
 5630 **return values from those interfaces.**

5631 **15.4.5.2.4 ACO\_REL.2.4C**

5632 The reliance information shall describe how the dependent TSF protects itself from interference and  
 5633 tampering by the base component.

5634 **15.4.5.3 Evaluator action elements**

5635 **15.4.5.3.1 ACO\_REL.2.1E**

5636 The evaluator shall confirm that the information provided meets all requirements for content and  
 5637 presentation of evidence.

5638 **15.5 Composed TOE testing (ACO\_CTT)**

5639 **15.5.1 Objectives**

5640 This family requires that testing of composed TOE and testing of the base component, as used in the  
 5641 composed TOE, is performed.

5642 **15.5.2 Component levelling**

5643 The components in this family are levelled on the basis of increasing rigour of interface testing and  
 5644 increasing rigour of the analysis of the sufficiency of the tests to demonstrate that the composed TSF  
 5645 operates in accordance with the reliance information and the composed TOE SFRs.

5646 **15.5.3 Application notes**

5647 There are two distinct aspects of testing associated with this family:

5648 a) testing of the interfaces between the base component and the dependent component, which the  
 5649 dependent component rely upon for enforcement of security functionality, to demonstrate their  
 5650 compatibility;

5651 b) testing of the composed TOE to demonstrate that the TOE behaves in accordance with the SFRs for  
 5652 the composed TOE.

5653 If the test configurations used during evaluation of the dependent component included use of the base  
 5654 component as a “platform” and the test analysis sufficiently demonstrates that the TSF behaves in  
 5655 accordance with the SFRs, the developer need perform no further testing of the composed TOE  
 5656 functionality. However, if the base component was not used in the testing of the dependent  
 5657 component, or the configuration of either component varied, then the developer is to perform testing  
 5658 of the composed TOE. This may take the form of repeating the dependent component developer  
 5659 testing of the dependent component, provided this adequately demonstrates the composed TOE TSF  
 5660 behaves in accordance with the SFRs.

5661 The developer is to provide evidence of testing the base component interfaces used in the  
 5662 composition. The operation of base component TSFIs would have been tested as part of the ATE: Tests  
 5663 activities during evaluation of the base component. Therefore, provided the appropriate interfaces  
 5664 were included within the test sample of the base component evaluation and it was determined in  
 5665 Composition rationale (ACO\_COR) that the base component is operating in accordance with the base

5666 component evaluated configuration, with all security functionality required by the dependent  
 5667 component included in the TSF, the evaluator action ACO\_CTT.1.1E may be met through reuse of the  
 5668 base component ATE: Tests verdicts.

5669 If this is not the case, the base component interfaces used relevant to the composition that are affected  
 5670 by any variations to the evaluated configuration and any additional security functionality will be tested  
 5671 to ensure they demonstrate the expected behaviour. The expected behaviour to be tested is that  
 5672 described in the reliance information (Reliance of dependent component (ACO\_REL) evidence).

#### 5673 **15.5.4 ACO\_CTT.1 Interface testing**

5674 Dependencies: ACO\_REL.1 Basic reliance information

5675 ACO\_DEV.1 Functional Description

##### 5676 **15.5.4.1 Objectives**

5677 The objective of this component is to ensure that each interface of the base component, on which the  
 5678 dependent component relies, is tested.

##### 5679 **15.5.4.2 Developer action elements**

###### 5680 **15.5.4.2.1 ACO\_CTT.1.1D**

5681 **The developer shall provide composed TOE test documentation.**

###### 5682 **15.5.4.2.2 ACO\_CTT.1.2D**

5683 **The developer shall provide base component interface test documentation.**

###### 5684 **15.5.4.2.3 ACO\_CTT.1.3D**

5685 **The developer shall provide the composed TOE for testing.**

###### 5686 **15.5.4.2.4 ACO\_CTT.1.4D**

5687 **The developer shall provide an equivalent set of resources to those that were used in the base**  
 5688 **component developer's functional testing of the base component.**

##### 5689 **15.5.4.3 Content and presentation elements**

###### 5690 **15.5.4.3.1 ACO\_CTT.1.1C**

5691 **The composed TOE and base component interface test documentation shall consist of test**  
 5692 **plans, expected test results and actual test results.**

###### 5693 **15.5.4.3.2 ACO\_CTT.1.2C**

5694 **The test documentation from the developer execution of the composed TOE tests shall**  
 5695 **demonstrate that the TSF behaves as specified.**



5696 **15.5.4.3.3 ACO\_CTT.1.3C**

5697 **The test documentation from the developer execution of the base component interface tests**  
 5698 **shall demonstrate that the base component interface relied upon by the dependent component**  
 5699 **behaves as specified.**

5700 **15.5.4.3.4 ACO\_CTT.1.4C**

5701 **The base component shall be suitable for testing.**

5702 **15.5.4.4 Evaluator action elements**

5703 **15.5.4.4.1 ACO\_CTT.1.1E**

5704 **The evaluator shall confirm that the information provided meets all requirements for content**  
 5705 **and presentation of evidence.**

5706 **15.5.4.4.2 ACO\_CTT.1.2E**

5707 **The evaluator shall execute a sample of test in the test documentation to verify the developer**  
 5708 **test results.**

5709 **15.5.4.4.3 ACO\_CTT.1.3E**

5710 **The evaluator shall test a subset of the TSF interfaces of the composed TOE to confirm that the**  
 5711 **composed TSF operates as specified.**

5712 **15.5.5 ACO\_CTT.2 Rigorous interface testing**

5713 Dependencies: ACO\_REL.2 Reliance information

5714 ACO\_DEV.2 Basic evidence of design

5715 **15.5.5.1 Objectives**

5716 The objective of this component is to ensure that each interface of the base component, on which the  
 5717 dependent component relies, is tested.

5718 **15.5.5.2 Developer action elements**

5719 **15.5.5.2.1 ACO\_CTT.2.1D**

5720 The developer shall provide composed TOE test documentation.

5721 **15.5.5.2.2 ACO\_CTT.2.2D**

5722 The developer shall provide base component interface test documentation.

5723 **15.5.5.2.3 ACO\_CTT.2.3D**

5724 The developer shall provide the composed TOE for testing.

5725 **15.5.5.2.4 ACO\_CTT.2.4D**

5726 The developer shall provide an equivalent set of resources to those that were used in the base  
 5727 component developer's functional testing of the base component.

5728 **15.5.5.3 Content and presentation elements**

5729 **15.5.5.3.1 ACO\_CTT.2.1C**

5730 The composed TOE and base component interface test documentation shall consist of test plans,  
5731 expected test results and actual test results.

5732 **15.5.5.3.2 ACO\_CTT.2.2C**

5733 The test documentation from the developer execution of the composed TOE tests shall demonstrate  
5734 that the TSF behaves as specified **and is complete**.

5735 **15.5.5.3.3 ACO\_CTT.2.3C**

5736 The test documentation from the developer execution of the base component interface tests shall  
5737 demonstrate that the base component interface relied upon by the dependent component behaves as  
5738 specified **and is complete**.

5739 **15.5.5.3.4 ACO\_CTT.2.4C**

5740 The base component shall be suitable for testing.

5741 **15.5.5.4 Evaluator action elements**

5742 **15.5.5.4.1 ACO\_CTT.2.1E**

5743 The evaluator shall confirm that the information provided meets all requirements for content and  
5744 presentation of evidence.

5745 **15.5.5.4.2 ACO\_CTT.2.2E**

5746 The evaluator shall execute a sample of test in the test documentation to verify the developer test  
5747 results.

5748 **15.5.5.4.3 ACO\_CTT.2.3E**

5749 The evaluator shall test a subset of the TSF interfaces of the composed TOE to confirm that the  
5750 composed TSF operates as specified.

5751 **15.6 Composition vulnerability analysis (ACO\_VUL)**

5752 **15.6.1 Objectives**

5753 This family calls for an analysis of vulnerability information available in the public domain and of  
5754 vulnerabilities that may be introduced as a result of the composition.

5755 **15.6.2 Component levelling**

5756 The components in this family are levelled on the basis of increasing scrutiny of vulnerability  
5757 information from the public domain and independent vulnerability analysis.

5758 **15.6.3 Application notes**

5759 The developer will provide details of any residual vulnerabilities reported during evaluation of the  
5760 components. These may be gained from the component developers or evaluation reports for the

5761 components. These will be used as inputs into the evaluator's vulnerability analysis of the composed  
5762 TOE in the operational environment.

5763 The operational environment of the composed TOE is examined to ensure that the assumptions and  
5764 objectives for the component operational environment (specified in each component ST) are satisfied  
5765 in the composed TOE. An initial analysis of the consistency of assumptions and objectives between the  
5766 components and the composed TOE STs will have been performed during the conduct of the ASE  
5767 activities for the composed TOE. However, this analysis is revisited with the knowledge acquired  
5768 during the ACO\_REL, ACO\_DEV and the ACO\_COR activities to ensure that, for example, assumptions of  
5769 the dependent component that were addressed by the environment in the dependent component ST  
5770 are not reintroduced as a result of composition (i.e. that the base component adequately addresses the  
5771 assumptions of the dependent component ST in the composed TOE).

5772 A search by the evaluator for issues in each component will identify potential vulnerabilities reported  
5773 in the public domain since completion of the evaluation of the components. Any potential  
5774 vulnerabilities will then be subject to testing.

5775 If the base component used in the composed TOE has been the subject of assurance continuity  
5776 activities since certification, the evaluator will consider during the composed TOE vulnerability  
5777 analysis activities the changes made in base component.

#### 5778 **15.6.4 ACO\_VUL.1 Composition vulnerability review**

5779 Dependencies: ACO\_DEV.1 Functional Description

##### 5780 **15.6.4.1 Developer action elements**

###### 5781 **15.6.4.1.1 ACO\_VUL.1.1D**

5782 **The developer shall provide the composed TOE for testing.**

##### 5783 **15.6.4.2 Content and presentation elements**

###### 5784 **15.6.4.2.1 ACO\_VUL.1.1C**

5785 **The composed TOE shall be suitable for testing.**

##### 5786 **15.6.4.3 Evaluator action elements**

###### 5787 **15.6.4.3.1 ACO\_VUL.1.1E**

5788 **The evaluator shall confirm that the information provided meets all requirements for content**  
5789 **and presentation of evidence.**

###### 5790 **15.6.4.3.2 ACO\_VUL.1.2E**

5791 **The evaluator shall perform an analysis to determine that any residual vulnerabilities**  
5792 **identified for the base and dependent components are not exploitable in the composed TOE in**  
5793 **its operational environment.**

###### 5794 **15.6.4.3.3 ACO\_VUL.1.3E**

5795 **The evaluator shall perform a search of public domain sources to identify possible**  
5796 **vulnerabilities arising from use of the base and dependent components in the composed TOE**  
5797 **operational environment.**

5798 **15.6.4.3.4 ACO\_VUL.1.4E**

5799 **The evaluator shall conduct penetration testing, based on the identified vulnerabilities, to**  
 5800 **demonstrate that the composed TOE is resistant to attacks by an attacker with basic attack**  
 5801 **potential.**

5802 **15.6.5 ACO\_VUL.2 Composition vulnerability analysis**

5803 Dependencies: ACO\_DEV.2 Basic evidence of design

5804 **15.6.5.1 Developer action elements**5805 **15.6.5.1.1 ACO\_VUL.2.1D**

5806 The developer shall provide the composed TOE for testing.

5807 **15.6.5.2 Content and presentation elements**5808 **15.6.5.2.1 ACO\_VUL.2.1C**

5809 The composed TOE shall be suitable for testing.

5810 **15.6.5.3 Evaluator action elements**5811 **15.6.5.3.1 ACO\_VUL.2.1E**

5812 The evaluator shall confirm that the information provided meets all requirements for content and  
 5813 presentation of evidence.

5814 **15.6.5.3.2 ACO\_VUL.2.2E**

5815 The evaluator shall perform an analysis to determine that any residual vulnerabilities identified for  
 5816 the base and dependent components are not exploitable in the composed TOE in its operational  
 5817 environment.

5818 **15.6.5.3.3 ACO\_VUL.2.3E**

5819 The evaluator shall perform a search of public domain sources to identify possible vulnerabilities  
 5820 arising from use of the base and dependent components in the composed TOE operational  
 5821 environment.

5822 **15.6.5.3.4 ACO\_VUL.2.4E**

5823 **The evaluator shall perform an independent vulnerability analysis of the composed TOE, using**  
 5824 **the guidance documentation, reliance information and composition rationale to identify**  
 5825 **potential vulnerabilities in the composed TOE.**

5826 **15.6.5.3.5 ACO\_VUL.2.5E**

5827 The evaluator shall conduct penetration testing, based on the identified vulnerabilities, to demonstrate  
 5828 that the composed TOE is resistant to attacks by an attacker with basic attack potential.

5829 **15.6.6 ACO\_VUL.3 Enhanced-Basic Composition vulnerability analysis**

5830 Dependencies: ACO\_DEV.3 Detailed evidence of design

5831 **15.6.6.1 Developer action elements**

5832 **15.6.6.1.1 ACO\_VUL.3.1D**

5833 The developer shall provide the composed TOE for testing.

5834 **15.6.6.2 Content and presentation elements**

5835 **15.6.6.2.1 ACO\_VUL.3.1C**

5836 The composed TOE shall be suitable for testing.

5837 **15.6.6.3 Evaluator action elements**

5838 **15.6.6.3.1 ACO\_VUL.3.1E**

5839 The evaluator shall confirm that the information provided meets all requirements for content and  
5840 presentation of evidence.

5841 **15.6.6.3.2 ACO\_VUL.3.2E**

5842 The evaluator shall perform an analysis to determine that any residual vulnerabilities identified for  
5843 the base and dependent components are not exploitable in the composed TOE in its operational  
5844 environment.

5845 **15.6.6.3.3 ACO\_VUL.3.3E**

5846 The evaluator shall perform a search of public domain sources to identify possible vulnerabilities  
5847 arising from use of the base and dependent components in the composed TOE operational  
5848 environment.

5849 **15.6.6.3.4 ACO\_VUL.3.4E**

5850 The evaluator shall perform an independent vulnerability analysis of the composed TOE, using the  
5851 guidance documentation, reliance information and composition rationale to identify potential  
5852 vulnerabilities in the composed TOE.

5853 **15.6.6.3.5 ACO\_VUL.3.5E**

5854 The evaluator shall conduct penetration testing, based on the identified vulnerabilities, to demonstrate  
5855 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.1ADV\_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.

5938 For cases where the TOE depends upon other IT entities to play a role in domain separation, that  
 5939 sharing of roles must be made clear. For example, a TOE that is solely application software relies upon  
 5940 the underlying operating system to correctly instantiate the domains that the TOE defines; if the TOE  
 5941 defines separate processing space, memory space, etc, for each domain, it depends upon the  
 5942 underlying operating system to operate correctly and benignly (e.g. allow the process to execute only  
 5943 in the execution space that is requested by the TOE software).

5944 For example, mechanisms that implement domain separation (e.g., memory management, protected  
 5945 processing modes provided by the hardware, etc.) would be identified and described. Or, the TSF  
 5946 might implement software protection constructs or coding conventions that contribute to  
 5947 implementing separation of software domains, perhaps by delineating user address space from system  
 5948 address space.

5949 The vulnerability analysis and testing (see AVA\_VAN) activities will likely include attempts to defeat  
 5950 the described TSF domain separation through the use of monitoring or direct attack the TSF.

#### 5951 **A.1.2.2 TSF Self-protection**

5952 In cases where the TOE exhibits self-protection entirely on its own, there would be a straightforward  
 5953 description of how this self-protection is attained. Mechanisms that provide domain separation to  
 5954 define a TSF domain that is protected from other (user) domains would be identified and described.

5955 For cases where the TOE depends upon other IT entities to play a role in protecting itself, that sharing  
 5956 of roles must be made clear. For example, a TOE that is solely application software relies upon the  
 5957 underlying operating system to operate correctly and benignly; the application cannot protect itself  
 5958 against a malicious operating system that subverts it (for example, by overwriting its executable code  
 5959 or TSF data).

5960 The security architecture description also covers how user input is handled by the TSF in such a way  
 5961 that the TSF does not subject itself to being corrupted by that user input. For example, the TSF might  
 5962 implement the notion of privilege and protect itself by using privileged-mode routines to handle user  
 5963 data. The TSF might make use of processor-based separation mechanisms (e.g. privilege levels or  
 5964 rings) to separate TSF code and data from user code and data. The TSF might implement software  
 5965 protection constructs or coding conventions that contribute to implementing separation of software,  
 5966 perhaps by delineating user address space from system address space.

5967 For TOEs that start up in a low-function mode (for example, a single-user mode accessible only to  
 5968 installers or administrators) and then transition to the evaluated secure configuration (a mode  
 5969 whereby untrusted users are able to login and use the services and resources of the TOE), the security  
 5970 architecture description also includes an explanation of how the TSF is protected against this  
 5971 initialisation code that does not run in the evaluated configuration. For such TOEs, the security  
 5972 architecture description would explain what prevents those services that should be available only  
 5973 during initialisation (e.g. direct access to resources) from being accessible in the evaluated  
 5974 configuration. It would also explain what prevents initialisation code from running while the TOE is in  
 5975 the evaluated configuration.

5976 There must also be an explanation of how the trusted initialisation code will maintain the integrity of  
 5977 the TSF (and of its initialisation process) such that the initialisation process is able to detect any  
 5978 modification that would result in the TSF being spoofed into believe it was in an initial secure state.

5979 The vulnerability analysis and testing (see AVA\_VAN) activities will likely include attempts to defeat  
 5980 the described TSF self protection through the use of tampering, direct attack, or monitoring of the TSF.



### 5981 **A.1.2.3 TSF Non-Bypassability**

5982 The property of non-bypassability is concerned with interfaces that permit the bypass of the  
 5983 enforcement mechanisms. In most cases this is a consequence of the implementation, where if a  
 5984 programmer is writing an interface that accesses or manipulates an object, it is that programmer's  
 5985 responsibility to use interfaces that are part of the SFR enforcement mechanism for the object and not  
 5986 to try to circumvent those interfaces. For the description pertaining to non-bypassability, then, there  
 5987 are two broad areas that have to be covered.

5988 The first consists of those interfaces to the SFR-enforcement. The property for these interfaces is that  
 5989 they contain no operations or modes that allow them to be used to bypass the TSF. It is likely that the  
 5990 evidence for ADV\_FSP and ADV\_TDS can be used in large part to make this determination. Because  
 5991 non-bypassability is the concern, if only certain operations available through these TSFIs are  
 5992 documented (because they are SFR-enforcing) and others are not, the developer should consider  
 5993 whether additional information (to that presented in ADV\_FSP and ADV\_TDS) is necessary to make a  
 5994 determination that the SFR-supporting and SFR-non-interfering operations of the TSFI do not afford  
 5995 an untrusted entity the ability to bypass the policy being enforced. If such information is necessary, it  
 5996 is included in the security architecture description.

5997 The second area of non-bypassability is concerned with those interfaces whose interactions are not  
 5998 associated with SFR-enforcement. Depending on the ADV\_FSP and ADV\_TDS components claimed,  
 5999 some information about these interfaces may or may not exist in the functional specification and TOE  
 6000 design documentation. The information presented for such interfaces (or groups of interfaces) should  
 6001 be sufficient so that a reader can make a determination (at the level of detail commensurate with the  
 6002 rest of the evidence supplied in the ADV: Development class) that the enforcement mechanisms cannot  
 6003 be bypassed.

6004 The property that the security functionality cannot be bypassed applies to all security functionality  
 6005 equally. That is, the design description should cover objects that are protected under the SFRs (e.g.  
 6006 FDP\_\* components) and functionality (e.g., audit) that is provided by the TSF. The description should  
 6007 also identify the interfaces that are associated with security functionality; this might make use of the  
 6008 information in the functional specification. This description should also describe any design  
 6009 constructs, such as object managers, and their method of use. For instance, if routines are to use a  
 6010 standard macro to produce an audit record, this convention is a part of the design that contributes to  
 6011 the non-bypassability of the audit mechanism. It is important to note that *non-bypassability* in this  
 6012 context is not an attempt to answer the question "could a part of the TSF implementation, if malicious,  
 6013 bypass the security functionality", but rather to document how the implementation does not bypass  
 6014 the security functionality.

6015 The vulnerability analysis and testing (see AVA\_VAN) activities will likely include attempts to defeat  
 6016 the described non-bypassability by circumventing the TSF.

### 6017 **A.2ADV\_FSP: Supplementary material on functional specification**

6018 The purpose in specifying the TSFIs is to provide the necessary information to conduct testing;  
 6019 without knowing the possible means interact with the TSF, one cannot adequately test the behaviour  
 6020 of the TSF.

6021 There are two parts to specifying the TSFIs: identifying them and describing them. Because of the  
 6022 diversity of possible TOEs, and of different TSFs therein, there is no standard set of interfaces that  
 6023 constitute "TSFIs". This annex provides guidance on the factors that determine which interfaces are  
 6024 TSFIs.

### 6025 **A.2.1 Non-TSF part of the TOE**

6026 The TSF comprises all parts of the TOE the user has to rely on in order to trust (in the end) the  
6027 security functionality.

6028 To say it in other words: Those parts of the TOE that do not belong to the TSF can be modified by  
6029 an attacker without any impact on the TOE security functionality. If this isn't the case, these parts  
6030 of the TOE have to be included in the TSF.

6031 If the TSF and the TSF implementation are defined then it is clear (physical view) whether there  
6032 exist further parts of the TOE which can be classified as non-TSF parts of the TOE. Such parts do  
6033 not have to be part of the TSF but they are still part of the TOE.

6034 The relationship between TSF and non-TSF parts of TOE is given by their definitions and the ARC  
6035 properties as follows: (1) non-TSF parts do not bypass the TSF and (2) parts of the TSF protects  
6036 themselves against tampering. But the question is how to decide whether specific functionality  
6037 (functional view) or component (physical view) belongs to TSF or not.

6038 A subsystem of the TOE which is not part of the TSF has to fulfil the following condition (described  
6039 as a rule of thumb<sup>8</sup>): The subsystem must not have any security impact of the TOE even if it were  
6040 substituted by an attacker.

6041 Therefore between the Non-TSF parts and the TSF parts it seems that some kind of "separation  
6042 mechanism" is necessary<sup>9</sup> because such "separation mechanism" could build the basis for the  
6043 assessment that there is no impact on the TSF parts from the Non-TSF parts possible.

6044 Such "separation mechanism" could be implemented by the security architecture or by an explicitly  
6045 realised part of the implementation (e.g. a firewall between TSF and Non-TSF parts of the TOE).

6046 The analysis of the "separation mechanism" is then subject of the vulnerability assessment  
6047 because it must withstand attacks by an attacker of the respective strength according to the VAN  
6048 level of the evaluation.

6049 The developer shall provide evidence for non-bypassability and self-protection in its security  
6050 architecture description and the evaluator shall analyse this evidence in subactivity for ADV\_ARC.1  
6051 and assess the effectiveness in the vulnerability assessment.

6052 The goal of TOE design documentation is to provide sufficient information to determine the TSF  
6053 boundary, and to describe *how* the TSF implements the SFR. Further attention is needed by the  
6054 fact that the family ADV\_TDS requires only identification of the non-TSF subsystems of the TOE.  
6055 No interface description is provided for these subsystems in ADV\_FSP or ADV\_TDS. SFR non-  
6056 interference of these subsystems is assumed but not demonstrated by the developer and not  
6057 examined in details by the evaluator. However from the TOE design point of view this is not that  
6058 important as long as the above mentioned separation mechanism is in place and the vulnerability  
6059 assessment confirms that it is strong enough. Therefore this "separation mechanism" implements  
6060 the TSF or enforces ARC properties as security feature. But non-bypassability may be enforced by  
6061 "pure architecture properties" as well.

---

<sup>8</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>9</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.

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 behaviour of the TSF. As such, all these interfaces (electrical contacts) need to be described: various voltages that might be applied, etc.

### 6103 A.2.2.2 Network protocol stack

6104 The TSFIs of a TOE that performs protocol processing would be those protocol layers to which a  
6105 potential attacker has direct access. This need not be the entire protocol stack, but it might be.

6106 For example, if the TOE were some sort of a network appliance that allowed potential attackers to  
6107 affect every level of the protocol stack (i.e. to send arbitrary signals, arbitrary voltages, arbitrary  
6108 packets, arbitrary datagrams, etc.), then the TSF boundary exists at each layer of the stack. Therefore,  
6109 the functional specification would have to address every protocol at every layer of the stack.

6110 If, however, the TOE were a firewall that protects an internal network from the Internet, a potential  
6111 attacker would have no means of directly manipulating the voltages that enter the TOE; any extreme  
6112 voltages would simply not be passed though the Internet. That is, the attacker would have access only  
6113 to those protocols at the Internet layer or above. The TSF boundary exists at each layer of the stack.  
6114 Therefore, the functional specification would have to address only those protocols at or above the  
6115 Internet layer: it would describe each of the different communication layers at which the firewall is  
6116 exposed in terms of what constitutes well-formed input for what might appear on the line, and the  
6117 result of both well-formed and malformed inputs. For example, the description of the Internet protocol  
6118 layer would describe what constitutes a well-formed IP packet and what happens when both  
6119 correctly-formed and malformed packets are received. Likewise, the description of the TCP layer  
6120 would describe a successful TCP connection and what happens both when successful connections are  
6121 established and when connections cannot be established or are inadvertently dropped. Presuming the  
6122 firewall's purpose is to filter application-level commands (like FTP or telnet), the description of the  
6123 application layer would describe the application-level commands that are recognised and filtered by  
6124 the firewall, as well as the results of encountering unknown commands.

6125 The descriptions of these layers would likely reference published communication standards (telnet,  
6126 FTP, TCP, etc.) that are used, noting which user-defined options are chosen.

### 6127 A.2.2.3 Wrappers

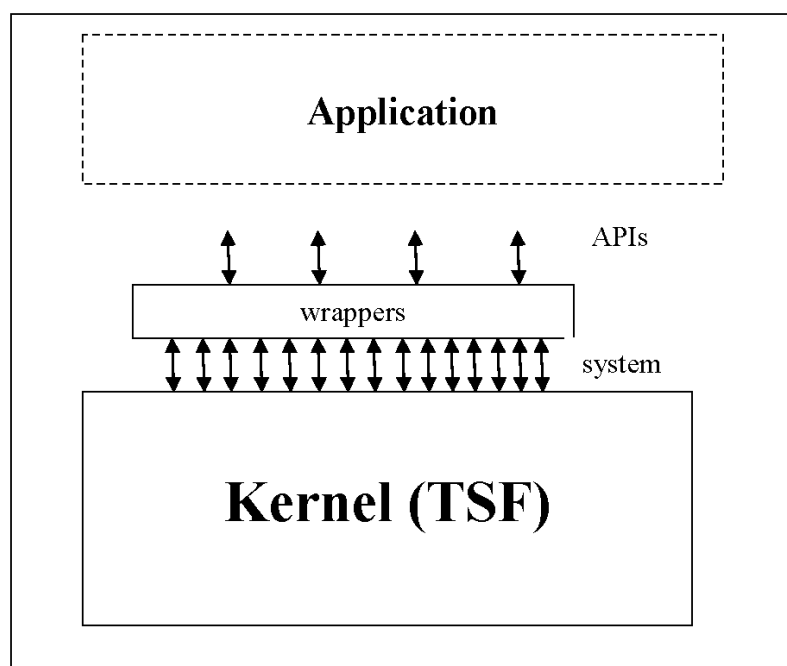


Figure A.1 — Wrappers

“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.

#### 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:

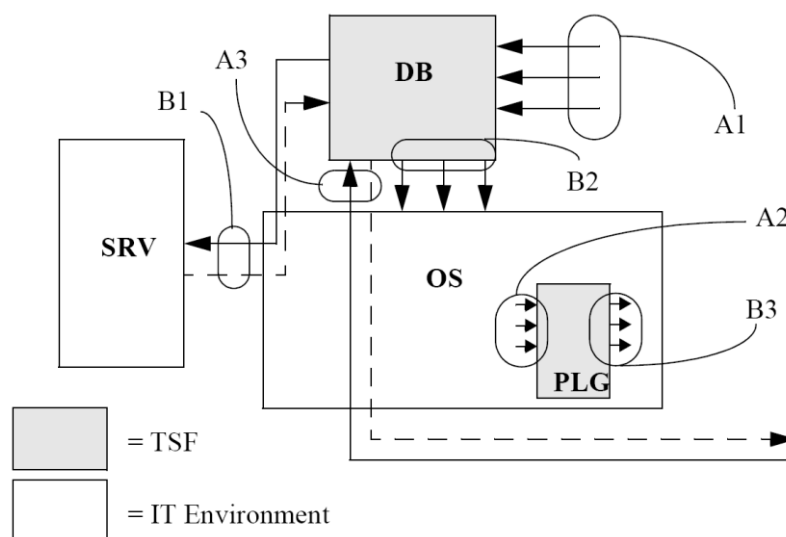
- a) If the security objectives for the operational environment for the stand-alone firewall state that “the firewall will be operational in a server room environment to which only trusted and trained personnel will have access, and which will be equipped with an interruptible power supply (against power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

For all of these examples, these inaccessible interfaces would not be TSFIs.

#### A.2.3 Example: A complex DBMS

Figure A.2 illustrates a complex TOE: a database management system that relies on hardware and software that is outside the TOE boundary (referred to as the *IT environment* in the rest of this discussion). To simplify this example, the TOE is identical to the TSF. The shaded boxes represent the TSF, while the unshaded boxes represent IT entities in the environment. The TSF comprises the database engine and management GUIs (represented by the box labelled *DB*) and a kernel module that runs as part of the OS that performs some security function (represented by the box labelled *PLG*). The

TSF kernel module has entry points defined by the OS specification that the OS will call to invoke some function (this could be a device driver, or an authentication module, etc.). The key is that this pluggable kernel module is providing security services specified by functional requirements in the ST.



**Figure A.2 — Interfaces in a DBMS system**

The IT environment consists of the operating system itself (represented by the box labelled *OS*), as well as an external server (labelled *SRV*). This external server, like the *OS*, provides a service that the TSF depends on, and thus needs to be in the IT environment. Interfaces in the figure are labelled *A<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.

Interface group *A1* represents the most obvious set of TSFI. These are interfaces used by users to directly access the database and its security functionality and resources.

Interface group *A2* represent the TSFI that the *OS* invokes to obtain the functionality provided by the pluggable module. These are contrasted with interface group *B3*, which represent calls that the pluggable module makes to obtain services from the IT environment.

Interface group *A3* represent TSFI that pass through the IT environment. In this case, the DBMS communicates over the network using a proprietary application-level protocol. While the IT environment is responsible for providing various supporting protocols (e.g., Ethernet, IP, TCP), the application layer protocol that is used to obtain services from the DBMS is a TSFI and must be documented as such. The dotted line indicates return values/services from the TSF over the network connection.

The interfaces labelled *B<sub>x</sub>* represent interfaces to functionality in the IT Environment. These interfaces are not TSFI and need only be discussed and analysed when the TOE is being used in a composite evaluation as part of the activities associated with the ACO class.

#### A.2.4 Example Functional Specification

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

6202 from the internal network. Administrator actions consist of authenticating, changing passwords,  
6203 reviewing the audit log, and setting or changing the addresses of the internal and external networks.

6204 The Example firewall presents the following interfaces to the internal network:

6205 a) IP datagrams

6206 b) Administrator Commands

6207 and the following interfaces to the external network:

6208 a) IP datagrams

### 6209 Interfaces Descriptions: IP Datagrams

6210 The datagrams are in the format specified by RFC 791.

6211 • Purpose - to transmit blocks of data ("datagrams") from source hosts to destination hosts  
6212 identified by fixed length addresses; also provides for fragmentation and reassembly of long  
6213 datagrams, if necessary, for transmission through small-packet networks.

6214 • Method of Use - they arrive from the lower-level (e.g. data link) protocol.

6215 • Parameters - the following fields of the IP datagram header: source address, destination address,  
6216 don't-fragment flag.

6217 • Parameter description - [As defined by RFC 791, subclause 3.1 ("Internet Header Format")]

6218 • Actions - Transmits datagrams that are not masquerading; fragments large datagrams if necessary;  
6219 reassembles fragments into datagrams.

6220 • Error messages - (none). No reliability guaranteed (reliability to be provided by upper-level  
6221 protocols) Undeliverable datagrams (e.g. must be fragmented for transmission, but don't-fragment  
6222 flag is set) dropped.

### 6223 Interfaces Descriptions: Administrator Commands

6224 The administrator commands provide a means for the administrator to interact with the firewall.  
6225 These **commands and responses ride atop a telnet (RFC 854) connection established from any**  
6226 **host on the internal** network. Available commands are:

#### 6227 • **Passwd**

6228 • Purpose - sets administrator password

6229 • Method of Use - **Passwd** <password>

6230 • Parameters - password

6231 • Parameter description - value of new password

6232 • Actions - changes password to new value supplied. There are no restrictions.

- 6233 • Error messages - none.
- 6234 • **Readaudit**
- 6235 • Purpose - presents the audit log to the administrator
- 6236 • Method of Use - **Readaudit**
- 6237 • Parameters - none
- 6238 • Parameter description - none
- 6239 • Actions - provides the text of the audit log
- 6240 • Error messages - none.
- 6241 • **Setintaddr**
- 6242 • Purpose - sets the address of the internal address.
- 6243 • Method of Use - **Setintaddr** <address>
- 6244 • Parameters - address
- 6245 • Parameter description - first three fields of an IP address (as defined in RFC 791). For example:  
6246 123.123.123.
- 6247 • Actions - changes the internal value of the variable defining the internal network, the value of  
6248 which is used to judge attempted masquerades.
- 6249 • Error messages - “address in use”: indicates the identified internal network is the same as the  
6250 external network.
- 6251 • **Setextaddr**
- 6252 • Purpose - sets the address of the external address
- 6253 • Method of Use - **Setextaddr** <address>
- 6254 • Parameters - address
- 6255 • Parameter description - first three fields of an IP address (as defined in RFC 791). For example:  
6256 123.123.123.
- 6257 • Actions - changes the internal value of the variable defining the external network.
- 6258 • Error messages - “address in use”: indicates the identified external network is the same as the  
6259 internal network.

### 6260 **A.3ADV\_INT: Supplementary material on TSF internals**

- 6261 The wide variety of TOEs makes it impossible to codify anything more specific than “well-structured”  
6262 or “minimum complexity”. Judgements on structure and complexity are expected to be derived from



6263 the specific technologies used in the TOE. For example, software is likely to be considered well-  
 6264 structured if it exhibits the characteristics cited in the software engineering disciplines.

6265 This annex provides supplementary material on assessing the structure and complexity of procedure-  
 6266 based software portions of the TSF. This material is based on information readily available in software  
 6267 engineering literature. For other kinds of internals (e.g. hardware, non-procedural software such as  
 6268 object-oriented code, etc.), corresponding literature on good practises should be consulted.

### 6269 **A.3.1 Structure of procedural software**

6270 The structure of procedural software is traditionally assessed according to its *modularity*. Software  
 6271 written with a modular design aids in achieving understandability by clarifying what dependencies a  
 6272 module has on other modules (*coupling*) and by including in a module only tasks that are strongly  
 6273 related to each other (*cohesion*). The use of modular design reduces the interdependence between  
 6274 elements of the TSF and thus reduces the risk that a change or error in one module will have effects  
 6275 throughout the TOE. Its use enhances clarity of design and provides for increased assurance that  
 6276 unexpected effects do not occur. Additional desirable properties of modular decomposition are a  
 6277 reduction in the amount of redundant or unneeded code.

6278 Minimising the amount of functionality in the TSF allows the evaluator as well as the developer to  
 6279 focus only on that functionality which is necessary for SFR enforcement, contributing further to  
 6280 understandability and further lowering the likelihood of design or implementation errors.

6281 The incorporation of modular decomposition, layering and minimisation into the design and  
 6282 implementation process must be accompanied by sound software engineering considerations. A  
 6283 practical, useful software system will usually entail some undesirable coupling among modules, some  
 6284 modules that include loosely-related functions, and some subtlety or complexity in a module's design.  
 6285 These deviations from the ideals of modular decomposition are often deemed necessary to achieve  
 6286 some goal or constraint, be it related to performance, compatibility, future planned functionality, or  
 6287 some other factors, and may be acceptable, based on the developer's justification for them. In applying  
 6288 the requirements of this class, due consideration must be given to sound software engineering  
 6289 principles; however, the overall objective of achieving understandability must be achieved.

#### 6290 **A.3.1.1 Cohesion**

6291 Cohesion is the manner and degree to which the tasks performed by a single software module are  
 6292 related to one another; types of cohesion include coincidental, communicational, functional, logical,  
 6293 sequential, and temporal. These types of cohesion are characterised below, listed in the order of  
 6294 decreasing desirability.

6295 a) *functional* cohesion - a module with functional cohesion performs activities related to a single  
 6296 purpose. A functionally cohesive module transforms a single type of input into a single type of  
 6297 output, such as a stack manager or a queue manager.

6298 b) *sequential* cohesion - a module with sequential cohesion contains functions each of whose output  
 6299 is input for the following function in the module. An example of a sequentially cohesive module is  
 6300 one that contains the functions to write audit records and to maintain a running count of the  
 6301 accumulated number of audit violations of a specified type.

6302 c) *communicational* cohesion - a module with communicational cohesion contains functions that  
 6303 produce output for, or use output from, other functions within the module. An example of a  
 6304 communicationally cohesive module is an access check module that includes mandatory,  
 6305 discretionary, and capability checks.

6306 d) *temporal* cohesion - a module with temporal cohesion contains functions that need to be executed  
 6307 at about the same time. Examples of temporally cohesive modules include initialisation, recovery,  
 6308 and shutdown modules.

6309 e) *logical* (or *procedural*) cohesion - a module with logical cohesion performs similar activities on  
 6310 different data structures. A module exhibits logical cohesion if its functions perform related, but  
 6311 different, operations on different inputs.

6312 f) *coincidental* cohesion - a module with coincidental cohesion performs unrelated, or loosely  
 6313 related, activities.

#### 6314 **A.3.1.2 Coupling**

6315 Coupling is the manner and degree of interdependence between software modules; types of coupling  
 6316 include call, common and content coupling. These types of coupling are characterised below, listed in  
 6317 the order of decreasing desirability:

6318 a) *call*: two modules are call coupled if they communicate strictly through the use of their  
 6319 documented function calls; examples of call coupling are data, stamp, and control, which are  
 6320 defined below.

6321 1) *data*: two modules are data coupled if they communicate strictly through the use of call  
 6322 parameters that represent single data items.

6323 2) *stamp*: two modules are stamp coupled if they communicate through the use of call  
 6324 parameters that comprise multiple fields or that have meaningful internal structures.

6325 3) *control*: two modules are control coupled if one passes information that is intended to  
 6326 influence the internal logic of the other.

6327 b) *common*: two modules are common coupled if they share a common data area or a common  
 6328 system resource. Global variables indicate that modules using those global variables are common  
 6329 coupled. Common coupling through global variables is generally allowed, but only to a limited  
 6330 degree. For example, variables that are placed into a global area, but are used by only a single  
 6331 module, are inappropriately placed, and should be removed. Other factors that need to be  
 6332 considered in assessing the suitability of global variables are:

6333 1) The number of modules that modify a global variable: In general, only a single module should  
 6334 be allocated the responsibility for controlling the contents of a global variable, but there may  
 6335 be situations in which a second module may share that responsibility; in such a case, sufficient  
 6336 justification must be provided. It is unacceptable for this responsibility to be shared by more  
 6337 than two modules. (In making this assessment, care should be given to determining the  
 6338 module actually responsible for the contents of the variable; for example, if a single routine is  
 6339 used to modify the variable, but that routine simply performs the modification requested by  
 6340 its caller, it is the calling module that is responsible, and there may be more than one such  
 6341 module). Further, as part of the complexity determination, if two modules are responsible for  
 6342 the contents of a global variable, there should be clear indications of how the modifications  
 6343 are coordinated between them.

6344 2) The number of modules that reference a global variable: Although there is generally no limit  
 6345 on the number of modules that reference a global variable, cases in which many modules  
 6346 make such a reference should be examined for validity and necessity.

6347 c) *content*: two modules are content coupled if one can make direct reference to the internals of the  
 6348 other (e.g. modifying code of, or referencing labels internal to, the other module). The result is that  
 6349 some or all of the content of one module are effectively included in the other. Content coupling can  
 6350 be thought of as using unadvertised module interfaces; this is in contrast to call coupling, which  
 6351 uses only advertised module interfaces.

### 6352 **A.3.2 Complexity of procedural software**

6353 Complexity is the measure of the decision points and logical paths of execution that code takes.  
 6354 Software engineering literature cites complexity as a negative characteristic of software because it  
 6355 impedes understanding of the logic and flow of the code. Another impediment to the understanding of  
 6356 code is the presence of code that is unnecessary, in that it is unused or redundant.

6357 The use of layering to separate levels of abstraction and minimise circular dependencies further  
 6358 enables a better understanding of the TSF, providing more assurance that the TOE security functional  
 6359 requirements are accurately and completely instantiated in the implementation.

6360 Reducing complexity also includes reducing or eliminating mutual dependencies, which pertains both  
 6361 to modules in a single layer and to those in separate layers. Modules that are mutually dependent may  
 6362 rely on one another to formulate a single result, which could result in a deadlock condition, or worse  
 6363 yet, a race condition (e.g., time of check vs. time of use concern), where the ultimate conclusion could  
 6364 be indeterminate and subject to the computing environment at the given instant in time.

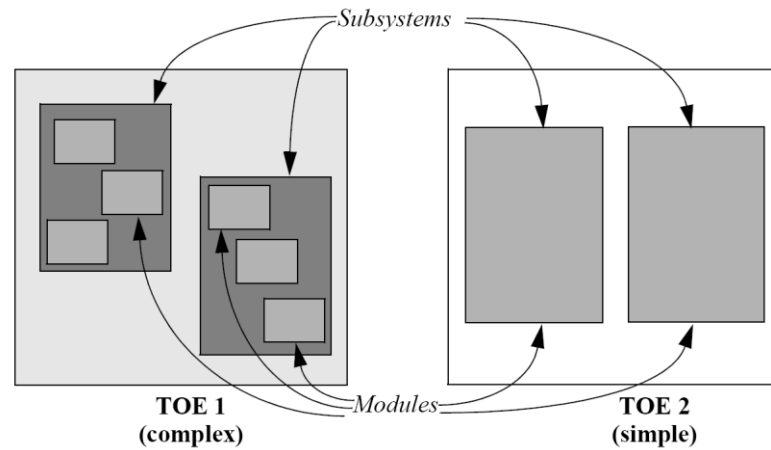
6365 Design complexity minimisation is a key characteristic of a reference validation mechanism, the  
 6366 purpose of which is to arrive at a TSF that is easily understood so that it can be completely analysed.  
 6367 (There are other important characteristics of a reference validation mechanism, such as TSF self-  
 6368 protection and non-bypassability; these other characteristics are covered by requirements in the  
 6369 ADV\_ARC family.)

## 6370 **A.4 ADV\_TDS: Subsystems and Modules**

6371 This subclause provides additional guidance on the TDS family, and its use of the terms “subsystem”  
 6372 and “module”. This is followed by a discussion of how, as more-detailed becomes available, the  
 6373 requirement for the less-detailed is reduced.

### 6374 **A.4.1 Subsystems**

6375 Figure A.3 shows that, depending on the complexity of the TSF, the design may be described in terms  
 6376 of subsystems *and* modules (where subsystems are at a higher level of abstraction than modules); or it  
 6377 may just be described in terms of one level of abstraction (e.g., *subsystems* at lower assurance levels,  
 6378 *modules* at higher levels). In cases where a lower level of abstraction (modules) is presented,  
 6379 requirements levied on higher-level abstractions (subsystems) are essentially met by default. This  
 6380 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.

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

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

#### A.4.2 Modules

A module is generally a relatively small architectural unit that can be characterised in terms of the properties discussed in TSF internals (ADV\_INT). When both ADV\_TDS.3 Basic modular design (or above) requirements and TSF internals (ADV\_INT) requirements are present in a PP or ST, a “module” in terms of the TOE design (ADV\_TDS) requirements refers to the same entity as a “module” for the

6413 TSF internals (ADV\_INT) requirements. Unlike subsystems, modules describe the implementation in a  
 6414 level of detail that can serve as a guide to reviewing the implementation representation.

6415 It is important to note that, depending on the TOE, modules and subsystems may refer to the same  
 6416 abstraction. For ADV\_TDS.1 Basic design and ADV\_TDS.2 Architectural design (which do not require  
 6417 description at the module level) the subsystem description provides the lowest level detail available  
 6418 about the TSF. For ADV\_TDS.3 Basic modular design (which require module descriptions) these  
 6419 descriptions provide the lowest level of detail, while the subsystem descriptions (if they exist as  
 6420 separate entities) merely serve to put to the module descriptions in context. That is, it is not necessary  
 6421 to provide detailed subsystem descriptions if module descriptions exist. In TOEs that are sufficiently  
 6422 simple, a separate “subsystem description” is not necessary; the requirements can be met through  
 6423 documentation provided by modules. For complex TOEs, the purpose of the subsystem description  
 6424 (with respect to the TSF) is to provide the reader context so they can focus their analysis  
 6425 appropriately. This difference is illustrated in Figure A.3.

6426 An SFR-enforcing module is a module that completely or partially implements a security functional  
 6427 requirement (SFR) in the ST. Such modules may implement an SFR-enforcing TSFI, but some  
 6428 functionality expressed in an SFR (for example, audit and object re-use functionality) may not be  
 6429 directly tied to a single TSFI. As was the case with subsystems, SFR-supporting modules are those  
 6430 modules that are depended upon by an SFR-enforcing module, but are not responsible for directly  
 6431 implementing an SFR. SFR-non-interfering modules are those modules that do not deal, directly or  
 6432 indirectly, with the enforcement of SFRs.

6433 It is important to note that the determination of what “directly implements” means is somewhat  
 6434 subjective. In the narrowest sense of the term, it could be interpreted to mean the one or two lines of  
 6435 code that actually perform a comparison, zeroing operation, etc. that implements a requirement. A  
 6436 broader interpretation might be that it includes the module that is invoked in response to a SFR-  
 6437 enforcing TSFI, and all modules that may be invoked in turn by that module (and so on until the  
 6438 completion of the call). Neither of these interpretations is particularly satisfying, since the narrowness  
 6439 of the first interpretation may lead to important modules being incorrectly categorised as SFR  
 6440 supporting, while the second leads to modules that are actually not SFR-enforcing being classified as  
 6441 such.

6442 A description of a module should be such that one could create an implementation of the module from  
 6443 the description, and the resulting implementation would be 1) identical to the actual TSF  
 6444 implementation in terms of the interfaces presented, 2) identical in the use of interfaces that are  
 6445 mentioned in the design, and 3) functionally equivalent to the description of the purpose of the TSF  
 6446 module. For instance, RFC 793 provides a high-level description of the TCP protocol. It is necessarily  
 6447 implementation independent. While it provides a wealth of detail, it is **not** a suitable design  
 6448 description because it is not specific to an implementation. An actual implementation can add to the  
 6449 protocol specified in the RFC, and implementation choices (for example, the use of global data vs. local  
 6450 data in various parts of the implementation) may have an impact on the analysis that is performed.  
 6451 The design description of the TCP module would list the interfaces presented by the implementation  
 6452 (rather than just those defined in RFC 793), as well as an algorithm description of the processing  
 6453 associated with the modules implementing TCP (assuming they were part of the TSF).

6454 In the design, modules are described in detail in terms of the function they provide (the purpose); the  
 6455 interfaces they present (when required by the criteria); the return values from such interfaces; the  
 6456 interfaces (presented by other modules) they use (provided those interfaces are required to be also  
 6457 described); and a description of how they provide their functionality using a technique appropriate to  
 6458 the method used to implement the module.

6459 The purpose of a module should be described indicating what function the module is providing. It  
 6460 should be sufficient so that the reader could get a general idea of what the module's function is in the  
 6461 architecture.

6462 The interfaces presented by a module are those interfaces used by other modules to invoke the  
 6463 functionality provided. Interfaces include both *explicit* interfaces (e.g., a calling sequence invoked by  
 6464 other modules) as well as *implicit* interfaces (e.g., global data manipulated by the module). Interfaces  
 6465 are described in terms of how they are invoked, and any values that are returned. This description  
 6466 would include a list of parameters, and descriptions of these parameters. If a parameter were expected  
 6467 to take on a set of values (e.g., a "flag" parameter), the complete set of values the parameter could take  
 6468 on that would have an effect on module processing would be specified. Likewise, parameters  
 6469 representing data structures are described such that each field of the data structure is identified and  
 6470 described. Global data should be described to the extent required to understand their purpose. The  
 6471 level of description required for a global data structure needs to be identical to the one for module  
 6472 interfaces, where the input parameter and return values correspond to the individual fields and their  
 6473 possible values in the data structure. Global data structures may be described separate from the  
 6474 modules that manipulate or read them as long as the design of the modules contain sufficient  
 6475 information about the global data structures updated or the information extracted from global data  
 6476 structures.

6477 Note that different programming languages may have additional "interfaces" that would be non-  
 6478 obvious; an example would be operator/function overloading in C++. This "implicit interface" in the  
 6479 class description would also be described as part of the module design. Note that although a module  
 6480 could present only one interface, it is more common that a module presents a small set of related  
 6481 interfaces.

6482 When it is required to describe the interfaces used by a module, it must be clear from either the design  
 6483 description of the module or the purpose of the module called, what service is expected from the  
 6484 module called. For example if Module A is being described, and it uses Module B's bubble sort routine,  
 6485 the description of the interaction between modules must allow to identify why Module B's bubble sort  
 6486 routine is called and what this call contributes to the implementation of the SFRs. The interface and  
 6487 purpose of Module B's bubble sort routine must be described as part of the interfaces of Module B  
 6488 (provided the level of ADV\_TDS and the classification of Module B require a description its interfaces)  
 6489 and so Module A just needs to identify what data it needs to have sorted using this routine. An  
 6490 adequate description would be: "Module A invokes Module B's interface *double\_bubble()* to sort the  
 6491 usernames in alphabetical order".

6492 Note that if this sorting of the user names is not important for the enforcement of any SFR (e. g. it is  
 6493 just done to speed up things and an algorithmically identical implementation of Module A could also  
 6494 avoid to have the usernames sorted), the use of Module B's bubble sort routine is not SFR-enforcing  
 6495 and it is sufficient to explain in the description of Module A that the usernames are sorted in  
 6496 alphabetical order to enhance performance. Module B may be classified as "SFR-supporting" only and  
 6497 the level of ADV\_TDS chosen indicates if the interfaces of SFR-supporting modules need to be  
 6498 described or if it is sufficient to just describe the purpose of Module B.

6499 As discussed previously, the algorithmic description of the module should describe in an algorithmic  
 6500 fashion the implementation of the module. This can be done in pseudo-code, through flow charts, or  
 6501 (at ADV\_TDS.3 Basic modular design) informal text. It discusses how the module inputs and called  
 6502 functions are used to accomplish the module's function. It notes changes to global data, system state,  
 6503 and return values produced by the module. It is at the level of detail that an implementation could be  
 6504 derived that would be very similar to the actual implementation of the TOE.

6505 It should be noted that source code does not meet the module documentation requirements. Although  
 6506 the module design describes the implementation, it is *not* the implementation. The comments  
 6507 surrounding the source code might be sufficient documentation if they provide an explanation of the  
 6508 intent of the source code. In-line comments that merely state what each line of code is doing are  
 6509 useless because they provide no explanation of what the module is meant to accomplish.

6510 In the elements below, the labels (SFR-enforcing, SFR-supporting, and SFR-non-interfering) discussed  
 6511 for subsystems and modules are used to describe the amount and type of information that needs to be  
 6512 made available by the developer. The elements have been structured so that there is no expectation  
 6513 that the developer provide *only* the information specified. That is, if the developer's documentation of  
 6514 the TSF provides the information in the requirements below, there is no expectation that the  
 6515 developer update their documentation and label subsystems and modules as SFR-enforcing, SFR-  
 6516 supporting or SFR-non-interfering. The primary purpose of this labelling is to allow developers with  
 6517 less mature development methodologies (and associated artifacts, such as detailed interface and  
 6518 design documentation) to provide the necessary evidence without undue cost.

#### 6519 **A.4.3 Levelling Approach**

6520 Because there is subjectivity in determining what is SFR-enforcing vs. SFR-supporting (and in some  
 6521 cases, even determining what is SFR-non-interfering the following paradigm has been adopted in this  
 6522 family. In early components of the family, the developer makes a determination about the  
 6523 classification of the subsystems into SFR-enforcing, etc., supplying the appropriate information, and  
 6524 there is little additional evidence for the evaluator to examine to support this claim. As the level of  
 6525 desired assurance increases, while the developer still makes a classification determination, the  
 6526 evaluator obtains more and more evidence that is used to confirm the developer's classification.

6527 In order to focus the evaluator's analysis on the SFR-related portions of the TOE, especially at lower  
 6528 levels of assurance, the components of the family are levelled such that initially detailed information is  
 6529 required only for SFR-enforcing architectural entities. As the level of assurance increases, more  
 6530 information is required for SFR-supporting and (eventually) SFR-non-interfering entities. It should be  
 6531 noted that even when complete information is required, it is not required that all of this information  
 6532 be analysed in the same level of detail. The focus should be in all cases on whether the *necessary*  
 6533 information has been provided and analysed.

6534 Table A.1 summarises the information required at each of the family components for the architectural  
 6535 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>(1)</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>(2)</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>(3)</sup>	purpose, all interfaces	purpose, all interfaces
ADV_TDS.6 Complete semiformal modular design with formal high-level design presentation (semiformal presentation; additional formal presentation)	description, interactions	description, interactions	description, interactions	purpose, all interfaces	purpose, all interfaces	purpose, all interfaces

<sup>(1)</sup> *designation support* means that only documentation sufficient to support the classification of the subsystem / module is needed.

<sup>(2)</sup> *SFR interfaces* means that the module description contains, for each SFR-related interface, the returned values and the called interfaces to other modules.



6540 <sup>(3)</sup> *All interfaces* means that the module description contains, for each interface, the returned values and the called  
 6541 interfaces to other modules.

## 6542 Table A.1 — Description Detail Levelling

### 6543 A.4.4 Security relevance

6544 The comments to WD2 regarding this chapter are pending as the contributor was not able to answer the  
 6545 comments until the deadline.

6546 ISO/IEC 15408 (all parts) concentrates the description, the evidence and the analysis on the security  
 6547 functionality of the TOE. This requires characterization of security relevance of functional and physical  
 6548 parts of the TOE. Interfaces, subsystems and modules may be categorised (either implicitly or  
 6549 explicitly) as “SFR-enforcing”, “SFR-supporting”, or “SFR-noninterfering”.

6550 The developer evidence and the evaluation analysis relates to the TOE and focus on the TSF and its  
 6551 SFR-enforcing and SFR-supporting implementation. The security architecture description shall  
 6552 demonstrate that the identified non-TSF subsystems of the TOE are not bypassing the TSF and the TSF  
 6553 protects themselves against corruption by non-TSF code or entities. The developer shall describe the  
 6554 SFR-noninterfering interfaces, subsystems and modules in the TOE design and demonstrate that they  
 6555 do not interfere with the TSF because of their purposes, interactions or separation of resources.

6556 An interface, subsystem or module is

- 6557 • SFR-enforcing, if it directly implements an SFR.
- 6558 • SFR-supporting if it has to operate functionally correctly in order to support the proper  
 6559 function of the SFRs.
- 6560 • SFR-non-interfering if it is not related to the implementation of the SFRs.

6561 The focus on security enforcing and security supporting functionality requires evidence of non-  
 6562 interference of the other functionality. Even correct implemented security enforcing functions and  
 6563 security mechanisms may be **bypassed, circumvented, deactivated, corrupted, or directly**  
 6564 **attacked**. Non-interference implies that the TSF cannot be misused and unauthorized access to the  
 6565 resources of the TSF implementation is prevented or impossible. Therefore the security architecture  
 6566 aspects of non-bypassability and self-protection are critical if security relevance of interfaces,  
 6567 subsystems and modules is categorized and this categorization is used in the vulnerability analysis.

6568 TSF self-protection is the security architecture property whereby the TSF cannot be corrupted by non-  
 6569 TSF code or entities. This includes non-TSF subsystems of TOE and non-TOE parts of the IT product. It  
 6570 is similar to the evidence for SFR-non-interfering subsystems/modules.

6571 The security domains are environments provided by the TSF for the use by untrusted entities in such a  
 6572 way that these environments are isolated and protected from each other. It is similar to the needed  
 6573 separation between “SFR-enforcing/supporting environments (resources)” and “SFR-non-interfering  
 6574 environments (entities)”.

6575 Therefore the analysis of non-interference during evaluation requires examination of the security  
 6576 architecture of the TOE (ADV\_ARC) and may need more information on non-TSF subsystems than only  
 6577 the TOE structure in terms of subsystems as provided for ADV\_TDS.x.1. The developers shall provide a  
 6578 rationale that TSF is correctly defined and the analysis of SFR-non-interfering module in terms of its  
 6579 purpose and interaction with other modules

6580 • **purpose:** how a module provides their functionality, no further design decisions are needed.

6581 • **interaction:** reason that subsystems or modules communicate, and characterizes the  
6582 information that is passed (less details than for interfaces).

6583 During evaluation non-interference shall be analyzed as part of the examination of functional  
6584 specification and TOE design, and the vulnerability analysis. The categorization of interfaces,  
6585 subsystems and modules as SFR-enforcing, SFR-supporting and SFR-noninterfering implies specific  
6586 examination of the functional specification, design and testing. An interpretation of TSFI as all  
6587 accessible external interfaces of the TSF would help this analysis. The functional tests of all TSF  
6588 subsystems (beginning with ATE\_DPT.1) and all TSF modules (ATE\_DPT.3 and higher) should provide  
6589 evidence for the correctness of their security categorization.

## 6590 A.5 Supplementary material on formal methods

6591 Formal methods provide a mathematical representation of the TSF and its behaviour and are required  
6592 by the ADV\_FSP.6 Complete semi-formal functional specification with additional formal specification,  
6593 ADV\_SPM.1 Formal TOE security policy model, and ADV\_TDS.6 Complete semiformal modular design  
6594 with formal high-level design presentation components. There are two aspects of formal methods: the  
6595 *specification language* that is used for formal expression, and the *theorem prover* that mathematically  
6596 proves the completeness and correctness of the formal specification.

6597 A formal specification is expressed within a formal system based upon well-established mathematical  
6598 concepts. These mathematical concepts are used to define well-defined semantics, syntax and rules of  
6599 inference. A formal system is an abstract system of identities and relations that can be described by  
6600 specifying a formal alphabet, a formal language over that alphabet which is based on a formal syntax,  
6601 and a set of formal rules of inference for constructing derivations of sentences in the formal language.

6602 The evaluator should examine the identified formal systems to make sure that:

6603 • The semantics, syntax and inference rules of the formal system are defined or a definition is  
6604 referenced.

6605 • Each formal system is accompanied by explanatory text that provides defined **semantics** so that:

6606 1) the explanatory text provides defined meanings of terms, abbreviations and acronyms that  
6607 are used in a context other than that accepted by normal usage;

6608 2) the use of a formal system and semiformal notation use is accompanied by supporting  
6609 explanatory text in informal style appropriate for unambiguous meaning;

6610 3) the formal system is able to express rules and characteristics of applicable SFPs, security  
6611 functionality and interfaces (providing details of effects, exceptions and error messages) of  
6612 TSF, their subsystems or modules to be specified for the assurance family for which the  
6613 notations are used;

6614 4) the notation provides rules to determine the meaning of syntactical valid constructs.

6615 • Each formal system uses a formal syntax that provides rules to unambiguously recognise  
6616 constructs.

6617 • Each formal system provides proof rules which

6618 5) support logical reasoning of well-established mathematical concepts,

6619 6) help to prevent derivation of contradictions.

6620 If the developer uses a formal system which is already accepted by the evaluation authority the  
 6621 evaluator can rely on the level of formality and strength of the system and focus on the instantiation of  
 6622 the formal system to the TOE specifications and correspondence proofs.

6623 The formal style supports mathematical proofs of the security properties based on the security  
 6624 features, the consistency of refinements and the correspondence of the representations. Formal tool  
 6625 support seems adequate whenever manual derivations would otherwise become long winded and  
 6626 incomprehensible. Formal tools are also apt to reduce the error probability inherent in manual  
 6627 derivations.

6628 Examples of formal systems:

- 6629 • The **Z specification language** is highly expressive, and supports many different methods or styles  
 6630 of formal specification. The use of Z has been predominantly for model-oriented specification,  
 6631 using *schemas* to formally specify operations. See [https://en.wikipedia.org/wiki/Z\\_notation](https://en.wikipedia.org/wiki/Z_notation) for  
 6632 more information.
- 6633 • **ACL2** is an open-source formal system comprising a LISP-based specification language and a  
 6634 theorem prover. See <http://www.cs.utexas.edu/users/moore/acl2/> for further information.
- 6635 • **Isabelle** is a popular generic theorem proving environment that allows mathematical formulae to  
 6636 be expressed in a formal language and provides tools for proving those formulae within a logical  
 6637 calculus (see e.g. <http://www.cl.cam.ac.uk/Research/HVG/Isabelle/> for additional information).
- 6638 • The **B method** is a formal system based on the propositional calculus, the first order predicate  
 6639 calculus with inference rules and set theory (see e.g. <https://en.wikipedia.org/wiki/B-Method> for  
 6640 further information).
- 6641 • **NuSMV** (based on its predecessor SMV) is a symbolic model checker designed to be an open  
 6642 architecture for model checking which can be reliably used for the verification of industrial  
 6643 designs, as a core for custom verification tools, and as a testbed for formal verification techniques.  
 6644 See <http://nusmv.fbk.eu/> for more information.
- 6645 • **Coq** is a formal proof management system that provides a formal language to write mathematical  
 6646 definitions, executable algorithms and theorems together with an environment for semi-  
 6647 interactive development of machine-checked proofs. See <https://coq.inria.fr/> for more  
 6648 information.
- 6649 • **SystemVerilog** is a combined hardware description language and hardware verification language  
 6650 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.

6692 The composed evaluation activities may take place at the same time as the dependent component  
6693 evaluation. This is due to two factors:

6694 a) Economic/business drivers - the dependent component developer will either be sponsoring the  
6695 composition evaluation activities or supporting these activities as the evaluation deliverables from  
6696 the dependent component evaluation are required for composed evaluation activities.

6697 b) Technical drivers - the components consider whether the requisite assurance is provided by the  
6698 base component (e.g. considering the changes to the base component since completion of the  
6699 component evaluation) with the understanding that the dependent component has recently  
6700 undergone (is undergoing) component evaluation and all evaluation deliverables associated with  
6701 the evaluation are available. Therefore, there are no activities during composition requesting the  
6702 dependent component evaluation activities to be re-verified. Also, it is verified that the base  
6703 component forms (one of) the test configurations for the testing of the dependent component  
6704 during the dependent component evaluation, leaving ACO\_CTT to consider the base component in  
6705 this configuration.

6706 The evaluation evidence from the evaluation of the dependent component is required input into the  
6707 composed TOE evaluation activities. The only evaluation material from the evaluation of the base  
6708 component that is required as input into the composed TOE evaluation activities:

6709 a) Residual vulnerabilities in the base component, as reported during the base component  
6710 evaluation. This is required for the ACO\_VUL activities.

6711 No other evaluation evidence from the base component activities should be required for the composed  
6712 TOE evaluation, as the evaluation results from the component evaluation of the base component  
6713 should be reused. Additional information about the base component may be required if the composed  
6714 TOE TSF includes more of the base component than was considered to be TSF during component  
6715 evaluation of the base component.

6716 The component evaluation of the base and dependent components are assumed to be complete by the  
6717 time final verdicts are assigned for the ACO components.

6718 The ACO\_VUL components only consider resistance against an attacker with an attack potential up to  
6719 Enhanced-Basic. This is due to the level of design information that can be provided of how the base  
6720 component provides the services on which the dependent component relies through application of the  
6721 ACO\_DEV activities. Therefore, the confidence arising from composed TOE evaluations using CAPs is  
6722 limited to a level similar to that obtained from EAL4 component TOE evaluations. Although assurance  
6723 in the components that comprise the composed TOE may be higher than EAL4.

## 6724 **B.2 Performing Security Target evaluation for a composed TOE**

6725 An ST will be submitted by the developer for the evaluation of the composed (base component +  
6726 dependent component) TOE. This ST will identify the assurance package to be applied to the  
6727 composed TOE, providing assurance in the composed entity by drawing upon the assurance gained in  
6728 the component evaluations.

6729 The purpose of considering the composition of components within an ST is to validate the  
6730 compatibility of the components from the point of view of both the environment and the requirements,  
6731 and also to assess that the composed TOE ST is consistent with the component STs and the security  
6732 policies expressed within them. This includes determining that the component STs and the security  
6733 policies expressed within them are compatible.

6734 The composed TOE ST may refer out to the content of the component STs, or the ST author may chose  
 6735 to reiterate the material of the component STs within the composed TOE ST providing a rationale of  
 6736 how the component STs are represented in the composed TOE ST.

6737 During the conduct of the ASE\_CCL evaluation activities for a composed TOE ST the evaluator  
 6738 determines that the component STs are accurately represented in the composed TOE ST. This is  
 6739 achieved through determining that the composed TOE ST demonstrably conforms to the component  
 6740 TOE STs. Also, the evaluator will need to determine that the dependencies of the dependent  
 6741 component on the operational environment are adequately fulfilled in the composed TOE.

6742 The composed TOE description will describe the composed solution. The logical and physical scope  
 6743 and boundary of the composed solution will be described, and the logical boundary(ies) between the  
 6744 components will also be identified. The description will identify the security functionality to be  
 6745 provided by each component.

6746 The statement of SFRs for the composed TOE will identify which component is to satisfy an SFR. If an  
 6747 SFR is met by both components, then the statement will identify which component meets the different  
 6748 aspects of the SFR. Similarly the composed TOE Summary Specification will identify which component  
 6749 provides the security functionality described.

6750 The package of ASE: Security Target evaluation requirements applied to the composed TOE ST should  
 6751 be consistent with the package of ASE: Security Target evaluation requirements used in the  
 6752 component evaluations.

6753 Reuse of evaluation results from the evaluation of component STs can be made in the instances that  
 6754 the composed TOE ST directly refers to the component STs. e.g. if the composed TOE ST refers to a  
 6755 component ST for part of its statement of SFRs, the evaluator can understand that the requirement for  
 6756 the completion of all assignment and selection operations (as stated in ASE\_REQ.\*.3C has been  
 6757 satisfied in the component evaluations.

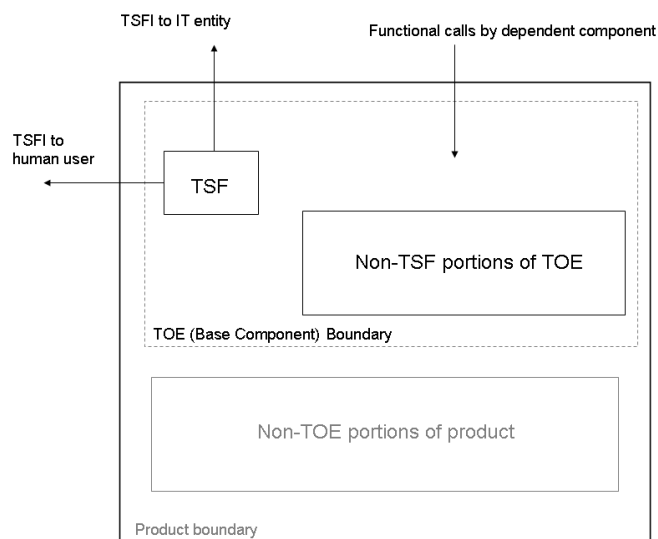
### 6758 **B.3 Interactions between composed IT entities**

6759 The TSF of the base component is often defined without knowledge of the dependencies of the  
 6760 possible applications with which it may be composed. The TSF of this base component is defined to  
 6761 include all parts of the base component that have to be relied upon for enforcement of the base  
 6762 component SFRs. This will include all parts of the base component required to implement the base  
 6763 component SFRs.

6764 The TSFI of this base component represents the interfaces provided by the TSF to the external entities  
 6765 defined in the statement of SFRs to invoke a service of the TSF. This includes interfaces to the human  
 6766 user and also interfaces to external IT entities. However, the TSFI only includes those interfaces to the  
 6767 TSF, and therefore is not necessarily an exhaustive interface specification of all possible interfaces  
 6768 available between an external entity and the base component. The base component may present  
 6769 interfaces to services that were not considered security-relevant, either because of the inherent  
 6770 purpose of the service (e.g., adjust type font) or because associated ISO/IEC 15408 SFRs are not being  
 6771 claimed in the base component's ST (e.g. the login interface when no FIA: Identification and  
 6772 authentication SFRs are claimed).

6773 The functional interfaces provided by the base component are in addition to the security interfaces  
 6774 (TSFIs), and are not required to be considered during the base component evaluation. These often  
 6775 include interfaces that are used by a dependent component to invoke a service provided by the base  
 6776 component.

6777 The base component may include some indirect interfaces through which TSFIs may be called, e.g.  
 6778 APIs that can be used to invoke a service of the TSF, which were not considered during the evaluation  
 6779 of the base component.



6780

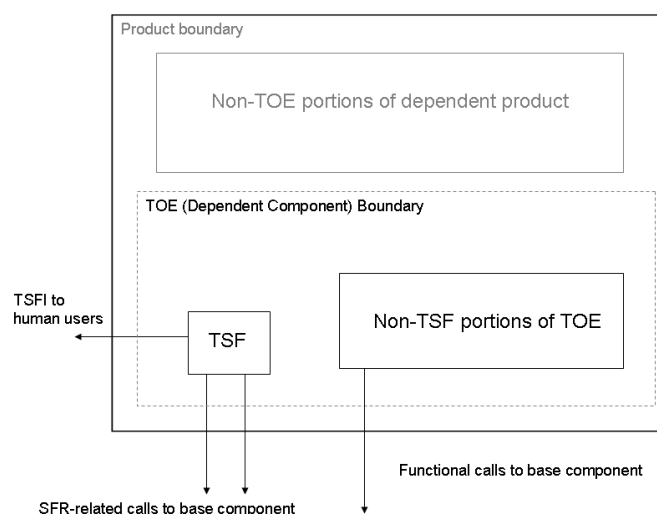
6781

**Figure B.1 — Base component abstraction**

6782 The dependent component, which relies on the base component, is similarly defined: interfaces to  
 6783 external entities defined in the SFRs of the component ST are categorised as TSFI and are examined in  
 6784 ADV\_FSP.

6785 Any call out from the dependent TSF to the environment in support of an SFR will indicate that the  
 6786 dependent TSF requires some service from the environment in order to satisfy the enforcement of the  
 6787 stated dependent component SFRs. Such a service is outside the dependent component boundary and  
 6788 the base component is unlikely to be defined in the dependent ST as an external entity. Hence, the calls  
 6789 for services made out by the dependent TSF to its underlying platform (the base component) will not  
 6790 be analysed as part of the Functional specification (ADV\_FSP) activities. These dependencies on the  
 6791 base component are expressed in the dependent component ST as security objectives for the  
 6792 environment.

6793 This abstraction of the dependent component and the interfaces is shown in Figure B.2 below.



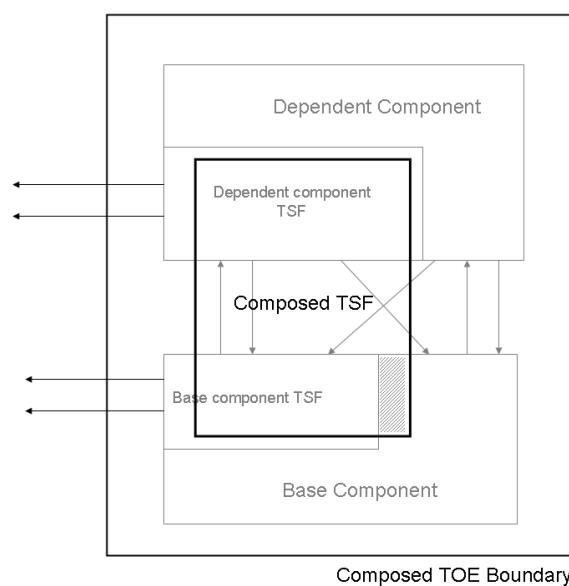
**Figure B.2 — Dependent component abstraction**

When considering the composition of the base component and the dependent component, if the dependent component's TSF requires services from the base component to support the implementation of the SFR, the interface to the service will need to be defined. If that service is provided by the base component's TSF, then that interface should be a TSFI of the base component and will therefore already be defined within the functional specification of the base component.

If, however, the service called by the dependent component's TSF is not provided by the TSF of the base component (i.e., it is implemented in the non-TSF portion of the base component or possibly even in the non-TOE portion of the base component (not illustrated in Figure B.3), there is unlikely to be a TSFI of the base component relating to the service, unless the service is mediated by the TSF of the base component. The interfaces to these services from the dependent component to the operational environment are considered in the family Reliance of dependent component (ACO\_REL).

The non-TSF portion of the base component is drawn into the TSF of the composed TOE due to the dependencies the dependent component has on the base component to support the SFRs of the dependent component. Therefore, in such cases, the TSF of the composed TOE would be larger than simply the sum of the components' TSFs.

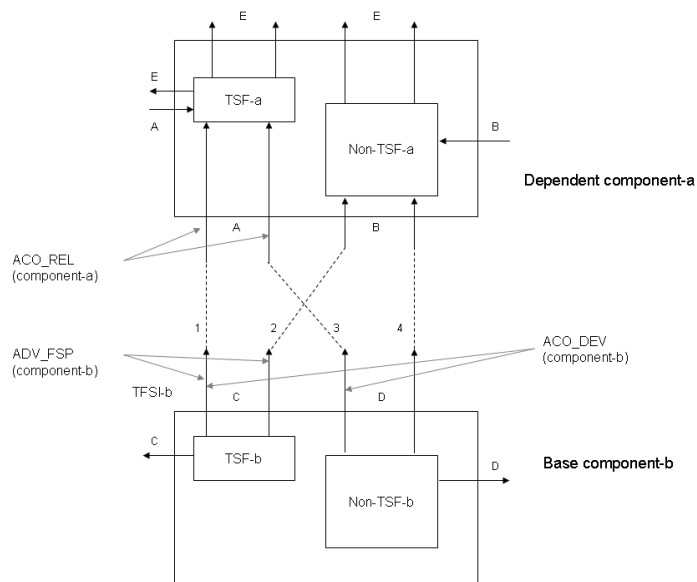




**Figure B.3 — Composed TOE abstraction**

It may be the case that the base component TSFI is being called in a manner that was unforeseen in the base component evaluation. Hence there would be a requirement for further testing of the base component TSFI.

The possible interfaces are further described in the following diagram (Figure B.4) and supporting text.



**Figure B.4 — Composed component interfaces**

- a) Arrows going *into* 'dependent component-a' (A and B) = where the component expects the environment to respond to a service request (responding to calls out from dependent component to the environment);

- 6823 b) Arrows coming *out* of 'base component-b' (C and D) = interfaces of services provided by the base  
6824 component to the environment;
- 6825 c) Broken lines between components = types of communication between pairs of interfaces;
- 6826 d) The other (grey) arrows = interfaces that are described by the given criteria.
- 6827 The following is a simplification, but explains the considerations that need to be made.
- 6828 There are components a ('dependent component-a') and b ('base component-b'): the arrows coming  
6829 *out* of TSF-a are services provided by TSF-a and are therefore TSFIs(a); likewise, the arrows coming  
6830 *out* of TSF-b ("C") are TSFIs(b). These are each detailed in their respective functional specs.  
6831 component-a is such that it requires services from its environment: those needed by the TSF(a) are  
6832 labelled "A"; the other (not related to TSF-a) services are labelled "B".
- 6833 When component-a and component-b are combined, there are four possible combinations of {services  
6834 needed by component-a} and {services provided by component-b}, shown as broken lines (types of  
6835 communication between pairs of interfaces). Any set of these might exist for a particular composition:
- 6836 a) TSF-a needs those services that are provided by TSF-b ("A" is connected to "C"): this is  
6837 straightforward: the details about "C" are in the FSP for component-b. In this instance the  
6838 interfaces should all be defined in the functional specifications for the component-b.
- 6839 b) Non-TSF-a needs those services that are provided by TSF-b ("B" is connected to "C"): this is  
6840 straightforward (again, the details about "C" are in the FSP for component-b), but unimportant:  
6841 security-wise.
- 6842 c) Non-TSF-a needs those services that are provided by non-TSF-b ("B" is connected to "D"): we have  
6843 no details about D, but there are no security implications about the use of these interfaces, so they  
6844 do not need to be considered in the evaluation, although they are likely to be an integration issue  
6845 for the developer.
- 6846 d) TSF-a needs those services that are provided by non-TSF-b ("A" is connected to "D"): this would  
6847 arise when component-a and component-b have different senses of what a "security service" is.  
6848 Perhaps component-b is making no claims about I&A (has no FIA SFRs in its ST), but component-a  
6849 needs authentication provided by its environment. There are no details about the "D" interfaces  
6850 available (they are not TSFI (b), so they are not in component-b's FSP).
- 6851 Note: if the kind of interaction described in case d above exists, then the TSF of the composed TOE  
6852 would be TSF-a + TSF-b + Non-TSF-b. Otherwise, the TSF of the composed TOE would be TSF-a + TSF-  
6853 b.
- 6854 Interfaces types 2 and 4 of Figure B.4 are not directly relevant to the evaluation of the composed TOE.  
6855 Interfaces 1 and 3 will be considered during the application of different families:
- 6856 a) Functional specification (ADV\_FSP) (for component-b) will describe the C interfaces.
- 6857 b) Reliance of dependent component (ACO\_REL) will describe the A interfaces.
- 6858 c) Development evidence (ACO\_DEV) will describe the C interfaces for connection type 1 and the D  
6859 interfaces for connection type 3.
- 6860 A typical example where composition may be applied is a database management system (DBMS) that  
6861 relies upon its underlying operating system (OS). During the evaluation of the DBMS component, there

6862 will be an assessment made of the security properties of that DBMS (to whatever degree of rigour is  
 6863 dictated by the assurance components used in the evaluation): its TSF boundary will be identified, its  
 6864 functional specification will be assessed to determine whether it describes the interfaces to the  
 6865 security services provided by the TSF, perhaps additional information about the TSF (its design,  
 6866 architecture, internal structure) will be provided, the TSF will be tested, aspects of its life-cycle and its  
 6867 guidance documentation will be assessed, etc.

6868 However, the DBMS evaluation will not call for any evidence concerning the dependency the DBMS has  
 6869 on the OS. The ST of the DBMS will most likely state assumptions about the OS in its Assumptions  
 6870 subclause and state security objectives for the OS in its Environment subclause. The DBMS ST may  
 6871 even instantiate those objectives for the environment in terms of SFRs for the OS. However, there will  
 6872 be no specification for the OS that mirrors the detail in the functional specification, architecture  
 6873 description, or other ADV evidence as for the DBMS. Reliance of dependent component (ACO\_REL) will  
 6874 fulfil that need.

6875 Reliance of dependent component (ACO\_REL) describes the interfaces of the dependent TOE that make  
 6876 the calls to the base component for the provision of services. These are the interfaces to which the  
 6877 base component is to respond. The interface descriptions are provided from the dependent  
 6878 component's viewpoint.

6879 Development evidence (ACO\_DEV) describes the interfaces provided by the base component, which  
 6880 respond to the dependent component service requests. These interfaces are mapped to the relevant  
 6881 dependent component interfaces that are identified in the reliance information. (The completeness of  
 6882 this mapping, whether the base component interfaces described represent all dependent component  
 6883 interfaces, is not verified here, but in Composition rationale (ACO\_COR)). At the higher levels of  
 6884 ACO\_DEV the subsystems providing the interfaces are described.

6885 Any interfaces required by the dependent component that have not been described for the base  
 6886 component are reported in the rationale for Composition rationale (ACO\_COR). The rationale also  
 6887 reports whether the interfaces of the base component on which the dependent component relies were  
 6888 considered within the base component evaluation. For any interfaces that were not considered in the  
 6889 base component evaluation, a rationale is provided of the impact of using the interface on the base  
 6890 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") or indirectly required (indicated by a dash "-"), by the row label component. If no character is presented, the component is not dependent upon another component.

	ADV_ARC.1	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	AGD_OPE.1	AGD_PRE.1	ALC_CMC.5	ALC_CMS.1	ALC_DVS.2	ALC_LCD.1	ALC_TAT.1
ADV_ARC.1		X	-						X								
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				-								
ADV_TDS.1			X						-								
ADV_TDS.2			-	X					-								
ADV_TDS.3			-		X				-								
ADV_TDS.4			-		-	X		-	-	-							-
ADV_TDS.5			-		-	X		-	-	-							-
ADV_TDS.6			-		-		X	-	-	-							-
ADV_COMP.1																	

**Table C.1 — Dependency table for Class ADV: Development**

	ADV_FSP.1
AGD_OPE.1	X
AGD_PRE.1	

**Table C.2 — Dependency table for Class AGD: Guidance documents**

	ADV_FSP.2	ADV_FSP.4	ADV_IMP.1	ADV_TDS.1	ADV_TDS.3	ALC_CMS.1	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_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_MOK.1										
ALC_COMP.1										

**Table C.3 — Dependency table for Class ALC: Life-cycle support**

	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_SPD.1	APE_REQ.1	APE_OBJ.2	APE_INT.1	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_SPD.1	ACE_REQ.1	ACE_OBJ.1	ACE_MCO.2	ACE_INT.1	ACE_ECD.1
ACE_CCL.1		X	-		X	X
ACE_CCO.1	-	X	-	X	X	-
ACE_ECD.1						
ACE_INT.1						
ACE_MCO.1	X	X	X		X	-
ACE_OBJ.1						
ACE_REQ.1			X			X
ACE_SPD.1						

**Table C.5 — Dependency table for Class ACE: Protection Profile Configuration evaluation**

	ASE_SPD.1	ASE_REQ.1	ASE_OBJ.2	ASE_INT.1	ASE_ECD.1	ADV_TDS.1	ADV_FSP.2	ADV_FSP.1	ADV_ARC.1
ASE_CCL.1		X		X	X				
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
ASE_COMP.1									

**Table C.6 — Dependency table for Class ASE: Security Target evaluation**

	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_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
ATE_COMP.1																

**Table C.7 Dependency table for Class ATE: Tests**

	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_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	-
AVA_COMP.1															

**Table C.8 Dependency table for Class AVA: Vulnerability assessment**

## Annex D (normative)

### Composite evaluation activities

Editors note:

This annex refers to the new SAR that addresses the composition model. It should be discussed whether the rules in this annex should better be integrated into 18045.

This annex aims to provide further information to the SAR that has been integrated in this part of ISO/IEC 15408.

#### D.1 Composite evaluation activities description

The current approach can be applied independent of the evaluation assurance level (EAL) for the composite product aimed. Where some evaluation activities are not applicable due to the EAL chosen, they are also not expected to be applied.

For the following paragraphs, we assume that the level of assurance of the platform is equivalent or higher compared to the composite product evaluation level.

Other cases must be discussed within the schemes.

##### D.1.1 Evaluation of the composite product Security Target

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

The Composite Product Evaluator has to examine that the Security Target of the composite product<sup>10</sup> does not contradict the Security Target of the underlying platform<sup>11</sup>. In particular, it means that the Composite Product Evaluator has to examine the Composite- and the Platform- Security Target for any conflicting assumptions, compatibility of security objectives, security requirements and security functionality needed by the application.

This task can be reduced, if some matching has been checked for Protection Profiles claimed by each Security Target.

The Composite Product Evaluation Sponsor must ensure that the security target of the platform is available to the Application Developer, to the Composite Product Evaluator and to the Composite Product Certification Body. The information available in public version of the security target may not be sufficient.

<sup>10</sup> denoted by Composite-ST in the following

<sup>11</sup> denoted by Platform-ST in the following



## 6950 **D.1.2 Integration of the application in the configuration management system**

6951 The Composite Product Evaluator shall verify that the evaluated version of the application has been  
6952 installed onto / embedded into the evaluated version of the underlying platform.

6953 The Composite Product Evaluation Sponsor must ensure that appropriate evidence generated by the  
6954 Composite Product Integrator is available to the Composite Product Evaluator. This evidence may  
6955 include, amongst other, the configuration list of the Platform Developer provided within its  
6956 acknowledgement statement.

## 6957 **D.1.3 Compatibility check for delivery and acceptance procedures**

6958 The Composite Product Evaluator shall verify that delivery procedures of the Application and Platform  
6959 Developers are compatible with the acceptance procedure used by the Composite Product Integrator.

6960 The Composite Product Evaluator shall verify that all configuration parameters prescribed by the  
6961 Application and Platform Developers (e.g. pre-personalization data, pre-personalisation scripts) are  
6962 used by the Composite Product Integrator.

6963 The Composite Product Evaluation Sponsor must ensure that appropriate evidence generated by the  
6964 Composite Product Integrator is available to the Composite Product Evaluator. This evidence may  
6965 include, amongst other, the

6966                                   ➤ Element of evidence for the application reception, acceptance and  
6967                                   parameterisation by the Platform Developer (in form of acknowledgement  
6968                                   statement).

## 6969 **D.1.4 Compliance of designs**

6970 The Composite Product Evaluator shall verify that stipulations for the Application Developer imposed  
6971 by the Platform Developer in its certified user guidance and referenced in the platform certification  
6972 report are fulfilled by the composite product, i.e. have been taken into account by the Application  
6973 Developer.

6974 The Composite Product Evaluation Sponsor must ensure that the following are made available to the  
6975 Composite Product Evaluator:

- 6976                                   ➤ The platform-related user guidance,
- 6977                                   ➤ ETR for Composition prepared by the Platform Evaluator, see chapter D.2 'ETR  
6978                                   for composite evaluation',
- 6979                                   ➤ The Certification Report for the platform prepared by the Platform Certification  
6980                                   Body,
- 6981                                   ➤ A rationale for secure composite product implementation including evidence  
6982                                   prepared by the Application Developer.

## 6983 **D.1.5 Composite product functional testing**

6984 Some application functionality testing can only be performed on emulators, before its  
6985 embedding/integration onto the platform, as effectiveness of this testing (pass/fail) may not be visible  
6986 using the interfaces of the composite product. Nevertheless, functional testing of the composite  
6987 product shall be performed also on composite product samples according to description of the security

6988 functions of the Composite TOE and using the standard approach as required by the relevant  
 6989 assurance class. No additional developer's action is required here.

6990 Since the amount, the coverage and the depth of the functional tests of the platform have already been  
 6991 validated by the platform certificate, it is not necessary to re-perform these tasks in the composite  
 6992 evaluation. Please note that ETR for Composition (see chapter D.2 'ETR for composite evaluation')  
 6993 does not provide any information on functional testing for the platform.

6994 The Composite Product Evaluation Sponsor must ensure that the following is available to the  
 6995 Composite Product Evaluator:

6996 ➤ Composite product samples suitable for testing.

#### 6997 **D.1.6 Composite product vulnerability analysis**

6998 The Composite Product Evaluator shall perform a vulnerability analysis for the composite product  
 6999 using, amongst other, the results of the platform evaluation and certification. This vulnerability  
 7000 analysis shall be confirmed by penetration testing.

7001 The Composite Product Evaluator has to check that the confidentiality protection of the embedded  
 7002 software in memory of the platform is consistent with the confidentiality level claimed by the  
 7003 Application Developer for ALC\_DVS.

7004 In special cases, the vulnerability analysis and the definition of attacks might be difficult, need  
 7005 considerable time and require extensive pre-testing, if only documentation is available. The platform  
 7006 may also be used in a way that was not foreseen by the Platform Developer and Platform Evaluator, or  
 7007 the Application Developer may not have followed the stipulations provided with the platform  
 7008 certification. Different possibilities exist to shorten composite vulnerability analysis in such cases:

7009 ➤ The Composite Product Evaluator can consult the Platform Evaluator and draw  
 7010 on his experience gained during the platform evaluation.

7011 ➤ Separation of vulnerabilities of application and platform with the use of "open  
 7012 samples" ("open samples" are samples of the platform on which the Composite  
 7013 Product Evaluator can load software on his own discretion). The intention is to  
 7014 use test software without the application countermeasures without  
 7015 deactivating any platform inherent countermeasure. The aim is clearly not to  
 7016 repeat the platform evaluation.

7017 The Composite Product Evaluation Sponsor must ensure that the following are made available to the  
 7018 Composite Product Evaluator:

7019 ➤ *The ETR for Composition* (ETR\_COMP) prepared by the Platform Evaluator, see  
 7020 chapter D.2 'ETR for composite evaluation' below, and

7021 ➤ The Certification Report for the platform prepared by the Platform Certification  
 7022 Body.

#### 7023 **D.1.7 Deliveries**

7024 The tables below summarize the documentation deliveries that are exchanged between parties to  
 7025 enable the composite evaluation activities as defined in the previous paragraphs.

7026 The Composite Product Evaluation Sponsor is in charge of the initialization of the process.

7027 The Composite Product Evaluation Sponsor is responsible for maintaining or creating any Non-  
 7028 Disclosure Agreement (NDA) that would be necessary between all the parties involved in the  
 7029 composition activities.

7030 The Non-Disclosure Agreement should be established according to the sensitivity and ownership of  
 7031 the information to be exchanged

# #	Document / Contribution	Description
1	Platform Security Target	Security Target of the platform as referenced in the platform certification report.
2	Platform open samples for testing	Platform samples as defined in [JIL AP] Chapter 3.8.
3	Platform user guidance	It encompasses all platform user guidance and manuals needed for the <b>Application Developer</b> and the <b>Composite Product Integrator</b> being referenced in the platform certification report.
4	Platform ETR_COMP	ETR for composition as defined in chapter D.2 and referenced in the platform certification report.
5	Platform certification report	Platform certification report issued by authorized <b>Platform Certification Body</b> .
6	Design compliance evidence	<p>It enfolds evidence elements on how the requirements on the application design, imposed by the platform's guidance and certification report, are fulfilled in the composite product.</p> <p>If such a requirement was not followed, a rationale that the chosen composite product implementation is still secure shall be given here.</p>

#	Document / Contribution	Description
7	Composite configuration evidence	<p>It comprises</p> <p>(i) Identification elements of the composite product</p> <ul style="list-style-type: none"> <li>- proving that the correct, certified version of the platform is used in the composite product,</li> <li>- proving that the correct, evaluated version of the application has been integrated;</li> </ul> <p>and</p> <p>(ii) Evidence elements that security measures prescribed by the <b>Platform</b> and <b>Application Developers</b> are actually being applied by the <b>Composite Product Integrator</b>.</p>
8	Delivery and acceptance procedures evidence	Evidence elements how the delivery procedures of the <b>Platform</b> and <b>Application Developers</b> are compatible with the acceptance procedure of the <b>Composite Product Integrator</b>

Table D1 - Definition of composition documents

The following table shows which documents/contributions of Table D1 shall be provided to which actor within the composite evaluation process:

		Actors				
##	Documents/contributions having to be provided to	Composite product evaluation Sponsor	Composite product integrator	Application developer	Composite product Evaluator	Composite product Certification Body
1	Platform Security Target	No	No	Yes	Yes	Yes
2	Platform open samples	No	No	No	Yes	No
3	Platform user guidance	No	Yes	Yes	Yes	Yes
4	Platform ETR_COMP	No	No	No	Yes	Yes
5	Platform certification report	Yes	Yes	Yes	Yes	Yes
6	Design compliance evidence	No	No	No	Yes	Yes

		Actors				
##	Documents/contributions having to be provided to	Composite product evaluation Sponsor	Composite product integrator	Application developer	Composite product Evaluator	Composite product Certification Body
7	Composite configuration evidence	No	No	No	Yes	Yes
8	Delivery and acceptance procedures evidence	No	No	No	Yes	Yes

7035

Table D2 - Main Deliveries between actors

7036 The next table shows some example of Composite TOE use cases with definition of the components  
7037 and the roles.

Composite TOE example			
Components & roles definitions	Smartcard -I	Smartcard -II	Java Card
	The Composite TOE is built of  - a Security IC with an application code loaded in ROM (Masking operation) and application data loaded in EEPROM.	The Composite TOE is built of  - a Security IC without ROM, but offering Flash technology and Flash loader  - an application code and data loaded into the flash by a smart Card manufacturer	The Composite TOE is built of  - a Java Card Platform  - a Java card application: the applet
	The <b>Platform</b> is	The Security IC with the Flash memory and the Flash Loader	The Java Card Platform including Card Manager with Applet loader facility
	The <b>Application</b> is	The Operating System code, Flash memory initialization data and application data	The Applet

Composite TOE example			
The <b>Platform Developer</b> is	The Security IC Manufacturer:  - Develops and manufactures the Security IC	The Security IC Manufacturer:  - Develops, manufactures and delivers the Security IC with Flash technology to the Composite Product Integrator	The Java Card Platform developer:  - Develops the Java Card with applet loading mechanism to the Composite Product Integrator.
The <b>Application Developer</b> is	The Smartcard Software developer:  - Develops the application;  - Provides the application to Composite product integrator	The Smartcard Software developer:  - Develops the application;  - Delivers the application to the Composite Product Integrator	The Applet developer:  - Develops the applet;  - Delivers the applet to the Composite Product Integrator
The <b>Composite Product Integrator</b> is	The Security IC Manufacturer:  - is in charge of OS masking in ROM and of loading Application data in EEPROM;  - Delivers the Composite TOE to be evaluated	The Card Manufacturer:  - is in charge of loading the application into the flash using Security IC flash loader;  - Delivers the Composite TOE to be evaluated	The Card Issuer:  - Loads the applet on the Java Card platform using applet loading mechanism;  - Delivers the Composite TOE to be evaluated

Table D3 - Example of composite TOE use cases

## D.2 ETR for composite evaluation

### D.2.1 Objective of the document

A standard Evaluation Technical Report (ETR) contains proprietary information that cannot be made public. The *ETR for composite evaluation* (ETR\_COMP) document is compiled from the ETR in order to provide sufficient information for composite product evaluation with a certified platform. The information that is presented in the ETR\_COMP document shall be a subset of the information presented in the full ETR. It should enable the Composite Product Evaluator and the respective Certification Body to understand the considered attack paths, the performed tests and the effectiveness of countermeasures implemented by the platform.

7048 **D.2.2 Generic rules:**

7049 The *ETR for composite evaluation* should be produced by the Platform Evaluator based on the platform  
7050 evaluation results. This task should be considered when determining the evaluation work program to  
7051 reduce additional cost and effort.

7052 The content of ETR\_COMP has to strike the right balance between protecting platform developer's  
7053 and/or Platform Evaluator's proprietary information and providing sufficient information for the  
7054 Composite Product Evaluator and the respective Certification Body, cf. Table D2 above.

7055 ETR\_COMP shall not include information affecting national security.

7056 The information provided must be approved by all parties involved in the platform evaluation (i.e. the  
7057 Evaluator, the Certification Body, the developer and sponsor of the evaluation). The platform  
7058 Certification Body shall validate its consistency with the original ETR. The platform certification report  
7059 shall reference the *ETR for composite evaluation*.

7060 If the current ETR\_COMP itself relies on a composite evaluation, and if there is direct interface with the  
7061 previous platform, the reference to this previous composite evaluation ETR\_COMP must be supplied.

7062 The ETR\_COMP is not meant to include copies of information from other available platform evidence,  
7063 as the Security Target and Guidance. However, the composite evaluation is much supported by  
7064 references to the relevant sections.

7065 **D.2.3 Exchange of the ETR for Composition**

7066 An ETR\_COMP contains intellectual property of the Platform Developer as well as of the Platform  
7067 Evaluator, and also the Platform Certification Body has a role in its content. At the minimum the  
7068 document should be considered restricted. The ETR\_COMP document is created and maintained by the  
7069 Platform Evaluator. However, at a given certification the Platform developer is the point of contact for  
7070 the Application Developer.

7071 The application developer will contact the Platform Developer for delivery of the ETR\_COMP to the  
7072 point of contact at the Composite Product Evaluator. The Platform Developer will check its  
7073 confidentiality management rules (existence of relevant NDA with Lab and CB, etc.) whether delivery  
7074 is possible. If necessary the platform developer will contact the Platform Certification Body about the  
7075 intent of the delivery of the ETR\_COMP.

7076 Next the Platform Developer will contact the Platform Evaluator to request the delivery (using a  
7077 secure method and only marked versions will be distributed) of the ETR\_COMP to the given contact  
7078 point of the Composite Product Evaluator. If the OK is granted, either the Platform Evaluator or the  
7079 Platform Developer will send the ETR\_COMP to the Composite Product Evaluator depending on the  
7080 agreements between these two parties.

7081 Depending on (contractual) agreement between the Platform Developer and Platform Evaluator, there  
7082 may be deviations from the described procedure of delivery of the ETR\_COMP to the Composite  
7083 Product Evaluator.

7084 If necessary the Platform Evaluator and the Composite Product Evaluator will exchange more detailed  
7085 information. This is always under control of the Platform Developer. In case of clarification the  
7086 Platform Evaluator and the Composite Product Evaluator will be the main parties. If an additional  
7087 assurance statement is required then also the Platform certification body will be involved in the  
7088 exchange.

7089

7090 **D.2.4 Content of the ETR for composite evaluation**

7091 The information required is focused on:

- 7092 1. Formal information about the platform like its exact identification, reference to the  
7093 certification report etc.
- 7094 2. Information about the Platform design.
- 7095 3. Information about the evaluated configuration of the Platform.
- 7096 4. Information on delivery procedures, involved sites and data exchange.
- 7097 5. Information about penetration testing of the Platform including the considered  
7098 attack paths and summary of test results.
- 7099 6. Information about penetration testing of the supporting functions in the platform
- 7100 7. Observations and recommendations for users.

7101 **D.2.4.1 Formal information**

7102 This section of ETR\_COMP shall provide formal information on the platform evaluation as:

- 7103 ➤ product identification,
- 7104 ➤ sponsor and developer identities,
- 7105 ➤ identities of the evaluation facility and the certification body,
- 7106 ➤ assurance level of the evaluation,
- 7107 ➤ formal evaluation and certification results like pass/fail,
- 7108 ➤ references to the ETR.

7109 **D.2.4.2 Platform design**

7110 This section of ETR\_COMP shall provide a high-level description of the IT product and its major  
7111 components based on the deliverables required by the assurance class ADV of the Common Criteria.  
7112 The intent of this section is to characterize the degree of architectural separation of the major  
7113 components and to show possible technical dependencies between the platform and an application  
7114 using the platform (e.g. dependencies between HW platform and SW application). This shall include an  
7115 outline of security mechanisms of the platform covered by the platform evaluation.

7116 **D.2.4.3 Evaluated configuration**

7117 This section of ETR\_COMP shall provide information about the evaluated configuration of the Platform  
7118 based on the developer's configuration list or relevant parts as needed or on a case by case basis. The  
7119 platform must unambiguously be identifiable and this identification shall be commensurate with the  
7120 evaluated configuration as stated in the platform certification report.



7121 If applicable, generation and installation parameter settings being security relevant for the Platform  
 7122 should be explained and their effect on the defence against attacks is outlined (e.g. key length, counters  
 7123 limits). This includes methods for the application developer and evaluator to verify the values of these  
 7124 settings, in order to verify that the expected evaluated configuration is used.

7125 This evidence may include TOE installation, generation and start-up procedures as outlined in  
 7126 AGD\_PRE to enforce that the platform is configured in a secure manner.

7127

#### 7128 **D.2.4.4 Delivery procedures, sites and data exchange**

7129 For supporting composite evaluation, evaluation evidence can be necessary for delivery of the  
 7130 platform, and acceptance procedures of the application and related data to be integrated during  
 7131 development and production. Therefore, evaluation evidence about AGD\_PRE and ALC\_DEL +  
 7132 AGD\_PRE might be relevant.

7133 The ETR\_COMP shall provide an overview of the sites involved in the development and production of  
 7134 the platform, including the role of each site and the date of latest site visit.

7135 For the composite evaluation, of an OS on an IC the description of phase 1 and 4 are needed and will be  
 7136 detailed in this document. The delivery of the IC dedicated software and guidance to the application  
 7137 developer should also be considered. In addition details on the fab-key protection mechanism should  
 7138 be identified.

7139 For an IC, the deliveries under consideration are:

7140 1. The delivery of the embedded application code to the microcontroller  
 7141 manufacturer, (in case of Flash products this may be replaced by the delivery of a  
 7142 key from the microcontroller manufacturer to the developer of the Security IC  
 7143 Embedded Software)

7144 2. The delivery of the microcontroller to the entity in charge of the next step (testing,  
 7145 embedding into micro-module, card manufacturing).

7146

7147 For an OS the deliveries under consideration are:

7148 1. The delivery of the embedded application code to the manufacturer (if the code will  
 7149 be embedded in ROM) or product integrator (if the code will be embedded in  
 7150 EEPROM or Flash).

7151 2. The delivery of the smart card/platform (IC with embedded OS) to the in charge of  
 7152 the next step (product integrator, personaliser, etc.)

7153 3. The delivery of security guidance

7154 4. The exchange of key-material for access to the smart card/platform (IC with  
 7155 embedded OS).

#### 7156 **D.2.4.5 Penetration Testing**

7157 This section of ETR\_COMP shall provide information about the independent vulnerability analysis  
 7158 performed by the Platform Evaluator with the attack scenarios having been considered, the  
 7159 penetration testing having been performed and the reference to the corresponding rating (quotation)  
 7160 of the attack potential.

7161 Information about penetration testing results should include:

- 7162 ➤ details necessary for understanding the attack scenarios/paths
- 7163 ➤ the assessments of penetration results as well as a summary showing that all  
 7164 attack methods were addressed during the vulnerability analysis.

7165 If a potential vulnerability has to be resolved by adhering to guidance this must be clear from the  
 7166 summary including a reference to a specific section in guidance or if possible a guidance element.

7167 The attack scenario descriptions should provide sufficient details to support the Composite Product  
 7168 Evaluator to reproduce attacks, which require additional countermeasures in the Composite TOE.

7169 In accordance with the requirements of IOS/IEC 18045, this information is available within the ETR.  
 7170 So it can be compiled for ETR\_COMP.

7171 This section shall also mention the rating of access to 'open samples' (i.e.  
 7172 public/restricted/sensitive/critical). The use of 'open samples' shall be considered in the assessment  
 7173 of the attack path. Please note that 'open samples' are evaluation tools, but do not represent a TOE.

#### 7174 **D.2.4.6 Observations and recommendations**

7175 The evaluated user guidance documentation shall contain all information required to use the TOE in a  
 7176 secure way as defined in the platform security target including recommendations on how to avoid  
 7177 residual vulnerabilities and unexpected behaviour. The recommendations and the user guidance  
 7178 documentation shall be consistent. The Platform Evaluator shall verify that the ETR for Composition  
 7179 only contains recommendations on the secure use that are also addressed as requirements in the user  
 7180 guidance. The user guidance requirements must be specific enabling the Application Developer to  
 7181 perform design compliance analysis

7182 However, in specific cases detailed information might be required in addition to the guidance  
 7183 documents such as:

- 7184 ➤ Observations on the evaluation results (e.g. specific TOE configuration for the  
 7185 evaluation),
- 7186 ➤ Recommendations/stipulations for the Composite Product Evaluator: specific  
 7187 information on use of the evaluation results (e.g. about specific testing  
 7188 necessary during a composition evaluation).

7189 Any such observation or recommendation/stipulation may come from the Platform Evaluator and the  
 7190 Platform Certification Body.

### 7191 **D.3 Evaluation/Certification reports and Platform certificate validity**

7192 Results of a composite evaluation shall be provided to the Composite Product Certification Body in  
 7193 form of an Evaluation Technical Report for the composite product. This Composite Product ETR shall

- 7194 contain, amongst others, the final overall verdict for the composite evaluation based on the partial  
7195 verdicts for each assurance component being in scope of the current composite evaluation.
- 7196 As the composite product certificate covers also the platform, the composite product certificate  
7197 validity is linked to the validity of the platform certificate.
- 7198 The Composite Product Certification Body needs an up-to-date certificate or an assessment from the  
7199 Platform Certification Body on the status of the platform certificate in question.
- 7200 As a general rule the Composite Product Certification Body will ask for a reassessment of the platform  
7201 if the date of the platform's ETR for Composition is more than one and a half year before the  
7202 submission of the report containing the full results of the composition penetration tests. This  
7203 reassessment consists of either a re-evaluation of the platform focussing on a renewal of the  
7204 vulnerability analysis (surveillance task) or alternatively, a confirmation statement of the Platform  
7205 Certification Body may be requested.
- 7206 Note that in the case the entire composite product is set up as a chain of composite products  
7207 constructed on top of each other (e.g. the platform itself is already a composite product) the maximum  
7208 validity period of 18 months is related to the eldest ETR for Composition used in this chain of  
7209 composite products. In addition, dependencies from a lower level ETR for Composition to a higher  
7210 level ETR for Composition need to be considered when reusing the results in the composite evaluation  
7211 on top.
- 7212 Note also that if the platform's ETR for Composition was issued less than a year and a half ago before  
7213 submission of the related composite evaluation tasks, but there was a major change in the state of the  
7214 art in performing relevant attacks on the platform (e.g. a major change in the "Application of Attack  
7215 Potential to Smart Cards" document [JIL AP] or a major change in attack methods or attack ratings)  
7216 then the Composite Product Certification Body has the right to require a reassessment focusing on the  
7217 new attack method.
- 7218 Validity and relevance of the platform certificate for the current composite product certification shall  
7219 be acknowledged by the Composite Product Certification Body and includes the determination of  
7220 equivalence of single assurance components (and, hence, of assurance levels) belonging to different  
7221 version of ISO/IEC 15408, if the platform certification was according to another version of ISO/IEC  
7222 15408 than the current composite certification is. Such equivalence shall be established /  
7223 acknowledged by the Composite Product Certification Body.
- 7224 The Composite Product Certification Body can issue a security certificate for the composite product, if
- 7225                   ➤ the verdicts for the Composite Product ETR is PASS and
- 7226                   ➤ validity and topicality of the platform certificate for the current composite  
7227 product is acknowledged by the Composite Product Certification Body.
- 7228 Note that, if the Composite Product Evaluator detects some failures resulting from Platform testing  
7229 (e.g. vulnerabilities due to improved attack methods or techniques), the results shall be communicated  
7230 to the Composite Product Certification Body. The Composite Product Certification Body shall then take  
7231 appropriate steps together with the Platform Certification Body, e.g. to invoke a re-assessment or re-  
7232 certification of the platform TOE.
- 7233 The Platform Certification Body shall verify that the recommendations in the ETR for composition of  
7234 the platform are consistent with the requirements provided in the platform user guidance before

7235 issuing the certification report. When inconsistencies are detected the Platform Certification Body has  
7236 the freedom to add missing information for the Application Developer in the certification report.

7237