

<b>COMMITTEE DRAFT</b>		Reference document: <b>SC 27 N18805</b>	
<b>ISO/IEC CD 15408-3, revision</b>			
Date: <b>2019-01-08</b>		Supersedes document: N18752	
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>2019-03-05</b> Please submit your comments via the online balloting application by the due date indicated.		
<b>ISO/IEC CD 15408-3, revision</b>			
Title: <b>IT Security techniques – Evaluation criteria for IT security — Part 3: Security assurance components</b>			
Project: 1.27.16.03 (ISO/IEC 15408-3, revision)			
<b>Explanatory Report</b>			
<b>Status</b>	<b>SC 27 Decision</b>	<b>Reference documents</b>	
		<b>Input</b>	<b>Output</b>
<i>For details regarding previous development stages refer to 2<sup>nd</sup> page of this explanatory report.</i>			
<b>ISO/IEC 15408-3</b> <b>1<sup>st</sup> WD</b>	54 <sup>th</sup> WG 3 meeting, April 2017, Recommendations 5,10, 11, 14 (N17041 = WG 3 N1413).	SoV (N17027).	Liaisons to: CCDB (WG 3 N1391); The Open Group (WG 3 N1394); ISO/TC 22/SC 32 (N17373); Text f. 1 <sup>st</sup> WD (WG 3 N1437).
<b>ISO/IEC 15408-3</b> <b>2<sup>nd</sup> WD</b>	55 <sup>th</sup> WG 3 meeting, Oct. 2017, Recommendations 8, 10 (N17666 = WG 3 N1494).	SoCom (WG 3 N1467); Draft DoC (WG 3 N1501).	Editor's report (WG 3 N1465); Liaisons to: CCDB (WG 3 N1455); ISO/TC 22/SC 32 (N18103); DoC (WG 3 N1462); Text f. 2 <sup>nd</sup> WD (WG 3 N1469).
<b>ISO/IEC 15408-3</b> <b>1<sup>st</sup> CD</b>	56 <sup>th</sup> WG 3 meeting, April 2018, Recommendations 8, 10 (N18471 = WG 3 N1557) / 30 <sup>th</sup> SC 27 Plenary, April 2018, Resolution 6 (N18710).	SoCom (WG 3 N1530); Late Com (WG 3 N1564).	Liaison to: CCDB (WG 3 N1521); DoC (WG 3 N1527); Text f. 1 <sup>st</sup> CD (N18752).
<b>ISO/IEC 15408-3</b> <b>2<sup>nd</sup> CD</b>	57 <sup>th</sup> WG 3 meeting / CRM, Sep / Oct 2018, Recommendations 11, 14 (N18820 = WG 3 N1610).	SoV (N18853).	Liaison to: CCDB (WG 3 N1619); DoC (N18802); Text f. 2 <sup>nd</sup> CD (N18805).
<b>2<sup>nd</sup> CD Consideration</b>			
In accordance with Recommendation 14 (see SC 27 N18820 = WG 3 N1610) of the 57 <sup>th</sup> SC 27/WG 3 meeting held in Gjøvik, Norway, 2018-09-30/10-04 the hereby attached document is being circulated for a 8-week 2 <sup>nd</sup> CD letter ballot closing by			
<b>2019-03-05</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 + 220			

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

ISO/IEC 15408-3:XXXX(E)

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

Secretariat: DIN

**IT security techniques — Evaluation criteria for IT security —  
Part 3: Security assurance components**

**CD stage**

**Warning for WDs and CDs**

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

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

© ISO 2017, Published in Switzerland

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

**International Organization for Standardization International Electrotechnical Commission**

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

47	7.7.4	APE_REQ.2 Derived security requirements.....	19
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).....	24
57	8.4.1	ACE_SPD.1 PP-Module Security problem definition.....	24
58	8.4.2	Application notes.....	24
59	8.5	PP-Module Security objectives (ACE_OBJ) .....	25
60	8.5.1	ACE_OBJ.1 PP-Module Security objectives .....	25
61	8.5.2	Application notes.....	25
62	8.6	PP-Module extended components definition (ACE_ECD).....	27
63	8.6.1	Objectives .....	27
64	8.6.2	ACE_ECD.1 PP-Module extended components definition.....	27
65	8.7	PP-Module security requirements (ACE_REQ) .....	28
66	8.7.1	Objectives .....	28
67	8.7.2	ACE_REQ.1 Stated security requirements.....	28
68	8.7.3	ACE_REQ.2 PP-Module security functional requirements .....	30
69	8.8	PP-Module consistency (ACE_MCO).....	31
70	8.8.1	Objectives .....	31
71	8.8.2	ACE_MCO.1 PP-Module consistency .....	31
72	8.9	PP-Configuration consistency (ACE_CCO) .....	33
73	8.9.1	Objectives.....	33
74	8.9.2	ACE_CCO.1 PP-Configuration consistency .....	33
75	9	Class ASE: Security Target evaluation.....	36
76	9.1	Introduction .....	36
77	9.2	ST introduction (ASE_INT).....	36
78	9.2.1	Objectives.....	36
79	9.2.2	ASE_INT.1 ST introduction.....	36
80	9.3	Conformance claims (ASE_CCL) .....	38
81	9.3.1	Objectives .....	38
82	9.3.2	ASE_CCL.1 Conformance claims .....	38
83	9.4	Security problem definition (ASE_SPD).....	39
84	9.4.1	Objectives.....	39
85	9.4.2	ASE_SPD.1 Security problem definition.....	39
86	9.5	Security objectives (ASE_OBJ) .....	40
87	9.5.1	Objectives .....	40
88	9.5.2	Component levelling.....	40
89	9.5.3	ASE_OBJ.1 Security objectives for the operational environment.....	40
90	9.5.4	ASE_OBJ.2 Security objectives .....	41
91	9.6	Extended components definition (ASE_ECD) .....	42
92	9.6.1	Objectives.....	42
93	9.6.2	ASE_ECD.1 Extended components definition.....	42
94	9.7	Security requirements (ASE_REQ).....	43
95	9.7.1	Objectives .....	43
96	9.7.2	Component levelling.....	43
97	9.7.3	ASE_REQ.1 Stated security requirements .....	43
98	9.7.4	ASE_REQ.2 Derived security requirements.....	45

99	<b>9.8</b>	<b>TOE summary specification (ASE_TSS)</b> .....	<b>46</b>	
100	<b>9.8.1</b>	<b>Objectives</b> .....	<b>46</b>	
101	<b>9.8.2</b>	<b>Component levelling</b> .....	<b>46</b>	
102	<b>9.8.3</b>	<b>ASE_TSS.1 TOE summary specification</b> .....	<b>46</b>	
103	<b>9.8.4</b>	<b>ASE_TSS.2 TOE summary specification with architectural design summary</b> .....	<b>47</b>	
104	<b>9.9</b>	<b>Consistency of composite product Security Target (ASE_COMP)</b> .....	<b>48</b>	
105	<b>9.9.1</b>	<b>Objectives</b> .....	<b>48</b>	
106	<b>9.9.2</b>	<b>ASE_COMP.1</b> .....	<b>Consistency of Security Target</b>	<b>48</b>
107	<b>10</b>	<b>Class ADV: Development</b> .....	<b>50</b>	
108	<b>10.1</b>	<b>Introduction</b> .....	<b>50</b>	
109	<b>10.2</b>	<b>Security Architecture (ADV_ARC)</b> .....	<b>55</b>	
110	<b>10.2.1</b>	<b>Objectives</b> .....	<b>55</b>	
111	<b>10.2.2</b>	<b>Component levelling</b> .....	<b>56</b>	
112	<b>10.2.3</b>	<b>Application notes</b> .....	<b>56</b>	
113	<b>10.2.4</b>	<b>ADV_ARC.1 Security architecture description</b> .....	<b>56</b>	
114	<b>10.3</b>	<b>Functional specification (ADV_FSP)</b> .....	<b>58</b>	
115	<b>10.3.1</b>	<b>Objectives</b> .....	<b>58</b>	
116	<b>10.3.2</b>	<b>Component levelling</b> .....	<b>58</b>	
117	<b>10.3.3</b>	<b>Application notes</b> .....	<b>58</b>	
118	<b>10.3.4</b>	<b>ADV_FSP.1 Basic functional specification</b> .....	<b>61</b>	
119	<b>10.3.5</b>	<b>ADV_FSP.2 Security-enforcing functional specification</b> .....	<b>61</b>	
120	<b>10.3.6</b>	<b>ADV_FSP.3 Functional specification with complete summary</b> .....	<b>62</b>	
121	<b>10.3.7</b>	<b>ADV_FSP.4 Complete functional specification</b> .....	<b>64</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>65</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>66</b>	
126	<b>10.4</b>	<b>Implementation representation (ADV_IMP)</b> .....	<b>67</b>	
127	<b>10.4.1</b>	<b>Objectives</b> .....	<b>67</b>	
128	<b>10.4.2</b>	<b>Component levelling</b> .....	<b>68</b>	
129	<b>10.4.3</b>	<b>Application notes</b> .....	<b>68</b>	
130	<b>10.4.4</b>	<b>ADV_IMP.1 Implementation representation of the TSF</b> .....	<b>69</b>	
131	<b>10.4.5</b>	<b>ADV_IMP.2 Complete mapping of the implementation representation of the TSF</b> .....	<b>69</b>	
132	<b>10.5</b>	<b>TSF internals (ADV_INT)</b> .....	<b>70</b>	
133	<b>10.5.1</b>	<b>Objectives</b> .....	<b>70</b>	
134	<b>10.5.2</b>	<b>Component levelling</b> .....	<b>70</b>	
135	<b>10.5.3</b>	<b>Application notes</b> .....	<b>71</b>	
136	<b>10.5.4</b>	<b>ADV_INT.1 Well-structured subset of TSF internals</b> .....	<b>71</b>	
137	<b>10.5.5</b>	<b>ADV_INT.2 Well-structured internals</b> .....	<b>72</b>	
138	<b>10.5.6</b>	<b>ADV_INT.3 Minimally complex internals</b> .....	<b>73</b>	
139	<b>10.6</b>	<b>Security policy modelling (ADV_SPM)</b> .....	<b>74</b>	
140	<b>10.6.1</b>	<b>Objectives</b> .....	<b>74</b>	
141	<b>10.6.2</b>	<b>Component levelling</b> .....	<b>74</b>	
142	<b>10.6.3</b>	<b>Application notes</b> .....	<b>74</b>	
143	<b>10.6.4</b>	<b>ADV_SPM.1 Formal TOE security policy model</b> .....	<b>75</b>	
144	<b>10.7</b>	<b>TOE design (ADV_TDS)</b> .....	<b>77</b>	
145	<b>10.7.1</b>	<b>Objectives</b> .....	<b>77</b>	
146	<b>10.7.2</b>	<b>Component levelling</b> .....	<b>77</b>	
147	<b>10.7.3</b>	<b>Application notes</b> .....	<b>78</b>	
148	<b>10.7.4</b>	<b>ADV_TDS.1 Basic design</b> .....	<b>79</b>	
149	<b>10.7.5</b>	<b>ADV_TDS.2 Architectural design</b> .....	<b>80</b>	
150	<b>10.7.6</b>	<b>ADV_TDS.3 Basic modular design</b> .....	<b>81</b>	

151	<b>10.7.7 ADV_TDS.4 Semiformal modular design .....</b>	<b>83</b>
152	<b>10.7.8 ADV_TDS.5 Complete semiformal modular design.....</b>	<b>84</b>
153	<b>10.7.9 ADV_TDS.6 Complete semiformal modular design with formal high-level design</b>	
154	<b>presentation .....</b>	<b>85</b>
155	<b>10.8 Composite design compliance (ADV_COMP).....</b>	<b>87</b>
156	<b>10.8.1 Objectives .....</b>	<b>87</b>
157	<b>10.8.2 Component levelling.....</b>	<b>87</b>
158	<b>10.8.3 Application notes.....</b>	<b>87</b>
159	<b>10.8.4 ADV_COMP.1 Design compliance with the platform certification report, guidance and ETR_COMP 8</b>	
160	<b>11 Class AGD: Guidance documents .....</b>	<b>88</b>
161	<b>11.1 Introduction .....</b>	<b>88</b>
162	<b>11.2 Operational user guidance (AGD_OPE).....</b>	<b>89</b>
163	<b>11.2.1 Objectives .....</b>	<b>89</b>
164	<b>11.2.2 Component levelling.....</b>	<b>89</b>
165	<b>11.2.3 Application notes.....</b>	<b>89</b>
166	<b>11.2.4 AGD_OPE.1 Operational user guidance.....</b>	<b>90</b>
167	<b>11.3 Preparative procedures (AGD_PRE) .....</b>	<b>91</b>
168	<b>11.3.1 Objectives .....</b>	<b>91</b>
169	<b>11.3.2 Component levelling.....</b>	<b>91</b>
170	<b>11.3.3 Application notes.....</b>	<b>91</b>
171	<b>11.3.4 AGD_PRE.1 Preparative procedures .....</b>	<b>91</b>
172	<b>12 Class ALC: Life-cycle support .....</b>	<b>92</b>
173	<b>12.1 Introduction .....</b>	<b>92</b>
174	<b>12.2 CM capabilities (ALC_CMC) .....</b>	<b>93</b>
175	<b>12.2.1 Objectives .....</b>	<b>93</b>
176	<b>12.2.2 Component levelling.....</b>	<b>94</b>
177	<b>12.2.3 Application notes.....</b>	<b>94</b>
178	<b>12.2.4 ALC_CMC.1 Labelling of the TOE .....</b>	<b>95</b>
179	<b>12.2.5 ALC_CMC.2 Use of the CM system.....</b>	<b>95</b>
180	<b>12.2.6 ALC_CMC.3 Authorisation controls.....</b>	<b>96</b>
181	<b>12.2.7 ALC_CMC.4 Production support, acceptance procedures and automation.....</b>	<b>98</b>
182	<b>12.2.8 ALC_CMC.5 Advanced support.....</b>	<b>100</b>
183	<b>12.3 CM scope (ALC_CMS) .....</b>	<b>103</b>
184	<b>12.3.1 Objectives .....</b>	<b>103</b>
185	<b>12.3.2 Component levelling.....</b>	<b>103</b>
186	<b>12.3.3 Application notes.....</b>	<b>103</b>
187	<b>12.3.4 ALC_CMS.1 TOE CM coverage .....</b>	<b>103</b>
188	<b>12.3.5 ALC_CMS.2 Parts of the TOE CM coverage .....</b>	<b>104</b>
189	<b>12.3.6 ALC_CMS.3 Implementation representation CM coverage.....</b>	<b>105</b>
190	<b>12.3.7 ALC_CMS.4 Problem tracking CM coverage .....</b>	<b>106</b>
191	<b>12.3.8 ALC_CMS.5 Development tools CM coverage.....</b>	<b>107</b>
192	<b>12.4 Delivery (ALC_DEL) .....</b>	<b>108</b>
193	<b>12.4.1 Objectives .....</b>	<b>108</b>
194	<b>12.4.2 Component levelling.....</b>	<b>108</b>
195	<b>12.4.3 Application notes.....</b>	<b>108</b>
196	<b>12.4.4 ALC_DEL.1 Delivery procedures .....</b>	<b>109</b>
197	<b>12.5 Developer environment security (ALC_DVS).....</b>	<b>109</b>
198	<b>12.5.1 Objectives .....</b>	<b>109</b>
199	<b>12.5.2 Component levelling.....</b>	<b>109</b>
200	<b>12.5.3 Application notes.....</b>	<b>109</b>
201	<b>12.5.4 ALC_DVS.1 Identification of security controls .....</b>	<b>110</b>
202	<b>12.5.5 ALC_DVS.2 Sufficiency of security controls.....</b>	<b>110</b>

203	<b>12.6</b>	<b>Flaw remediation (ALC_FLR)</b> .....	<b>111</b>	
204	<b>12.6.1</b>	<b>Objectives</b> .....	<b>111</b>	
205	<b>12.6.2</b>	<b>Component levelling</b> .....	<b>111</b>	
206	<b>12.6.3</b>	<b>Application notes</b> .....	<b>111</b>	
207	<b>12.6.4</b>	<b>ALC_FLR.1 Basic flaw remediation</b> .....	<b>112</b>	
208	<b>12.6.5</b>	<b>ALC_FLR.2 Flaw reporting procedures</b> .....	<b>112</b>	
209	<b>12.6.6</b>	<b>ALC_FLR.3 Systematic flaw remediation</b> .....	<b>114</b>	
210	<b>12.7</b>	<b>Development Life-cycle definition (ALC_LCD)</b> .....	<b>116</b>	
211	<b>12.7.1</b>	<b>Objectives</b> .....	<b>116</b>	
212	<b>12.7.2</b>	<b>Component levelling</b> .....	<b>116</b>	
213	<b>12.7.3</b>	<b>Application notes</b> .....	<b>116</b>	
214	<b>12.7.4</b>	<b>ALC_LCD.1 Developer defined life-cycle processes</b> .....	<b>116</b>	
215	<b>12.7.5</b>	<b>ALC_LCD.2 Measurable life-cycle model</b> .....	<b>117</b>	
216	<b>12.8</b>	<b>Practices for trustable development (ALC_TDA)</b> .....	<b>118</b>	
217	<b>12.8.1</b>	<b>Objectives</b> .....	<b>118</b>	
218	<b>12.8.2</b>	<b>Component levelling</b> .....	<b>119</b>	
219	<b>12.8.3</b>	<b>Application notes</b> .....	<b>119</b>	
220	<b>12.8.4</b>	<b>ALC_TDA.1 Uniquely identifying implementation representation</b> .....	<b>119</b>	
221	<b>12.8.5</b>	<b>ALC_TDA.2 Matching CMS scope of implementation representation</b> .....	<b>122</b>	
222	<b>12.8.6</b>	<b>ALC_TDA.3 Regenerate TOE with well-defined development tools</b> .....	<b>124</b>	
223	<b>12.9</b>	<b>Tools and techniques (ALC_TAT)</b> .....	<b>127</b>	
224	<b>12.9.1</b>	<b>Objectives</b> .....	<b>127</b>	
225	<b>12.9.2</b>	<b>Component levelling</b> .....	<b>127</b>	
226	<b>12.9.3</b>	<b>Application notes</b> .....	<b>127</b>	
227	<b>12.9.4</b>	<b>ALC_TAT.1 Well-defined development tools</b> .....	<b>127</b>	
228	<b>12.9.5</b>	<b>ALC_TAT.2 Compliance with implementation standards</b> .....	<b>128</b>	
229	<b>12.9.6</b>	<b>ALC_TAT.3 Compliance with implementation standards - all parts</b> .....	<b>129</b>	
230	<b>12.10</b>	<b>Integration of composition parts and consistency check of delivery procedures</b>		
231		<b>(ALC_COMP)</b> .....	<b>130</b>	
232	<b>12.10.1</b>	.....	<b>Objectives</b>	<b>130</b>
233	<b>12.10.2</b>	.....	<b>Component levelling</b>	<b>130</b>
234	<b>12.10.3</b>	.....	<b>ALC_COMP.1</b>	<b>Inte</b>
235	<b>13</b>	<b>Class ATE: Tests</b> .....	<b>131</b>	
236	<b>13.1</b>	<b>Introduction</b> .....	<b>131</b>	
237	<b>13.2</b>	<b>Coverage (ATE_COV)</b> .....	<b>132</b>	
238	<b>13.2.1</b>	<b>Objectives</b> .....	<b>132</b>	
239	<b>13.2.2</b>	<b>Component levelling</b> .....	<b>132</b>	
240	<b>13.2.3</b>	<b>Application notes</b> .....	<b>132</b>	
241	<b>13.2.4</b>	<b>ATE_COV.1 Evidence of coverage</b> .....	<b>132</b>	
242	<b>13.2.5</b>	<b>ATE_COV.2 Analysis of coverage</b> .....	<b>133</b>	
243	<b>13.2.6</b>	<b>ATE_COV.3 Rigorous analysis of coverage</b> .....	<b>134</b>	
244	<b>13.3</b>	<b>Depth (ATE_DPT)</b> .....	<b>134</b>	
245	<b>13.3.1</b>	<b>Objectives</b> .....	<b>134</b>	
246	<b>13.3.2</b>	<b>Component levelling</b> .....	<b>135</b>	
247	<b>13.3.3</b>	<b>Application notes</b> .....	<b>135</b>	
248	<b>13.3.4</b>	<b>ATE_DPT.1 Testing: basic design</b> .....	<b>135</b>	
249	<b>13.3.5</b>	<b>ATE_DPT.2 Testing: security enforcing modules</b> .....	<b>136</b>	
250	<b>13.3.6</b>	<b>ATE_DPT.3 Testing: modular design</b> .....	<b>137</b>	
251	<b>13.3.7</b>	<b>ATE_DPT.4 Testing: implementation representation</b> .....	<b>137</b>	
252	<b>13.4</b>	<b>Functional tests (ATE_FUN)</b> .....	<b>138</b>	
253	<b>13.4.1</b>	<b>Objectives</b> .....	<b>138</b>	
254	<b>13.4.2</b>	<b>Component levelling</b> .....	<b>139</b>	

255	<b>13.4.3 Application notes</b> .....	<b>139</b>
256	<b>13.4.4 ATE_FUN.1 Functional testing</b> .....	<b>139</b>
257	<b>13.4.5 ATE_FUN.2 Ordered functional testing</b> .....	<b>140</b>
258	<b>13.5 Independent testing (ATE_IND)</b> .....	<b>141</b>
259	<b>13.5.1 Objectives</b> .....	<b>141</b>
260	<b>13.5.2 Component levelling</b> .....	<b>141</b>
261	<b>13.5.3 Application notes</b> .....	<b>141</b>
262	<b>13.5.4 ATE_IND.1 Independent testing - conformance</b> .....	<b>142</b>
263	<b>13.5.5 ATE_IND.2 Independent testing - sample</b> .....	<b>143</b>
264	<b>13.5.6 ATE_IND.3 Independent testing - complete</b> .....	<b>144</b>
265	<b>13.6 Composite functional testing (ATE_COMP)</b> .....	<b>145</b>
266	<b>13.6.1 Objectives</b> .....	<b>145</b>
267	<b>13.6.2 Application notes</b> .....	<b>145</b>
268	<b>13.6.3 ATE_COMP.1 Composite product functional testing</b> .....	<b>146</b>
269	<b>14 Class AVA: Vulnerability assessment</b> .....	<b>146</b>
270	<b>14.1 Introduction</b> .....	<b>146</b>
271	<b>14.2 Application notes</b> .....	<b>147</b>
272	<b>14.3 Vulnerability analysis (AVA_VAN)</b> .....	<b>147</b>
273	<b>14.3.1 Objectives</b> .....	<b>147</b>
274	<b>14.3.2 Component levelling</b> .....	<b>148</b>
275	<b>14.3.3 AVA_VAN.1 Vulnerability survey</b> .....	<b>148</b>
276	<b>14.3.4 AVA_VAN.2 Vulnerability analysis</b> .....	<b>149</b>
277	<b>14.3.5 AVA_VAN.3 Focused vulnerability analysis</b> .....	<b>150</b>
278	<b>14.3.6 AVA_VAN.4 Methodical vulnerability analysis</b> .....	<b>151</b>
279	<b>14.3.7 AVA_VAN.5 Advanced methodical vulnerability analysis</b> .....	<b>152</b>
280	<b>14.4 Composite vulnerability assessment (AVA_COMP)</b> .....	<b>154</b>
281	<b>14.4.1 Objectives</b> .....	<b>154</b>
282	<b>14.4.2 AVA_COMP.1 Composite product vulnerability assessment</b> .....	<b>154</b>
283	<b>15 Class ACO: Composition</b> .....	<b>155</b>
284	<b>15.1 Introduction</b> .....	<b>155</b>
285	<b>15.2 Composition rationale (ACO_COR)</b> .....	<b>158</b>
286	<b>15.2.1 Objectives</b> .....	<b>158</b>
287	<b>15.2.2 Component levelling</b> .....	<b>158</b>
288	<b>15.2.3 ACO_COR.1 Composition rationale</b> .....	<b>158</b>
289	<b>15.3 Development evidence (ACO_DEV)</b> .....	<b>158</b>
290	<b>15.3.1 Objectives</b> .....	<b>158</b>
291	<b>15.3.2 Component levelling</b> .....	<b>158</b>
292	<b>15.3.3 Application notes</b> .....	<b>159</b>
293	<b>15.3.4 ACO_DEV.1 Functional Description</b> .....	<b>159</b>
294	<b>15.3.5 ACO_DEV.2 Basic evidence of design</b> .....	<b>160</b>
295	<b>15.3.6 ACO_DEV.3 Detailed evidence of design</b> .....	<b>161</b>
296	<b>15.4 Reliance of dependent component (ACO_REL)</b> .....	<b>162</b>
297	<b>15.4.1 Objectives</b> .....	<b>162</b>
298	<b>15.4.2 Component levelling</b> .....	<b>162</b>
299	<b>15.4.3 Application notes</b> .....	<b>162</b>
300	<b>15.4.4 ACO_REL.1 Basic reliance information</b> .....	<b>163</b>
301	<b>15.4.5 ACO_REL.2 Reliance information</b> .....	<b>163</b>
302	<b>15.5 Composed TOE testing (ACO_CTT)</b> .....	<b>164</b>
303	<b>15.5.1 Objectives</b> .....	<b>164</b>
304	<b>15.5.2 Component levelling</b> .....	<b>164</b>
305	<b>15.5.3 Application notes</b> .....	<b>164</b>
306	<b>15.5.4 ACO_CTT.1 Interface testing</b> .....	<b>165</b>

307	<b>15.5.5 ACO_CTT.2 Rigorous interface testing.....</b>	<b>166</b>
308	<b>15.6 Composition vulnerability analysis (ACO_VUL).....</b>	<b>167</b>
309	<b>15.6.1 Objectives .....</b>	<b>167</b>
310	<b>15.6.2 Component levelling.....</b>	<b>167</b>
311	<b>15.6.3 Application notes.....</b>	<b>167</b>
312	<b>15.6.4 ACO_VUL.1 Composition vulnerability review .....</b>	<b>168</b>
313	<b>15.6.5 ACO_VUL.2 Composition vulnerability analysis.....</b>	<b>169</b>
314	<b>15.6.6 ACO_VUL.3 Enhanced-Basic Composition vulnerability analysis.....</b>	<b>169</b>
315	<b>Annex A (informative) Development (ADV) .....</b>	<b>171</b>
316	<b>A.1 ADV_ARC: Supplementary material on security architectures .....</b>	<b>171</b>
317	<b>A.1.1 Security architecture properties .....</b>	<b>171</b>
318	<b>A.1.2 Security architecture descriptions.....</b>	<b>172</b>
319	<b>A.2 ADV_FSP: Supplementary material on functional specification .....</b>	<b>174</b>
320	<b>A.2.1 Non-TSF part of the TOE.....</b>	<b>175</b>
321	<b>A.2.2 Determining the TSFI .....</b>	<b>176</b>
322	<b>A.2.3 Example: A complex DBMS.....</b>	<b>179</b>
323	<b>A.2.4 Example Functional Specification.....</b>	<b>180</b>
324	<b>A.3 ADV_INT: Supplementary material on TSF internals.....</b>	<b>182</b>
325	<b>A.3.1 Structure of procedural software .....</b>	<b>182</b>
326	<b>A.3.2 Complexity of procedural software.....</b>	<b>184</b>
327	<b>A.4 ADV_TDS: Subsystems and Modules .....</b>	<b>185</b>
328	<b>A.4.1 Subsystems.....</b>	<b>185</b>
329	<b>A.4.2 Modules .....</b>	<b>186</b>
330	<b>A.4.3 Levelling Approach .....</b>	<b>188</b>
331	<b>A.4.4 Security relevance .....</b>	<b>190</b>
332	<b>A.5 Supplementary material on formal methods .....</b>	<b>191</b>
333	<b>Annex B (informative) Composition (ACO) .....</b>	<b>193</b>
334	<b>B.1 Necessity for composed TOE evaluations .....</b>	<b>193</b>
335	<b>B.2 Performing Security Target evaluation for a composed TOE.....</b>	<b>194</b>
336	<b>B.3 Interactions between composed IT entities.....</b>	<b>195</b>
337	<b>Annex C (informative) Cross reference of assurance component dependencies.....</b>	<b>201</b>
338		

339 **Foreword**

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

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

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

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

356 ISO/IEC 15408-3 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*,  
 357 Subcommittee SC 27, *IT Security techniques*.

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

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

362 — *Part 1: Introduction and general model*

363 — *Part 2: Security functional components*

364 — *Part 3: Security assurance components*

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

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

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

368 This corrected version of ISO/IEC 15408-3:XXXX incorporates miscellaneous editorial corrections  
 369 mainly related to EAL4 and EAL6 assurance components, ADV\_FSP, ADV\_TDS, ATE\_DPT.2, ATE\_IND,  
 370 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.

379 380	Australia/New Zealand:	The Defence Signals Directorate and the Government Communications 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;
386 387	Spain: Nacional;	Ministerio de Administraciones Públicas and Centro Criptológico
388	United Kingdom:	Communications-Electronic Security Group;
389 390	United States: and	The National Security Agency and the National Institute of Standards Technology.

## 391 Introduction

392 Security assurance components, as defined in this document, are the basis for the security assurance  
393 requirements expressed in a Protection Profile (PP) or a Security Target (ST).

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

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

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

410 **Information technology Security techniques — Evaluation**  
411 **criteria for IT security —**

412 **Part 3:**  
413 **Security assurance components**

414 **1 Scope**

415 This document defines the assurance requirements of ISO/IEC 15408. It includes the individual  
416 assurance components from which the evaluation assurance levels and other packages contained in  
417 part 5 are composed, and the criteria for evaluation of Protection Profiles (PPs) and Security Targets  
418 (STs).

419 **2 Normative references**

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

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

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

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

429

430 **3 Terms and definitions, symbols and abbreviated terms**

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

433 **4 Overview**

434 **4.1 Organisation of this document**

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

436 Clause 6 describes the presentation structure of the assurance classes, families, components,  
437 evaluation assurance levels along with their relationships, and the structure of the composed  
438 assurance packages. It also characterises the assurance classes and families found in Clauses 7 through  
439 15.

440 Clauses 7 through 15 provide the detailed definitions of this document 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 Annex A 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 document 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) delivery, installation and configuration – that is, an IT product has vulnerabilities introduced  
503 during the 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 f) maintenance – that is, an IT product is maintained in such a way that new vulnerabilities are  
507 introduced.

### 508 **5.3.4 ISO/IEC 15408 assurance**

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

### 513 **5.3.5 Assurance through evaluation**

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

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

#### 527 **5.4 ISO/IEC 15408 evaluation assurance scale**

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

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

### 535 **6 Security assurance components**

#### 536 **6.1 Security assurance classes, families and components structure**

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

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

##### 544 **6.1.1 Assurance class structure**

545 Figure 1 illustrates the assurance class structure.

546 **6.1.1.1 Class name**

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

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

552 **6.1.1.2 Class introduction**

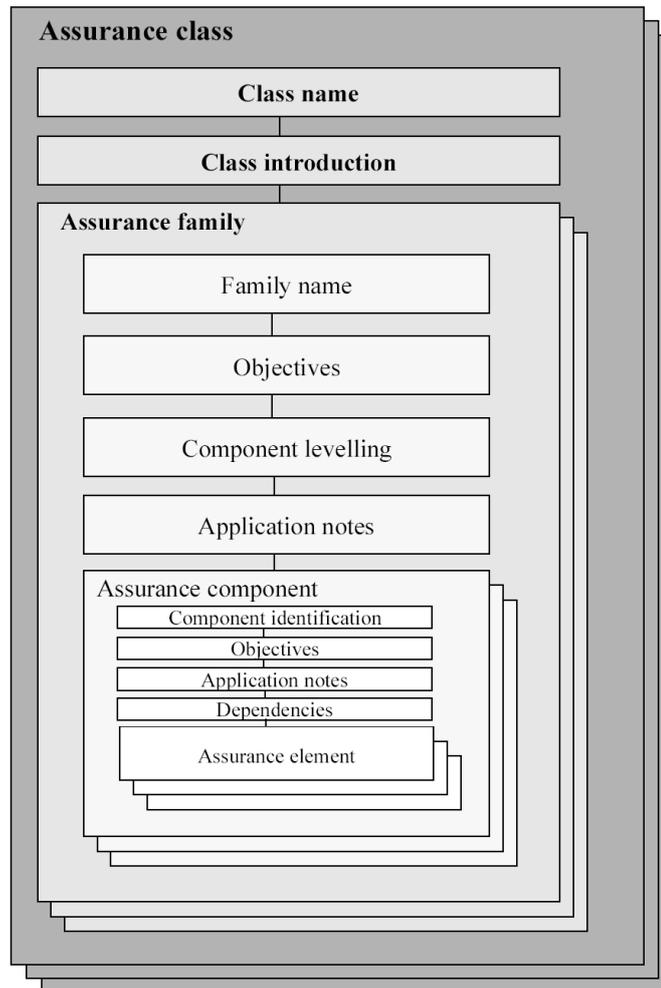
553 Each assurance class has an introductory subclause that describes the composition of the class and  
554 contains supportive text covering the intent of the class.

555 **6.1.1.3 Assurance families**

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

558 Figure 1 illustrates the assurance family structure.

## Common criteria assurance requirements



559

560

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

561

### 6.1.2 Assurance family structure

562

#### 6.1.2.1 Family name

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

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

569

#### 6.1.2.2 Objectives

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

571 This subclause describes the objectives, particularly those related to ISO/IEC 15408 assurance  
 572 paradigm, that the family is intended to address. The description for the assurance family is kept at a

573 general level. Any specific details required for objectives are incorporated in the particular assurance  
574 component.

### 575 6.1.2.3 Component levelling

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

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

### 582 6.1.2.4 Application notes

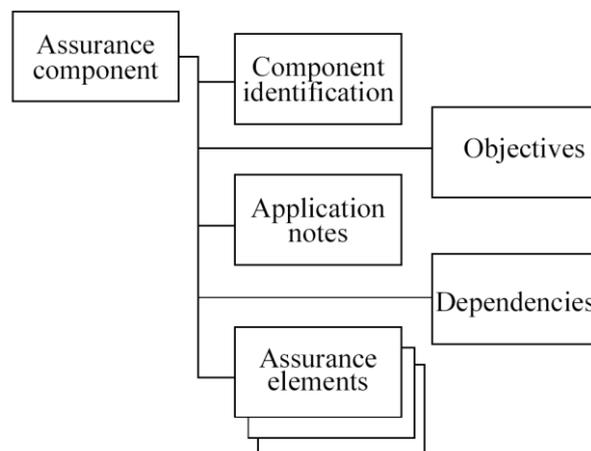
583 The application notes subclause of the assurance family, if present, contains additional information for  
584 the assurance family. This information should be of particular interest to users of the assurance family  
585 (e.g. PP and ST authors, designers of TOEs, evaluators). The presentation is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

### 587 6.1.2.5 Assurance components

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

## 590 6.1.3 Assurance component structure

591 Figure 2 illustrates the assurance component structure.



592

593 **Figure 2 — Assurance component structure**

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

### 597 6.1.3.1 Component identification

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

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

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

#### 607 **6.1.3.2 Objectives**

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

#### 611 **6.1.3.3 Application notes**

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

#### 614 **6.1.3.4 Dependencies**

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

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

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

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

#### 626 **6.1.3.5 Assurance elements**

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

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

- 631 a) Developer action elements: the activities that shall be performed by the developer. This set of  
 632 actions is further qualified by evidential material referenced in the following set of elements.  
 633 Requirements for developer actions are identified by appending the letter “D” to the element  
 634 number.
- 635 b) Content and presentation of evidence elements: the evidence required, what the evidence shall  
 636 demonstrate, and what information the evidence shall convey. Requirements for content and  
 637 presentation of evidence are identified by appending the letter “C” to the element number.

638 c) Evaluator action elements: the activities that shall be performed by the evaluator. This set of  
 639 actions explicitly includes confirmation that the requirements prescribed in the content and  
 640 presentation of evidence elements have been met. It also includes explicit actions and analysis that  
 641 shall be performed in addition to that already performed by the developer. Implicit evaluator  
 642 actions are also to be performed as a result of developer action elements which are not covered by  
 643 content and presentation of evidence requirements. Requirements for evaluator actions are  
 644 identified by appending the letter “E” to the element number.

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

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

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

#### 657 **6.1.4 Assurance elements**

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

#### 661 **6.1.5 Component taxonomy**

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



665

666

**Figure 3 — Sample class decomposition diagram**

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

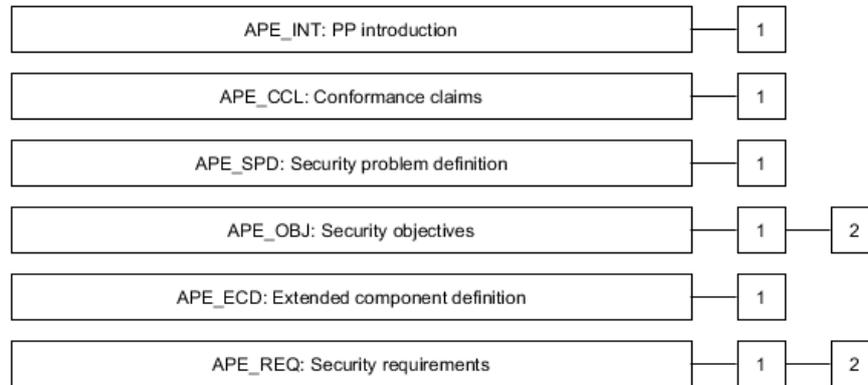
## 672 **7 Class APE: Protection Profile evaluation**

### 673 **7.1 Introduction**

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

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

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



681

682 **Figure 4 — APE: Protection Profile evaluation class decomposition**

## 683 7.2 PP introduction (APE\_INT)

### 684 7.2.1 Objectives

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

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

### 688 7.2.2 APE\_INT.1 PP introduction

689 Dependencies: No dependencies.

#### 690 7.2.2.1 Developer action elements

##### 691 7.2.2.1.1 APE\_INT.1.1D

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

#### 693 7.2.2.2 Content and presentation elements

##### 694 7.2.2.2.1 APE\_INT.1.1C

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

##### 696 7.2.2.2.2 APE\_INT.1.2C

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

##### 698 7.2.2.2.3 APE\_INT.1.3C

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

700 **7.2.2.2.4 APE\_INT.1.4C**

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

702 **7.2.2.2.5 APE\_INT.1.5C**

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

705 **7.2.2.3 Evaluator action elements**

706 **7.2.2.3.1 APE\_INT.1.1E**

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

709 **7.3 Conformance claims (APE\_CCL)**

710 **7.3.1 Objectives**

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

713 **7.3.2 APE\_CCL.1 Conformance claims**

714 Dependencies: APE\_INT.1 PP introduction

715 APE\_ECD.1 Extended components definition

716 APE\_REQ.1 Stated security requirements

717 **7.3.2.1 Developer action elements**

718 **7.3.2.1.1 APE\_CCL.1.1D**

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

720 **7.3.2.1.2 APE\_CCL.1.2D**

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

722 **7.3.2.1.3 APE\_CCL.1.3D**

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

724 **7.3.2.2 Content and presentation elements**

725 **7.3.2.2.1 APE\_CCL.1.1C**

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

728 **7.3.2.2.2 APE\_CCL.1.2C**

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

731 **7.3.2.2.3 APE\_CCL.1.3C**

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

734 **7.3.2.2.4 APE\_CCL.1.4C**

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

737 **7.3.2.2.5 APE\_CCL.1.5C**

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

740 **7.3.2.2.6 APE\_CCL.1.6C**

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

743 **7.3.2.2.7 APE\_CCL.1.7C**

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

746 **7.3.2.2.8 APE\_CCL.1.8C**

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

750 **7.3.2.2.9 APE\_CCL.1.9C**

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

754 **7.3.2.2.10 APE\_CCL.1.10C**

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

758 **7.3.2.2.11 APE\_CCL.1.11C**

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

761 **7.3.2.2.12 APE\_CCL.1.12C**

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

764 **7.3.2.2.13 APE\_CCL.1.13C**

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

767 **7.3.2.2.14 APE\_CCL.1.14C**

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

- 770 • **any Evaluation Methods and Evaluation Activities that are specified for the PP under**  
 771 **evaluation**
- 772 • **any Evaluation Methods and Evaluation Activities specified in conformance statements**  
 773 **of PPs to which conformance is being claimed by the PP under evaluation**
- 774 • **any Evaluation Methods and Evaluation Activities specified in Application Notes of**  
 775 **packages to which conformance is being claimed by the PP under evaluation.**

776 **7.3.2.3 Evaluator action elements**777 **7.3.2.3.1 APE\_CCL.1.1E**

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

780 **7.4 Security problem definition (APE\_SPD)**781 **7.4.1 Objectives**

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

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

786 **7.4.2 APE\_SPD.1 Security problem definition**

787 Dependencies: No dependencies.

788 **7.4.2.1 Developer action elements**789 **7.4.2.1.1 APE\_SPD.1.1D**

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

791 **7.4.2.2 Content and presentation elements**792 **7.4.2.2.1 APE\_SPD.1.1C**

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

794 **7.4.2.2.2 APE\_SPD.1.2C**

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

796 **7.4.2.2.3 APE\_SPD.1.3C**

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

798 **7.4.2.2.4 APE\_SPD.1.4C**

799 **The security problem definition shall describe the assumptions about the operational**  
800 **environment of the TOE.**

801 **7.4.2.3 Evaluator action elements**

802 **7.4.2.3.1 APE\_SPD.1.1E**

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

805 **7.5 Security objectives (APE\_OBJ)**

806 **7.5.1 Objectives**

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

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

812 **7.5.2 Component levelling**

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

815 **7.5.3 APE\_OBJ.1 Security objectives for the operational environment**

816 Dependencies: No dependencies.

817 **7.5.3.1 Developer action elements**

818 **7.5.3.1.1 APE\_OBJ.1.1D**

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

820 **7.5.3.2 Content and presentation elements**

821 **7.5.3.2.1 APE\_OBJ.1.1C**

822 **The statement of security objectives shall describe the security objectives for the operational**  
823 **environment.**

824

825 **7.5.3.3 Evaluator action elements**

826 **7.5.3.3.1 APE\_OBJ.1.1E**

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

829 **7.5.4 APE\_OBJ.2 Security objectives**

830 Dependencies: APE\_SPD.1 Security problem definition

831 **7.5.4.1 Developer action elements**

832 **7.5.4.1.1 APE\_OBJ.2.1D**

833 The developer shall provide a statement of security objectives.

834 **7.5.4.1.2 APE\_OBJ.2.2D**

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

836 **7.5.4.2 Content and presentation elements**

837 **7.5.4.2.1 APE\_OBJ.2.1C**

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

840 **7.5.4.2.2 APE\_OBJ.2.2C**

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

843 **7.5.4.2.3 APE\_OBJ.2.3C**

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

847 **7.5.4.2.4 APE\_OBJ.2.4C**

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

850 **7.5.4.2.5 APE\_OBJ.2.5C**

851 **The security objectives rationale shall demonstrate that the security objectives enforce all**  
852 **OSPs.**

853 **7.5.4.2.6 APE\_OBJ.2.6C**

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

856 **7.5.4.3 Evaluator action elements**

857 **7.5.4.3.1 APE\_OBJ.2.1E**

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

860 **7.6 Extended components definition (APE\_ECD)**

861 **7.6.1 Objectives**

862 Extended security requirements are requirements that are not based on components from ISO/IEC  
863 15408-2 or this document, but are based on extended components: components defined by the PP  
864 author.

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

868 **7.6.2 APE\_ECD.1 Extended components definition**

869 Dependencies: No dependencies.

870 **7.6.2.1 Developer action elements**

871 **7.6.2.1.1 APE\_ECD.1.1D**

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

873 **7.6.2.1.2 APE\_ECD.1.2D**

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

875 **7.6.2.2 Content and presentation elements**

876 **7.6.2.2.1 APE\_ECD.1.1C**

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

878 **7.6.2.2.2 APE\_ECD.1.2C**

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

881 **7.6.2.2.3 APE\_ECD.1.3C**

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

884 **7.6.2.2.4 APE\_ECD.1.4C**

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

887 **7.6.2.2.5 APE\_ECD.1.5C**

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

890 **7.6.2.3 Evaluator action elements**891 **7.6.2.3.1 APE\_ECD.1.1E**

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

894 **7.6.2.3.2 APE\_ECD.1.2E**

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

897 **7.7 Security requirements (APE\_REQ)**898 **7.7.1 Objectives**

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

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

904 **7.7.2 Component levelling**

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

907 **7.7.3 APE\_REQ.1 Stated security requirements**

908 Dependencies: APE\_ECD.1 Extended components definition

909 **7.7.3.1 Developer action elements**910 **7.7.3.1.1 APE\_REQ.1.1D**

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

912 **7.7.3.1.2 APE\_REQ.1.2D**

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

914 **7.7.3.2 Content and presentation elements**915 **7.7.3.2.1 APE\_REQ.1.1C**

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

917 **7.7.3.2.2 APE\_REQ.1.2C**

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

920 **7.7.3.2.3 APE\_REQ.1.3C**

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

925 **7.7.3.2.4 APE\_REQ.1.4C**

926 **The statement of security requirements shall identify all operations on the security**  
927 **requirements.**

928 **7.7.3.2.5 APE\_REQ.1.5C**

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

930 **7.7.3.2.6 APE\_REQ.1.6C**

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

933 **7.7.3.2.7 APE\_REQ.1.7C**

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

936 **7.7.3.2.8 APE\_REQ.1.8C**

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

940 **7.7.3.2.9 APE\_REQ.1.9C**

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

943 **7.7.3.2.10 APE\_REQ.1.10C**

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

946 **7.7.3.2.11 APE\_REQ.1.11C**

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

949 **7.7.3.2.12 APE\_REQ.1.12C**

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

951 **7.7.3.3 Evaluator action elements**

952 **7.7.3.3.1 APE\_REQ.1.1E**

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

955 **7.7.4 APE\_REQ.2 Derived security requirements**

956 Dependencies: APE\_OBJ.2 Security objectives

957 APE\_ECD.1 Extended components definition

958 **7.7.4.1 Developer action elements**

959 **7.7.4.1.1 APE\_REQ.2.1D**

960 The developer shall provide a statement of security requirements.

961 **7.7.4.1.2 APE\_REQ.2.2D**

962 The developer shall provide a security requirements rationale.

963 **7.7.4.2 Content and presentation elements**

964 **7.7.4.2.1 APE\_REQ.2.1C**

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

966 **7.7.4.2.2 APE\_REQ.2.2C**

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

969 **7.7.4.2.3 APE\_REQ.2.3C**

970 **Editor's Note:**

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

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

976 **7.7.4.2.4 APE\_REQ.2.4C**

977 **The statement of security requirements shall identify all operations on the security**  
978 **requirements.**

979 **7.7.4.2.5 APE\_REQ.2.5C**

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

981 **7.7.4.2.6 APE\_REQ.2.6C**

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

984 **7.7.4.2.7 APE\_REQ.2.7C**

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

987 **7.7.4.2.8 APE\_REQ.2.8C**

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

990 **7.7.4.2.9 APE\_REQ.2.9C**

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

992 **7.7.4.2.10 APE\_REQ.2.10C**

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

994 **7.7.4.2.11 APE\_REQ.2.11C**

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

996 **7.7.4.3 Evaluator action elements**997 **7.7.4.3.1 APE\_REQ.2.1E**

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

1000 **8 Class ACE: Protection Profile Configuration evaluation**1001 **8.1 Introduction**

1002 Evaluating a PP-Configuration is required to demonstrate that the PP-Configuration is sound and  
 1003 consistent. These properties are necessary for the PP-Configuration to be suitable for use as the basis  
 1004 for writing an ST.

1005 The class ACE is defined for the evaluation of a PP-Configuration composed of PPs and PP-Modules<sup>1</sup>.  
 1006 The evaluation of PPs is addressed in Class APE. The present class ACE defines the requirements for

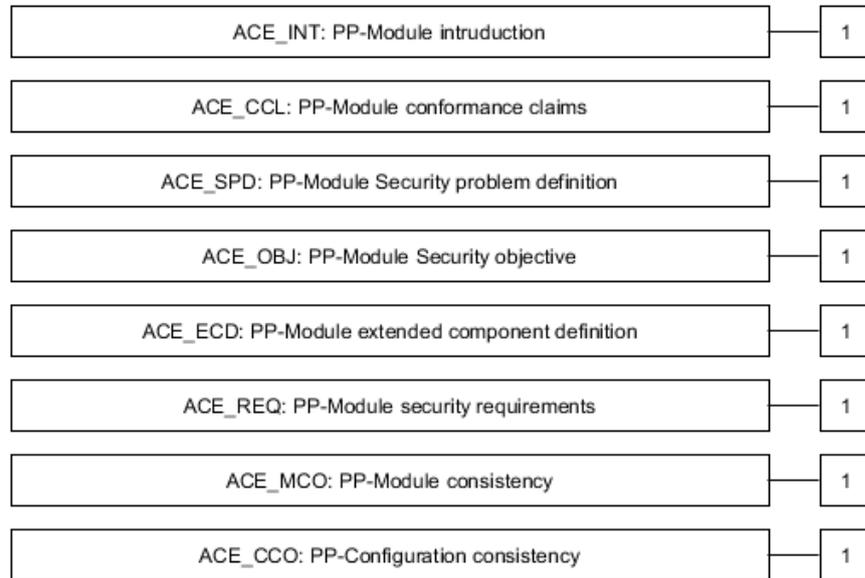
- 1007 • Evaluating the PP-Modules under the assumption that their base PPs/PP-Modules is internally  
 1008 consistent.
- 1009 • Evaluating the consistency of the combination of all the PPs and PP-Modules that belong to the  
 1010 PP-Configuration.

---

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

1011 The evaluator shall decide the order in which the unevaluated components of a PP-Configuration (PPs  
1012 and PP-Modules) are evaluated.

1013 This Clause should be used in conjunction with Annexes B and D in ISO/IEC 1540-1, as these Annexes  
1014 clarify the concepts and provide examples.



1015

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

1016

## 1017 **8.2 PP-Module introduction (ACE\_INT)**

### 1018 **8.2.1 Objectives**

1019 The objective of this family is to describe the TOE in a narrative way.

1020 The evaluation of the PP-Module introduction is required to demonstrate that the PP-Module is  
1021 correctly identified, and that the PP-Module reference and TOE overview are consistent with each  
1022 other.

### 1023 **8.2.2 ACE\_INT.1 PP-Module introduction**

1024 Dependencies: No dependencies.

#### 1025 **8.2.2.1 Developer action elements**

##### 1026 **8.2.2.1.1 ACE\_INT.1.1D**

1027 **The developer shall provide a PP-Module introduction.**

#### 1028 **8.2.2.2 Content and presentation elements**

##### 1029 **8.2.2.2.1 ACE\_INT.1.1C**

1030 **The PP-Module introduction shall contain a PP-Module reference and a TOE overview.**

- 1031 **8.2.2.2.2 ACE\_INT.1.2C**
- 1032 **The PP-Module reference shall uniquely identify the PP-Module.**
- 1033 **8.2.2.2.3 ACE\_INT.1.3C**
- 1034 **The TOE overview shall summarise the usage and major security features of the TOE.**
- 1035 **8.2.2.2.4 ACE\_INT.1.4C**
- 1036 **8.2.2.2.5 The TOE overview shall identify the TOE type.**
- 1037 **8.2.2.2.6 ACE\_INT.1.5C**
- 1038 **The TOE overview shall identify any non-TOE hardware/software/firmware available to the**  
1039 **TOE.**
- 1040 **8.2.2.2.7 ACE\_INT.1.6C**
- 1041 **The PP-Module introduction shall uniquely identify the base PPs and PP-Modules it depends**  
1042 **on.**
- 1043 **8.2.2.2.8 ACE\_INT.1.7C**
- 1044 **The PP-Module introduction shall describe the dependency structure of the base PPs and PP-**  
1045 **Modules.**
- 1046 **8.2.2.2.9 ACE\_INT.1.8C**
- 1047 **The TOE overview shall describe the differences of the TOE with regard to the TOEs defined in**  
1048 **the base PPs and PP-Modules.**
- 1049 **8.2.2.3 Evaluator action elements**
- 1050 **8.2.2.3.1 ACE\_INT.1.1E**
- 1051 **The evaluator shall confirm that the information provided meets all requirements for content**  
1052 **and presentation of evidence.**
- 1053 **8.3 PP-Module conformance claims (ACE\_CCL)**
- 1054 **8.3.1 Objectives**
- 1055 The objective of this family is to determine the validity of the conformance claim and conformance  
1056 statement. Unlike standard Protection Profiles, a PP-Module cannot claim conformance to another PP  
1057 or PP-Module.
- 1058 **8.3.2 ACE\_CCL.1 PP-Module conformance claims**
- 1059 Dependencies: ACE\_INT.1 PP-Module introduction
- 1060 ACE\_ECD.1 PP-Module extended components definition
- 1061 ACE\_REQ.1 PP-Module stated security requirements or ACE\_REQ.2 PP-Module  
1062 security requirements

- 1063 **8.3.2.1.1 ACE\_CCL.1.1D**
- 1064 **The developer shall provide a conformance claim.**
- 1065 **8.3.2.1.2 ACE\_CCL.1.2D**
- 1066 **The developer shall provide a conformance statement.**
- 1067 **8.3.2.2 Content and presentation elements**
- 1068 **8.3.2.2.1 ACE\_CCL.1.1C**
- 1069 **The conformance claim shall contain an ISO/IEC 15408 conformance claim that identifies the**  
1070 **ISO/IEC 15408-1 edition to which the PP-Module claims conformance.**
- 1071 **8.3.2.2.2 ACE\_CCL.1.2C**
- 1072 **ISO/IEC 15408 conformance claim shall describe the conformance of the PP-Module to ISO/IEC**  
1073 **15408-2 as either ISO/IEC 15408-2 conformant or ISO/IEC 15408-2 extended.**
- 1074 **8.3.2.2.3 ACE\_CCL.1.3C**
- 1075 **The ISO/IEC 15408 conformance claim shall describe the conformance of the PP-Module to this**  
1076 **document as either “ISO/IEC 15408-3 conformant” or ISO/IEC 15408-3 extended.”**
- 1077 **8.3.2.2.4 ACE\_CCL.1.4C**
- 1078 **ISO/IEC 15408 conformance claim shall be consistent with the extended components**  
1079 **definition.**
- 1080 **8.3.2.2.5 ACE\_CCL.1.5C**
- 1081 **The conformance claim shall identify all security requirement packages to which the PP claims**  
1082 **conformance.**
- 1083 **8.3.2.2.6 ACE\_CCL.1.6C**
- 1084 **The conformance claim shall describe any conformance of the PP-Module to a package as either**  
1085 **package-conformant or package-augmented.**
- 1086 **8.3.2.2.7 ACE\_CCL.1.7C**
- 1087 **The conformance statement shall describe the conformance required of any PP-**  
1088 **Configuration/ST to the PP-Module as one of exact, strict, or demonstrable.**
- 1089 **8.3.2.2.8 ACE\_CCL.1.8C**
- 1090 **The conformance statement shall identify the set of PPs and PP-Modules to which, in**  
1091 **combination with the PP-Module under evaluation, exact conformance is allowed to be claimed.**
- 1092 **8.3.2.2.9 ACE\_CCL.1.10C**
- 1093 **The conformance statement shall identify the set of derived Evaluation Methods and Evaluation**  
1094 **Activities (if any) that shall be used with the PP-Module under evaluation. This list shall contain**  
1095 **any Evaluation Methods and Evaluation Activities that are specified in the PP-Module but also**

1096 **any Evaluation Activities and Evaluation Methods specified in the base PPs and/or PP-modules**  
 1097 **and/or in the packages (if any) for which conformance is being claimed by the PP-Module**  
 1098 **under evaluation.**

1099 **8.3.2.3 Evaluator action elements**

1100 **8.3.2.3.1 ACE\_CCL.1.1E**

1101 **The evaluator shall confirm that the information provided meets all requirements for content**  
 1102 **and presentation of evidence.**

1103 **8.4 PP-Module Security problem definition (ACE\_SPD)**

1104 **8.4.1 Objectives**

1105 This part of the PP-Module defines the security problem to be addressed by the TOE and the  
 1106 operational environment of the TOE.

1107 Evaluation of the security problem definition is required to demonstrate that the security problem  
 1108 intended to be addressed by the TOE and its operational environment, is clearly defined.

1109 **8.4.2 ACE\_SPD.1 PP-Module Security problem definition**

1110 Dependencies: No dependencies.

1111 **8.4.2.1 Developer action elements**

1112 **8.4.2.1.1 ACE\_SPD.1.1D**

1113 **The developer shall provide a security problem definition.**

1114 **8.4.2.2 Content and presentation elements**

1115 **8.4.2.2.1 ACE\_SPD.1.1C**

1116 **The security problem definition shall describe the threats.**

1117 **8.4.2.2.2 ACE\_SPD.1.2C**

1118 **All threats shall be described in terms of a threat agent, an asset, and an adverse action.**

1119 **8.4.2.2.3 ACE\_SPD.1.3C**

1120 **The security problem definition shall describe the OSPs.**

1121 **8.4.2.2.4 ACE\_SPD.1.4C**

1122 The security problem definition shall describe the assumptions about the operational environment of  
 1123 the TOE.

1124 **8.4.2.3 Evaluator action elements**

1125 **8.4.2.3.1 ACE\_SPD.1.1E**

1126 **The evaluator shall confirm that the information provided meets all requirements for content**  
 1127 **and presentation of evidence.**

- 1128 **8.5 PP-Module Security objectives (ACE\_OBJ)**
- 1129 **8.5.1 Objectives**
- 1130 The security objectives are a concise statement of the intended response to the security problem  
1131 defined through the Security problem definition (APE\_SPD) family.
- 1132 Evaluation of the security objectives is required to demonstrate that the security objectives adequately  
1133 and completely address the security problem definition and that the division of this problem between  
1134 the TOE and its operational environment is clearly defined.
- 1135 **8.5.2 Component levelling**
- 1136 The components in this family are levelled on whether they prescribe only security objectives for the  
1137 operational environment (see ACE\_OBJ.1), or also security objectives for the TOE (see ACE\_OBJ.2).
- 1138 **8.5.3 ACE\_OBJ.1 Direct Rationale PP-Module Security objectives**
- 1139 Dependencies: No dependencies.
- 1140 **8.5.4 Application notes**
- 1141 If the PP-Module uses the Direct Rationale approach then all the elements defined in ACE\_OBJ.1 hold.
- 1142 **8.5.4.1 Developer action elements**
- 1143 **8.5.4.1.1 ACE\_OBJ.1.1D**
- 1144 **The developer shall provide a statement of security objectives for the PP-Module.**
- 1145 **8.5.4.2 Content and presentation elements**
- 1146 **8.5.4.2.1 ACE\_OBJ.1.1C**
- 1147 **The statement of security objectives shall describe the security objectives for the operational**  
1148 **environment.**
- 1149 **8.5.4.3 Evaluator action elements**
- 1150 **8.5.4.3.1 ACE\_OBJ.1.1E**
- 1151 **The evaluator shall confirm that the information provided meets all requirements for content**  
1152 **and presentation of evidence.**
- 1153 **8.5.5 ACE\_OBJ.2 PP-Module Security objectives**
- 1154 Dependencies: ACE\_SPD.1 PP-Module security problem definition.
- 1155 **8.5.6 Application notes**
- 1156 If the PP-Module does not use the Direct Rationale approach then all elements of ACE\_OBJ.2 hold.

1157 **8.5.6.1 Developer action elements**

1158 **8.5.6.1.1 ACE\_OBJ.2.1D**

1159 **The developer shall provide a statement of security objectives for the PP-Module.**

1160 **8.5.6.1.2 ACE\_OBJ.2.2D**

1161 **The developer shall provide a security objectives rationale for the PP-Module.**

1162 **8.5.6.2 Content and presentation elements**

1163 **8.5.6.2.1 ACE\_OBJ.2.1C**

1164 **The statement of security objectives shall describe the security objectives for the TOE and the**  
1165 **security objectives for the operational environment.**

1166 **8.5.6.2.2 ACE\_OBJ.2.2C**

1167 **The security objectives rationale shall trace each security objective for the TOE back to threats**  
1168 **countered by that security objective and OSPs enforced by that security objective.**

1169 **8.5.6.2.3 ACE\_OBJ.2.3C**

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

1173 **8.5.6.2.4 ACE\_OBJ.2.4C**

1174 **The security objectives rationale shall demonstrate that the security objectives counter all**  
1175 **threats.**

1176 **8.5.6.2.5 ACE\_OBJ.2.5C**

1177 **The security objectives rationale shall demonstrate that the security objectives enforce all**  
1178 **OSPs.**

1179 **8.5.6.2.6 ACE\_OBJ.2.6C**

1180 **The security objectives rationale shall demonstrate that the security objectives for the operational**  
1181 **environment uphold all assumptions.**

1182 **8.5.6.3 Evaluator action elements**

1183 **8.5.6.3.1 ACE\_OBJ.2.1E**

1184 **The evaluator shall confirm that the information provided meets all requirements for content**  
1185 **and presentation of evidence.**

1186 **8.6 PP-Module extended components definition (ACE\_ECD)**

1187 **8.6.1 Objectives**

1188 Extended security functional requirements are requirements that are not based on components from  
1189 **ISO/IEC 15408-2**, but are based on extended components: components defined by the PP-Module  
1190 author.

1191 Evaluation of the definition of extended functional components is necessary to determine that they are  
1192 clear and unambiguous, and that they are necessary, i.e. they may not be clearly expressed using  
1193 existing **ISO/IEC 15408-2** components.

1194 **8.6.2 ACE\_ECD.1 PP-Module extended components definition**

1195 Dependencies: No dependencies.

1196 **8.6.2.1 Developer action elements**

1197 **8.6.2.1.1 ACE\_ECD.1.1D**

1198 **The developer shall provide a statement of security requirements for the PP-Module.**

1199 **8.6.2.1.2 ACE\_ECD.1.2D**

1200 **The developer shall provide an extended components definition for the PP-Module.**

1201 **8.6.2.2 Content and presentation elements**

1202 **8.6.2.2.1 ACE\_ECD.1.1C**

1203 **The statement of security requirements shall identify all the extended security requirements.**

1204 **8.6.2.2.2 ACE\_ECD.1.2C**

1205 **The extended components definition shall define an extended component for each extended**  
1206 **security requirement.**

1207 **8.6.2.2.3 ACE\_ECD.1.3C**

1208 **The extended components definition shall describe how each extended component is related to**  
1209 **the existing ISO/IEC 15408 components, families, and classes.**

1210 **8.6.2.2.4 ACE\_ECD.1.4C**

1211 **The extended components definition shall use the existing ISO/IEC 15408 components,**  
1212 **families, classes, and methodology as a model for presentation.**

1213 **8.6.2.2.5 ACE\_ECD.1.5C**

1214 **8.6.2.2.6 The extended components shall consist of measurable and objective elements such**  
1215 **that conformance or nonconformance to these elements can be demonstrated.**

1216 **8.6.2.3 Evaluator action elements**

1217 **8.6.2.3.1 ACE\_ECD.1.1E**

1218 **The evaluator shall confirm that the information provided meets all requirements for content**  
1219 **and presentation of evidence.**

1220 **8.6.2.3.2 ACE\_ECD.1.2E**

1221 **The evaluator shall confirm that no extended component may be clearly expressed using**  
1222 **existing components.**

1223 **8.7 PP-Module security requirements (ACE\_REQ)**

1224 **8.7.1 Objectives**

1225 The SFRs form a clear, unambiguous and well-defined description of the expected security behaviour  
1226 of the TOE. The SARs form a clear, unambiguous and well-defined description of the expected activities  
1227 that will be undertaken to gain assurance in the TOE.

1228 Evaluation of the security requirements is required to ensure that they are clear, unambiguous and  
1229 well-defined.

1230 **8.7.2 Component levelling**

1231 The components in this family are levelled on whether they are stated as is (see ACE\_REQ.1), or  
1232 whether the SFRs are derived from security objectives for the TOE (see ACE\_REQ.2.).

1233 **Editor's note:**

1234 **The title of ACE\_REQ.1 is confusing. We propose to rename it as "Direct rationale PP-Module security**  
1235 **requirements".**

1236 **Unless experts pronounce themselves against this proposal, this change will be made in the next draft.**

1237 **The same applies to the title of APE\_REQ.1**

1238 **8.7.3 ACE\_REQ.1 PP-Module stated security requirements**

1239 Dependencies: APE\_ECD.1 Extended components definition

1240 ACE\_SPD.1 PP-Module security problem definition

1241 **8.7.3.1 Developer action elements**

1242 **8.7.3.1.1 ACE\_REQ.1.1D**

1243 **The developer shall provide a statement of security requirements for the PP-Module.**

1244 **8.7.3.1.2 ACE\_REQ.1.2D**

1245 **The developer shall provide a security requirements rationale for the PP-Module.**

1246 **8.7.3.2 Content and presentation elements**

1247 **8.7.3.2.1 ACE\_REQ.1.1C**

1248 **The statement of security requirements shall describe the SFRs and the SARs.**

1249 **8.7.3.2.2 ACE\_REQ.1.2C**

1250 **All subjects, objects, operations, security attributes, external entities and other terms that are**  
1251 **used in the SFRs and the SARs shall be defined.**

1252 **8.7.3.2.3 ACE\_REQ.1.3C**

1253 **The statement of security requirements shall include a natural language description, part of**  
1254 **which describes how the SFRs combine together to provide security functionality in terms of**  
1255 **the architecture that is observable to Administrators and other users, or in terms of internal**  
1256 **features or properties.**

1257 **8.7.3.2.4 ACE\_REQ.1.4C**

1258 **The statement of security requirements shall identify all operations on the security**  
1259 **requirements.**

1260 **8.7.3.2.5 ACE\_REQ.1.5C**

1261 **All operations shall be performed correctly.**

1262 **8.7.3.2.6 ACE\_REQ.1.6C**

1263 **Each dependency of the security requirements shall either be satisfied, or the security**  
1264 **requirements rationale shall justify the dependency not being satisfied.**

1265 **8.7.3.2.7 ACE\_REQ.1.7C**

1266 **The security requirements rationale shall trace each SFR back to the threats countered by that**  
1267 **SFR and the OSPs enforced by that SFR.**

1268 **8.7.3.2.8 ACE\_REQ.1.8C**

1269 **The security requirements rationale shall trace each security objective for the operational**  
1270 **environment back to the threats countered by that security objective, the OSPs enforced by that**  
1271 **security objective, and the assumptions upheld by that security objective.**

1272 **8.7.3.2.9 ACE\_REQ.1.9C**

1273 **The security requirements rationale shall demonstrate that the SFRs (in conjunction with the**  
1274 **security objectives for the environment) counter all the threats for the TOE.**

1275 **8.7.3.2.10 ACE\_REQ.1.10C**

1276 **The security requirements rationale shall demonstrate that the SFRs (in conjunction with the**  
 1277 **security objectives for the environment) enforce all the OSPs for the TOE.**

1278 **8.7.3.2.11 ACE\_REQ.1.11C**

1279 **The security requirements rationale shall demonstrate that the security objectives for the**  
 1280 **operational environment uphold all assumptions.**

1281 **8.7.3.2.12 ACE\_REQ.1.12C**

1282 The statement of security requirements shall be internally consistent.

1283 **8.7.3.3 Evaluator action elements**1284 **8.7.3.3.1 ACE\_REQ.1.1E**

1285 **The evaluator shall confirm that the information provided meets all requirements for content**  
 1286 **and presentation of evidence.**

1287 **8.7.4 ACE\_REQ.2 PP-Module derived security requirements**

1288 Dependencies: ACE\_ECD.1 PP-Module extended components definition

1289 ACE\_OBJ.1 PP-Module Security objectives

1290 **8.7.4.1 Developer action elements**1291 **8.7.4.1.1 ACE\_REQ.2.1D**

1292 **The developer shall provide a statement of security requirements for the PP-Module.**

1293 **8.7.4.1.2 ACE\_REQ.2.2D**

1294 **The developer shall provide a security requirement rationale for the PP-Module.**

1295 **8.7.4.2 Content and presentation elements**1296 **8.7.4.2.1 ACE\_REQ.2.1C**

1297 **The statement of security requirements shall describe the SFRs and the SARs.**

1298 **8.7.4.2.2 ACE\_REQ.2.2C**

1299 **All subjects, objects, operations, security attributes, external entities and other terms that are**  
 1300 **used in the SFRs and the SARs shall be defined.**

1301 **8.7.4.2.3 ACE\_REQ.2.3C**

1302 **The statement of security requirements shall include a natural language description, part of**  
 1303 **which describes how the SFRs combine together to provide security functionality in terms of**  
 1304 **the architecture that is observable to Administrators and other users, or in terms of internal**  
 1305 **features or properties.**

- 1306 **8.7.4.2.4 ACE\_REQ.2.4C**
- 1307 **The statement of security requirements shall identify all operations on the security**  
1308 **requirements.**
- 1309 **8.7.4.2.5 ACE\_REQ.2.5C**
- 1310 **All operations shall be performed correctly.**
- 1311 **8.7.4.2.6 ACE\_REQ.2.6C**
- 1312 **Each dependency of the security requirements shall either be satisfied, or the security**  
1313 **requirements rationale shall justify the dependency not being satisfied.**
- 1314 **8.7.4.2.7 ACE\_REQ.2.7C**
- 1315 **The security requirements rationale shall trace each SFR back to the security objectives for the**  
1316 **TOE and OSPs enforced by that SFR.**
- 1317 **8.7.4.2.8 ACE\_REQ.2.8C**
- 1318 **The security requirements rationale shall demonstrate that the SFRs meet all security**  
1319 **objectives for the TOE.**
- 1320 **8.7.4.2.9 ACE\_REQ.2.9C**
- 1321 **The security requirements rationale shall demonstrate that the SFRs enforce all OSPs.**
- 1322 **8.7.4.2.10 ACE\_REQ.2.10C**
- 1323 **The security requirements rationale shall explain why the SARs were chosen.**
- 1324 **8.7.4.2.11 ACE\_REQ.2.11C**
- 1325 **The statement of security requirements shall be internally consistent.**
- 1326 **8.7.4.3 Evaluator action elements**
- 1327 **8.7.4.3.1 ACE\_REQ.2.1E**
- 1328 **The evaluator shall confirm that the information provided meets all requirements for content**  
1329 **and presentation of evidence.**
- 1330 **8.8 PP-Module consistency (ACE\_MCO)**
- 1331 **8.8.1 Objectives**
- 1332 **The objective of this family is to determine the consistency of the PP-Module.**
- 1333 **8.8.2 ACE\_MCO.1 PP-Module consistency**
- 1334 **Dependencies: ACE\_INT.1 PP-Module introduction**
- 1335 **ACE\_SPD.1 PP-Module Security problem definition**

1336 ACE\_OBJ.1 Direct Rationale PP-Module Security objectives for the environment or  
1337 ACE\_OBJ.2 PP-Module Security objectives

1338 ACE\_REQ.1 PP-Module stated security requirements or ACE\_REQ.2 PP-Module  
1339 security requirements

## 1340 8.8.2.1 Developer action elements

### 1341 8.8.2.1.1 ACE\_MCO.1.1D

1342 **The developer shall provide a consistency rationale of the PP-Module for each of the alternative**  
1343 **sets of Base-PPs and PP-Modules identified in the PP-Module introduction.**

## 1344 8.8.2.2 Content and presentation elements

### 1345 8.8.2.2.1 ACE\_MCO.1.1C

1346 **The consistency rationale shall demonstrate that the TOE type of the PP-Module and the TOE**  
1347 **types of its base PPs and PP-Modules are consistent.**

### 1348 8.8.2.2.2 ACE\_MCO.1.2C

1349 **Editor's Note: this is also meaningful for APE and ASE when the ST claims conformance to more than one PP**  
1350 **or when the ST adds elements to the PPs it conforms to: The change has not been proposed yet in ASE/APE,**  
1351 **but if experts agree, we suggest cascading this change in the next CD.**

1352 **The consistency rationale shall identify the assets of the PP-Module that also belong to some of**  
1353 **its base PP(s) and/or PP-Module(s) and amongst them those for which the PP-Module and the**  
1354 **base PP(s) and PP-Module(s) define different security problems.**

### 1355 8.8.2.2.3 ACE\_MCO.1.3C

1356 **The consistency rationale shall demonstrate that the security problem definition of the PP-**  
1357 **Module and the security problem definition of its base PPs and PP-Modules are consistent.**

### 1358 8.8.2.2.4 ACE\_MCO.1.4C

1359 **The consistency rationale shall demonstrate that the security objectives of the PP-Module and**  
1360 **the security objectives of its base PPs and PP-Modules are consistent.**

### 1361 8.8.2.2.5 ACE\_MCO.1.5C

1362 **The consistency rationale shall demonstrate that the security functional requirements of the**  
1363 **PP-Module and the security functional requirements of its base PPs and PP-Modules are**  
1364 **consistent.**

### 1365 8.8.2.2.6 ACE\_MCO.1.6C

1366 **The consistency rationale shall demonstrate that the security assurance requirements of the**  
1367 **PP-Module and the security assurance requirements of its base PPs and PP-Modules are**  
1368 **consistent.**

1369 **8.8.2.3 Evaluator action elements**1370 **8.8.2.3.1 ACE\_MCO.1.1E**

1371 **The evaluator shall confirm that the information provided meets all requirements for content**  
 1372 **and presentation of evidence. If the PP-Module specifies alternative sets of Base-PPs and PP-**  
 1373 **Modules, the evaluator shall perform this action for each consistency rationale.**

1374 **8.9 PP-Configuration consistency (ACE\_CCO)**1375 **8.9.1 Objectives**

1376 The objective of this family is to determine the well-formedness and the consistency of the PP-  
 1377 Configuration.

1378 **8.9.2 ACE\_CCO.1 PP-Configuration consistency**

1379 Dependencies: ACE\_INT.1 PP-Module introduction

1380 ACE\_CCL.1 PP-Module conformance claims

1381 ACE\_SPD.1 PP-Module security problem definition

1382 ACE\_OBJ.1 Direct Rationale PP-Module security objectives for the environment or  
 1383 ACE\_OBJ.2 PP-Module Security objectives

1384 ACE\_ECD.1 PP-Module extended component definition

1385 ACE\_REQ.1 PP-Module stated security requirements or ACE\_REQ.2 PP-Module  
 1386 security requirements

1387 ACE\_MCO.1 PP-Module consistency

1388 APE\_\*

1389 **8.9.2.1 Developer action elements**1390 **8.9.2.1.1 ACE\_CCO.1.1D**

1391 **The developer shall provide the reference of the PP-Configuration.**

1392 **8.9.2.1.2 ACE\_CCO.1.2D**

1393 **The developer shall provide a components list.**

1394 **8.9.2.1.3 ACE\_CCO.1.3D**

1395 **The developer shall provide a TOE overview.**

1396 **8.9.2.1.4 ACE\_CCO.1.4D**

1397 **The developer shall provide a conformance claim.**

- 1398 **8.9.2.1.5 ACE\_CCO.1.5D**
- 1399 **The developer shall provide a conformance statement.**
- 1400 **8.9.2.1.6 ACE\_CCO.1.7D**
- 1401 **The developer shall provide a consistency rationale.**
- 1402 **8.9.2.2 Content and presentation elements**
- 1403 **8.9.2.2.1 ACE\_CCO.1.1C**
- 1404 **The PP-Configuration reference shall uniquely identify the PP-Configuration.**
- 1405 **8.9.2.2.2 ACE\_CCO.1.2C**
- 1406 **The components list shall uniquely identify the PPs and PP-Modules that compose the PP-**  
1407 **Configuration.**
- 1408 **8.9.2.2.3 ACE\_CCO.1.3C**
- 1409 **For each PP-Module identified in the components list of the PP-Configuration, the list contains**  
1410 **at least one of its sets of base PPs and PP-Modules.**
- 1411 **8.9.2.3 ACE\_CCO.1.4C**
- 1412 **The TOE overview shall identify the TOE type.**
- 1413 **8.9.2.4 ACE\_CCO.1.5C**
- 1414 **The TOE overview shall describe the organisation of the TOE in terms of the sub-TSFs (TSF**  
1415 **parts) defined in the PPs and PP-Modules that belong to the components list.**
- 1416 **8.9.2.5 ACE\_CCO.1.6C**
- 1417 **The conformance claim shall contain an ISO/IEC 15408 conformance claim that identifies the**  
1418 **ISO/IEC 15408-1 edition(s) to which the PPs and PP-Modules that compose the PP-**  
1419 **Configuration claim conformance.**
- 1420 **8.9.2.6 ACE\_CCO.1.7C**
- 1421 **ISO/IEC 15408 conformance claim shall describe the conformance of the PP-Configuration to**  
1422 **ISO/IEC 15408-2 as either ISO/IEC 15408-2 conformant or ISO/IEC 15408-2 extended.**
- 1423 **8.9.2.7 ACE\_CCO.1.8C**
- 1424 **The ISO/IEC 15408 conformance claim shall describe the conformance of the PP-Configuration**  
1425 **to this document as either “ISO/IEC 15408-3 conformant” or ISO/IEC 15408-3 extended.”**
- 1426 **8.9.2.8 ACE\_CCO.1.9C**
- 1427 **ISO/IEC 15408 conformance claim shall be consistent with the extended components definition**  
1428 **of the composing PPs and PP-Modules.**

1429 **8.9.2.9 ACE\_CCO.1.10C**

1430 **The conformance statement shall specify the required conformance to the PP-Configuration as**  
 1431 **one of exact, strict, demonstrable, or list of strict and demonstrable types inherited from its**  
 1432 **composing PPs and PP-Modules.**

1433 **8.9.2.10 ACE\_CCO.1.11C**

1434 **The conformance statement of a PP-Configuration of strict, demonstrable, or strict and**  
 1435 **demonstrable conformance shall define the applicable SARs/assurance packages:**

- 1436 • **The global set of SARs/assurance package that applies to the entire TOE.**
- 1437 • **For each sub-TSF (TSF part) defined in the composing PPs and PP-Modules, the**  
 1438 **applicable set of SARs/assurance package.**

1439 **8.9.2.11 ACE\_CCO.1.12C**

1441 **The conformance statement of a PP-Configuration of exact conformance type shall identify the**  
 1442 **set of derived Evaluation Methods and Evaluation Activities (if any) that shall be used with the**  
 1443 **PP under evaluation. This list shall contain any Evaluation Methods and Evaluation Activities**  
 1444 **that are specified in the PP it but also any Evaluation Activities and Evaluation Methods**  
 1445 **specified in PPs and/or PP-modules and/or packages for which conformance is being claimed**  
 1446 **by the PP under evaluation.**

1447 **8.9.2.1 ACE\_CCO.1.13C**

1448 **The consistency rationale shall demonstrate that the TOE type defined in the PP-Configuration**  
 1449 **is consistent with the TOE types defined in the PPs and PP-Modules that belong to the PP-**  
 1450 **Configuration components list.**

1451 **8.9.2.2 ACE\_CCO.1.14C**

1452 **The consistency rationale shall demonstrate that the union of all the SPDs, security objectives**  
 1453 **and security functional requirements defined in the PPs and PP-Modules of the PP-**  
 1454 **Configuration components list is consistent.**

1455 **8.9.2.3 ACE\_CCO.1.15C**

1456 **The consistency rationale of a PP-Configuration of strict, demonstrable, or strict and**  
 1457 **demonstrable conformance type shall demonstrate**

- 1458 • **the consistency of the global set of SARs/assurance package with regard to the threat**  
 1459 **models as defined in the SPDs of the component PPs and PP-Modules, and**
- 1460 • **the consistency of the global set of SARs/assurance package and all the sets of**  
 1461 **SARs/assurance packages for the sub-TSFs (TSF parts) with each other.**

1462 **8.9.2.4 Evaluator action elements**1463 **8.9.2.4.1 ACE\_CCO.1.1E**

1464 **The evaluator shall confirm that the information provided meets all requirements for content**  
 1465 **and presentation of evidence.**

1466 **8.9.2.4.2 ACE\_CCO.1.2E**

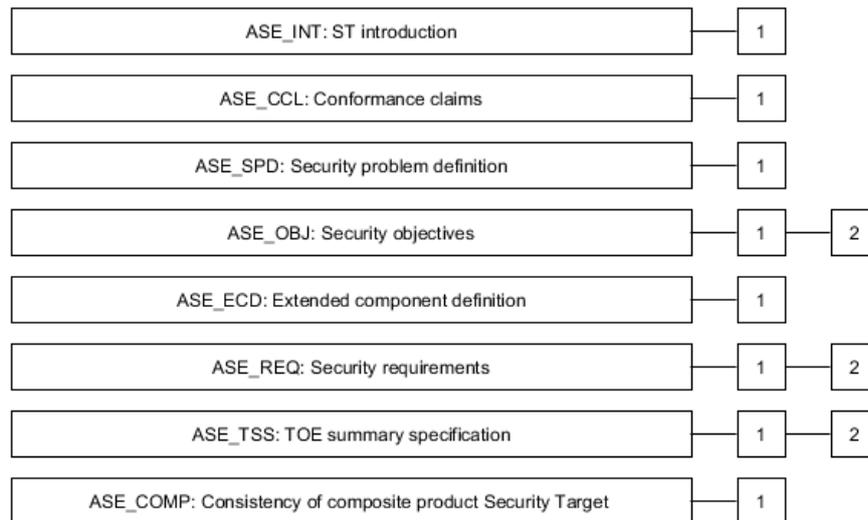
1467 **The evaluator shall check that the PP-Configuration consisting of all the PPs and PP-Modules**  
 1468 **identified in the components list is consistent.**

1469 **9 Class ASE: Security Target evaluation**1470 **9.1 Introduction**

1471 Evaluating an ST is required to demonstrate that the ST is sound and internally consistent, and, if the  
 1472 ST is based on a PP-Configuration, or one or more PPs or packages, that the ST is a correct  
 1473 instantiation of the PP-Configuration, PPs, and packages. These properties are necessary for the ST to  
 1474 be suitable for use as the basis for a TOE evaluation.

1475 This clause should be used in conjunction with Annexes A, B and C in ISO/IEC 15408-1:XXXX, as these  
 1476 annexes clarify the concepts here and provide many examples.

1477 Figure 6 shows the families within this class, and the hierarchy of components within the families.



1478

1479 **Figure 6 — ASE: Security Target evaluation class decomposition**

1480 **9.2 ST introduction (ASE\_INT)**1481 **9.2.1 Objectives**

1482 The objective of this family is to describe the TOE in a narrative way on three levels of abstraction:  
 1483 TOE reference, TOE overview and TOE description.

1484 Evaluation of the ST introduction is required to demonstrate that the ST and the TOE are correctly  
 1485 identified, that the TOE is correctly described at three levels of abstraction and that these three  
 1486 descriptions are consistent with each other.

1487 **9.2.2 ASE\_INT.1 ST introduction**

1488 Dependencies: No dependencies.

- 1489 **9.2.2.1 Developer action elements**
- 1490 **9.2.2.1.1 ASE\_INT.1.1D**
- 1491 **The developer shall provide an ST introduction.**
- 1492 **9.2.2.2 Content and presentation elements**
- 1493 **9.2.2.2.1 ASE\_INT.1.1C**
- 1494 **The ST introduction shall contain an ST reference, a TOE reference, a TOE overview and a TOE**  
1495 **description.**
- 1496 **9.2.2.2.2 ASE\_INT.1.2C**
- 1497 **The ST reference shall uniquely identify the ST.**
- 1498 **9.2.2.2.3 ASE\_INT.1.3C**
- 1499 **The TOE reference shall uniquely identify the TOE.**
- 1500 **9.2.2.2.4 ASE\_INT.1.4C**
- 1501 **The TOE overview shall summarise the usage and major security features of the TOE.**
- 1502 **9.2.2.2.5 ASE\_INT.1.5C**
- 1503 **The TOE overview shall identify the TOE type.**
- 1504 **9.2.2.2.6 ASE\_INT.1.6C**
- 1505 **The TOE overview shall identify any non-TOE hardware/software/firmware required by the**  
1506 **TOE.**
- 1507 **9.2.2.2.7 ASE\_INT.1.7C**
- 1508 **The TOE description shall describe the physical scope of the TOE.**
- 1509 **9.2.2.2.8 ASE\_INT.1.8C**
- 1510 **The TOE description shall describe the logical scope of the TOE.**
- 1511 **9.2.2.3 Evaluator action elements**
- 1512 **9.2.2.3.1 ASE\_INT.1.1E**
- 1513 **The evaluator shall confirm that the information provided meets all requirements for content**  
1514 **and presentation of evidence.**
- 1515 **9.2.2.3.2 ASE\_INT.1.2E**
- 1516 **The evaluator shall confirm that the TOE reference, the TOE overview, and the TOE description**  
1517 **are consistent with each other.**

1518 **9.3 Conformance claims (ASE\_CCL)**

1519 **9.3.1 Objectives**

1520 The objective of this family is to determine the validity of the conformance claim. In addition, this  
1521 family specifies how STs are to claim conformance with the PP or PP-Configuration.

1522 **9.3.2 ASE\_CCL.1 Conformance claims**

1523 Dependencies: ASE\_INT.1 ST introduction

1524 ASE\_ECD.1 Extended components definition

1525 ASE\_REQ.1 Stated security requirements

1526 **9.3.2.1 Developer action elements**

1527 **9.3.2.1.1 ASE\_CCL.1.1D**

1528 **The developer shall provide a conformance claim.**

1529 **9.3.2.1.2 ASE\_CCL.1.2D**

1530 **The developer shall provide a conformance claim rationale.**

1531 **9.3.2.2 Content and presentation elements**

1532 **9.3.2.2.1 ASE\_CCL.1.1C**

1533 **The conformance claim shall contain an ISO/IEC 15408 conformance claim that identifies the**  
1534 **edition of ISO/IEC 15408 to which the ST and the TOE claim conformance.**

1535 **9.3.2.2.2 ASE\_CCL.1.2C**

1536 **ISO/IEC 15408 conformance claim shall describe the conformance of the ST to ISO/IEC 15408-2**  
1537 **as either ISO/IEC 15408-2 conformant or ISO/IEC 15408-2 extended.**

1538 **9.3.2.2.3 ASE\_CCL.1.3C**

1539 **ISO/IEC 15408 conformance claim shall describe the conformance of the ST to this document as**  
1540 **either “ISO/IEC 15408-3 conformant” or ISO/IEC 15408-3 extended.”**

1541 **9.3.2.2.4 ASE\_CCL.1.4C**

1542 **ISO/IEC 15408 conformance claim shall be consistent with the extended components**  
1543 **definition.**

1544 **9.3.2.2.5 ASE\_CCL.1.5C**

1545 **The conformance claim shall identify a PP-Configuration, or all PPs and security requirement**  
1546 **packages to which the ST claims conformance.**

1547 **9.3.2.2.6 ASE\_CCL.1.6C**

1548 **The conformance claim shall describe any conformance of the ST to a package as either**  
1549 **package-conformant or package-augmented.**

- 1550 **9.3.2.2.7 ASE\_CCL.1.7C**
- 1551 **The conformance claim rationale shall demonstrate that the TOE type is consistent with the**  
1552 **TOE type in the PP-Configuration or PPs for which conformance is being claimed.**
- 1553 **9.3.2.2.8 ASE\_CCL.1.8C**
- 1554 **The conformance claim rationale shall demonstrate that the statement of the security problem**  
1555 **definition is consistent with the statement of the security problem definition in the PP-**  
1556 **Configuration or PPs for which conformance is being claimed.**
- 1557 **9.3.2.2.9 ASE\_CCL.1.9C**
- 1558 **The conformance claim rationale shall demonstrate that the statement of security objectives is**  
1559 **consistent with the statement of security objectives in the PP-Configuration or PPs for which**  
1560 **conformance is being claimed.**
- 1561 **9.3.2.2.10 ASE\_CCL.1.10C**
- 1562 **The conformance claim rationale shall demonstrate that the statement of security**  
1563 **requirements is consistent with the statement of security requirements in the PP-Configuration**  
1564 **or PPs for which conformance is being claimed.**
- 1565 **9.3.2.2.11 ASE\_CCL.1.11C**
- 1566 The conformance claim for PP(s) and PP-Configuration(s) shall be exact, strict, or demonstrable.
- 1567 **9.3.2.3 Evaluator action elements**
- 1568 **9.3.2.3.1 ASE\_CCL.1.1E**
- 1569 **The evaluator shall confirm that the information provided meets all requirements for content**  
1570 **and presentation of evidence.**
- 1571 **9.4 Security problem definition (ASE\_SPD)**
- 1572 **9.4.1 Objectives**
- 1573 This part of the ST defines the security problem to be addressed by the TOE and the operational  
1574 environment of the TOE.
- 1575 Evaluation of the security problem definition is required to demonstrate that the security problem  
1576 intended to be addressed by the TOE and its operational environment, is clearly defined.
- 1577 **9.4.2 ASE\_SPD.1 Security problem definition**
- 1578 Dependencies: No dependencies.
- 1579 **9.4.2.1 Developer action elements**
- 1580 **9.4.2.1.1 ASE\_SPD.1.1D**
- 1581 **The developer shall provide a security problem definition.**

- 1582 **9.4.2.2 Content and presentation elements**
- 1583 **9.4.2.2.1 ASE\_SPD.1.1C**
- 1584 **The security problem definition shall describe the threats.**
- 1585 **9.4.2.2.2 ASE\_SPD.1.2C**
- 1586 **All threats shall be described in terms of a threat agent, an asset, and an adverse action.**
- 1587 **9.4.2.2.3 ASE\_SPD.1.3C**
- 1588 **The security problem definition shall describe the OSPs.**
- 1589 **9.4.2.2.4 ASE\_SPD.1.4C**
- 1590 **The security problem definition shall describe the assumptions about the operational**  
 1591 **environment of the TOE.**
- 1592 **9.4.2.3 Evaluator action elements**
- 1593 **9.4.2.3.1 ASE\_SPD.1.1E**
- 1594 **The evaluator shall confirm that the information provided meets all requirements for content**  
 1595 **and presentation of evidence.**
- 1596 **9.5 Security objectives (ASE\_OBJ)**
- 1597 **9.5.1 Objectives**
- 1598 The security objectives are a concise statement of the intended response to the security problem  
 1599 defined through the Security problem definition (ASE\_SPD) family.
- 1600 Evaluation of the security objectives is required to demonstrate that the security objectives adequately  
 1601 and completely address the security problem definition, that the division of this problem between the  
 1602 TOE and its operational environment is clearly defined.
- 1603 **9.5.2 Component levelling**
- 1604 The components in this family are levelled on whether they prescribe only security objectives for the  
 1605 operational environment, or also security objectives for the TOE.
- 1606 **9.5.3 ASE\_OBJ.1 Security objectives for the operational environment**
- 1607 Dependencies: No dependencies
- 1608 **9.5.3.1 Developer action elements**
- 1609 **9.5.3.1.1 ASE\_OBJ.1.1D**
- 1610 **The developer shall provide a statement of security objectives.**

- 1611 **9.5.3.2 Content and presentation elements**
- 1612 **9.5.3.2.1 ASE\_OBJ.1.1C**
- 1613 **The statement of security objectives shall describe the security objectives for the operational**  
1614 **environment.**
- 1615 **9.5.3.3 Evaluator action elements**
- 1616 **9.5.3.3.1 ASE\_OBJ.1.1E**
- 1617 **The evaluator shall confirm that the information provided meets all requirements for content**  
1618 **and presentation of evidence.**
- 1619 **9.5.4 ASE\_OBJ.2 Security objectives**
- 1620 Dependencies: ASE\_SPD.1 Security problem definition
- 1621 **9.5.4.1 Developer action elements**
- 1622 **9.5.4.1.1 ASE\_OBJ.2.1D**
- 1623 The developer shall provide a statement of security objectives.
- 1624 **9.5.4.1.2 ASE\_OBJ.2.2D**
- 1625 **The developer shall provide a security objectives rationale.**
- 1626 **9.5.4.2 Content and presentation elements**
- 1627 **9.5.4.2.1 ASE\_OBJ.2.1C**
- 1628 **The statement of security objectives shall describe the security objectives for the TOE and the**  
1629 **security objectives for the operational environment.**
- 1630 **9.5.4.2.2 ASE\_OBJ.2.2C**
- 1631 **The security objectives rationale shall trace each security objective for the TOE back to threats**  
1632 **countered by that security objective and OSPs enforced by that security objective.**
- 1633 **9.5.4.2.3 ASE\_OBJ.2.3C**
- 1634 **The security objectives rationale shall trace each security objective for the operational**  
1635 **environment back to threats countered by that security objective, OSPs enforced by that**  
1636 **security objective, and assumptions upheld by that security objective.**
- 1637 **9.5.4.2.4 ASE\_OBJ.2.4C**
- 1638 The security objectives **rationale** shall **demonstrate that** the security objectives **counter all threats.**
- 1639 **9.5.4.2.5 ASE\_OBJ.2.5C**
- 1640 **The security objectives rationale shall demonstrate that the security objectives enforce all**  
1641 **OSPs.**

1642 **9.5.4.2.6 ASE\_OBJ.2.6C**

1643 **The security objectives rationale shall demonstrate that the security objectives for the**  
 1644 **operational environment uphold all assumptions.**

1645 **9.5.4.3 Evaluator action elements**

1646 **9.5.4.3.1 ASE\_OBJ.2.1E**

1647 The evaluator shall confirm that the information provided meets all requirements for content and  
 1648 presentation of evidence.

1649 **9.6 Extended components definition (ASE\_ECD)**

1650 **9.6.1 Objectives**

1651 Extended security requirements are requirements that are not based on components from ISO/IEC  
 1652 15408-2 or this document, but are based on extended components: components defined by the ST  
 1653 author.

1654 Evaluation of the definition of extended components is necessary to determine that they are clear and  
 1655 unambiguous, and that they are necessary, i.e. they may not be clearly expressed using existing  
 1656 ISO/IEC 15408-2 or this document components.

1657 **9.6.2 ASE\_ECD.1 Extended components definition**

1658 Dependencies: No dependencies.

1659 **9.6.2.1 Developer action elements**

1660 **9.6.2.1.1 ASE\_ECD.1.1D**

1661 **The developer shall provide a statement of security requirements.**

1662 **9.6.2.1.2 ASE\_ECD.1.2D**

1663 **The developer shall provide an extended components definition.**

1664 **9.6.2.2 Content and presentation elements**

1665 **9.6.2.2.1 ASE\_ECD.1.1C**

1666 **The statement of security requirements shall identify all extended security requirements.**

1667 **9.6.2.2.2 ASE\_ECD.1.2C**

1668 **The extended components definition shall define an extended component for each extended**  
 1669 **security requirement.**

1670 **9.6.2.2.3 ASE\_ECD.1.3C**

1671 **The extended components definition shall describe how each extended component is related to**  
 1672 **the existing ISO/IEC 15408 components, families, and classes.**

1673 **9.6.2.2.4 ASE\_ECD.1.4C**

1674 **The extended components definition shall use the existing ISO/IEC 15408 components,**  
1675 **families, classes, and methodology as a model for presentation.**

1676 **9.6.2.2.5 ASE\_ECD.1.5C**

1677 **The extended components shall consist of measurable and objective elements such that**  
1678 **conformance or nonconformance to these elements can be demonstrated.**

1679 **9.6.2.3 Evaluator action elements**

1680 **9.6.2.3.1 ASE\_ECD.1.1E**

1681 **The evaluator shall confirm that the information provided meets all requirements for content**  
1682 **and presentation of evidence.**

1683 **9.6.2.3.2 ASE\_ECD.1.2E**

1684 **The evaluator shall confirm that no extended component can be clearly expressed using**  
1685 **existing components.**

1686 **9.7 Security requirements (ASE\_REQ)**

1687 **9.7.1 Objectives**

1688 The SFRs form a clear, unambiguous and well-defined description of the expected security behaviour  
1689 of the TOE. The SARs form a clear, unambiguous and canonical description of the expected activities  
1690 that will be undertaken to gain assurance in the TOE.

1691 Evaluation of the security requirements is required to ensure that they are clear, unambiguous and  
1692 well-defined.

1693 **9.7.2 Component levelling**

1694 The components in this family are levelled on whether they are stated as is.

1695 **9.7.3 ASE\_REQ.1 Stated security requirements**

1696 Dependencies: ASE\_ECD.1 Extended components definition

1697 **9.7.3.1 Developer action elements**

1698 **9.7.3.1.1 ASE\_REQ.1.1D**

1699 **The developer shall provide a statement of security requirements.**

1700 **9.7.3.1.2 ASE\_REQ.1.2D**

1701 **The developer shall provide a security requirements rationale.**

1702 **9.7.3.2 Content and presentation elements**

1703 **9.7.3.2.1 ASE\_REQ.1.1C**

1704 **The statement of security requirements shall describe the SFRs and the SARs.**

1705 **9.7.3.2.2 ASE\_REQ.1.2C**

1706 **All subjects, objects, operations, security attributes, external entities and other terms that are**  
1707 **used in the SFRs and the SARs shall be defined.**

1708 **9.7.3.2.3 ASE\_REQ.1.3C**

1709 **The statement of security requirements shall include a natural language description, part of**  
1710 **which describes how the SFRs combine together to provide security functionality in terms of**  
1711 **the architecture that is observable to Administrators and other users, or in terms of internal**  
1712 **features or properties.**

1713 **9.7.3.2.4 ASE\_REQ.1.4C**

1714 **The statement of security requirements shall identify all operations on the security**  
1715 **requirements.**

1716 **9.7.3.2.5 ASE\_REQ.1.5C**

1717 **All operations shall be performed correctly.**

1718 **9.7.3.2.6 ASE\_REQ.1.6C**

1719 **Each dependency of the security requirements shall either be satisfied, or the security**  
1720 **requirements rationale shall justify the dependency not being satisfied.**

1721 **9.7.3.2.7 ASE\_REQ.1.7C**

1722 **The security requirements rationale shall trace each SFR back to the security objectives threats**  
1723 **countered by that SFR and OSPs enforced by that SFR.**

1724 **9.7.3.2.8 ASE\_REQ.1.8C**

1725 **The security requirements rationale shall trace each security objective for the operational**  
1726 **environment back to threats countered by that security objective, OSPs enforced by that**  
1727 **security objective, and assumptions upheld by that security objective.**

1728 **9.7.3.2.9 ASE\_REQ.1.9C**

1729 **The security requirements rationale shall demonstrate that the SFRs counter all threats for the**  
1730 **TOE.**

1731 **9.7.3.2.10 ASE\_REQ.1.10C**

1732 **The security requirements rationale shall demonstrate that the SFRs enforce all OSPs.**

1733 **9.7.3.2.11 ASE\_REQ.1.11C**

1734 **The security requirements rationale shall demonstrate that the security objectives for the**  
1735 **operational environment uphold all assumptions.**

1736 **9.7.3.2.12 ASE\_REQ.1.12C**

1737 **The statement of security requirements shall be internally consistent.**

1738 **9.7.3.3 Evaluator action elements**

1739 **9.7.3.3.1 ASE\_REQ.1.1E**

1740 **The evaluator shall confirm that the information provided meets all requirements for content**  
 1741 **and presentation of evidence.**

1742 **9.7.4 ASE\_REQ.2 Derived security requirements**

1743 Dependencies: ASE\_OBJ.2 Security objectives

1744 ASE\_ECD.1 Extended components definition

1745 **9.7.4.1 Developer action elements**

1746 **9.7.4.1.1 ASE\_REQ.2.1D**

1747 The developer shall provide a statement of security requirements.

1748 **9.7.4.1.2 ASE\_REQ.2.2D**

1749 The developer shall provide a security requirements rationale.

1750 **9.7.4.2 Content and presentation elements**

1751 **9.7.4.2.1 ASE\_REQ.2.1C**

1752 The statement of security requirements shall describe the SFRs and the SARs.

1753 **9.7.4.2.2 ASE\_REQ.2.2C**

1754 All subjects, objects, operations, security attributes, external entities and other terms that are used in  
 1755 the SFRs and the SARs shall be defined.

1756 **9.7.4.2.3 ASE\_REQ.2.3C**

1757 The statement of security requirements shall identify all operations on the security requirements.

1758 **9.7.4.2.4 ASE\_REQ.2.4C**

1759 All operations shall be performed correctly.

1760 **9.7.4.2.5 ASE\_REQ.2.5C**

1761 Each dependency of the security requirements shall either be satisfied, or the security requirements  
 1762 rationale shall justify the dependency not being satisfied.

1763 **9.7.4.2.6 ASE\_REQ.2.6C**

1764 The security requirements rationale shall trace each SFR back to the SPD elements **for the TOE.**

1765 **9.7.4.2.7 ASE\_REQ.2.8C**

1766 **The security requirements rationale shall demonstrate that the SFRs meet all security**  
 1767 **objectives for the TOE.**

1768 **9.7.4.2.8 ASE\_REQ.2.9C**1769 **The security requirements rationale shall explain why the SARs were chosen.**1770 **9.7.4.2.9 ASE\_REQ.2.10C**

1771 The statement of security requirements shall be internally consistent.

1772 **9.7.4.2.10 ASE\_REQ.2.11C**1773 **9.7.4.3** The security requirements rationale shall demonstrate that the security objectives for the  
1774 operational environment uphold all assumptions.1775 **9.7.4.4 Evaluator action elements**1776 **9.7.4.4.1 ASE\_REQ.2.1E**1777 The evaluator shall confirm that the information provided meets all requirements for content and  
1778 presentation of evidence.1779 **9.8 TOE summary specification (ASE\_TSS)**1780 **9.8.1 Objectives**1781 The TOE summary specification enables evaluators and potential consumers to gain a general  
1782 understanding of how the TOE is implemented.1783 Evaluation of the TOE summary specification is necessary to determine whether it is adequately  
1784 described how the TOE:

1785 • meets its SFRs;

1786 • protects itself against interference, logical tampering and bypass;

1787 and whether the TOE summary specification is consistent with other narrative descriptions of the  
1788 TOE.1789 **9.8.2 Component levelling**1790 The components in this family are levelled on whether the TOE summary specification only needs to  
1791 describe how the TOE meets the SFRs, or whether the TOE summary specification also needs to  
1792 describe how the TOE protects itself against logical tampering and bypass. This additional description  
1793 may be used in special circumstances where there might be a specific concern regarding the TOE  
1794 security architecture.1795 **9.8.3 ASE\_TSS.1 TOE summary specification**

1796 Dependencies: ASE\_INT.1 ST introduction

1797 ASE\_REQ.1 Stated security requirements

1798 ADV\_FSP.1 Basic functional specification

1799 **9.8.3.1 Developer action elements**

1800 **9.8.3.1.1 ASE\_TSS.1.1D**

1801 **The developer shall provide a TOE summary specification.**

1802 **9.8.3.2 Content and presentation elements**

1803 **9.8.3.2.1 ASE\_TSS.1.1C**

1804 **The TOE summary specification shall describe how the TOE meets each SFR.**

1805 **9.8.3.3 Evaluator action elements**

1806 **9.8.3.3.1 ASE\_TSS.1.1E**

1807 **The evaluator shall confirm that the information provided meets all requirements for content**  
 1808 **and presentation of evidence.**

1809 **9.8.3.3.2 ASE\_TSS.1.2E**

1810 **The evaluator shall confirm that the TOE summary specification is consistent with the TOE**  
 1811 **overview and the TOE description.**

1812 **9.8.4 ASE\_TSS.2 TOE summary specification with architectural design summary**

1813 Dependencies: ASE\_INT.1 ST introduction

1814 ASE\_REQ.1 Stated security requirements

1815 ADV\_ARC.1 Security architecture description

1816 **9.8.4.1 Developer action elements**

1817 **9.8.4.1.1 ASE\_TSS.2.1D**

1818 **The developer shall provide a TOE summary specification.**

1819 **9.8.4.2 Content and presentation elements**

1820 **9.8.4.2.1 ASE\_TSS.2.1C**

1821 **The TOE summary specification shall describe how the TOE meets each SFR.**

1822 **9.8.4.2.2 ASE\_TSS.2.2C**

1823 **The TOE summary specification shall describe how the TOE protects itself against interference**  
 1824 **and logical tampering.**

1825 **9.8.4.2.3 ASE\_TSS.2.3C**

1826 **The TOE summary specification shall describe how the TOE protects itself against bypass.**

1827 **9.8.4.3 Evaluator action elements**1828 **9.8.4.3.1 ASE\_TSS.2.1E**

1829 The evaluator shall confirm that the information provided meets all requirements for content and  
1830 presentation of evidence.

1831 **9.8.4.3.2 ASE\_TSS.2.2E**

1832 The evaluator shall confirm that the TOE summary specification is consistent with the TOE overview  
1833 and the TOE description.

1834 **9.9 Consistency of composite product Security Target (ASE\_COMP)**1835 **9.9.1 Objectives**

1836 The aim of this activity is to determine whether the Security Target of the composite product<sup>2</sup> does not  
1837 contradict the Security Target of the underlying platform<sup>3</sup>.

1838 **9.9.2 ASE\_COMP.1 Consistency of Security Target**

1839 Dependencies: No dependencies

1840 **9.9.2.1 Application notes**

1841 These application notes aid the developer to create as well as the evaluator to analyse a composite  
1842 Security Target and describe a general methodology for it. For detailed information / guidance please  
1843 refer to the single work units below.

1844 In order to create a composite Security Target, the developer should perform the following steps:

1845 Step 1: The developer formulates a preliminary Security Target for the composite product (the  
1846 Composite-ST) using the standard code of practice. The Composite-ST can be formulated  
1847 independently of the Security Target of the underlying platform (Platform-ST) – at least as long as  
1848 there are no formal PP conformance claims.

1849 Step 2: The developer determines the overlap between Platform-ST and Composite-ST through  
1850 analysing and comparing their TOE Security Functionality (TSF)<sup>45</sup>:

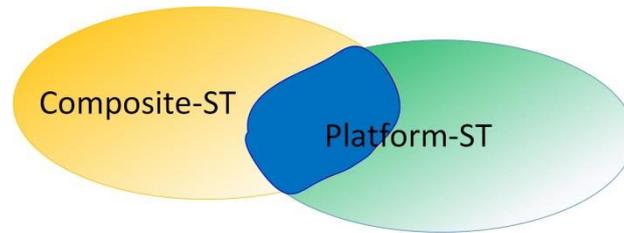
---

<sup>2</sup> denoted by Composite-ST in the following

<sup>3</sup> denoted by Platform-ST in the following. Generally, a Security Target expresses a security policy for the TOE defined.

<sup>4</sup> because the TSF enforce the Security Target (together with organisational measures enforcing security objectives for the operational environment of the TOE).

<sup>5</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.



1851

1852

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

1853 Step 3: The developer determines under which conditions he can trust in and rely on the Platform-  
1854 TSF being used by the Composite-ST without a new examination.

1855 Having undertaken these steps the developer completes the preliminary Security Target for the  
1856 composite product.

1857 It is not mandatory that the platform and the composite TOE are being certified according to same  
1858 edition of 15408. It is due to the fact that the application can rely on some security services of the  
1859 platform, if (i) the assurance level of the platform covers the intended assurance level of the composite  
1860 TOE and (ii) the platform's security certificate is valid and up-to-date. Equivalence of single assurance  
1861 components (and, hence, of assurance levels) belonging to different **ISO/IEC 15408 editions** shall be  
1862 established / acknowledged by the Composite Product Certification Body, cf. chapter Annex A.

1863 If a PP conformance is claimed (e.g. composite ST claim conformance to a PP that claims conformance  
1864 to a hardware PP), the consistency check can be reduced to the elements of the Security Target having  
1865 not already been covered by these Protection Profiles.

1866 The fact of compliance to a PP is not sufficient to avoid inconsistencies. Assume the following situation,  
1867 where  $\rightarrow$  stands for "complies with"

1868 Composite-ST  $\rightarrow$  SW PP  $\rightarrow$  HW PP  $\leftarrow$  platform-ST

1869 The SW PP may require any kind of conformance<sup>6</sup>, but this does not change the 'additional elements'  
1870 that the platform-ST may introduce to the HW PP. In conclusion, these additions are not necessarily  
1871 consistent with the composite-ST/SW PP additions: There is no scenario that ensures the consistency  
1872 'by construction'.

1873 Note that consistency may not be direct matching: e.g. objectives for the platform environment may  
1874 become objectives for the composite TOE.

## 1875 9.9.2.2 Developer action elements

### 1876 9.9.2.2.1 ASE\_COMP.1.1D

1877 **The developer shall provide a statement of compatibility between the Composite Security**  
1878 **Target and the Platform Security Target. This statement can be provided within the Composite**  
1879 **Product Security Target.**

---

<sup>6</sup> e.g. "strict", "exact" or "demonstrable" according to ISO/IEC 15408.

1880 **9.9.2.3 Content and presentation elements**1881 **9.9.2.3.1 ASE\_COMP.1.1C**

1882 **The statement of compatibility shall describe the separation of the Platform-TSF into relevant**  
 1883 **Platform-TSF being used by the Composite-ST and others.**

1884 **9.9.2.3.2 ASE\_COMP.1.2C**

1885 **The statement of compatibility between the Composite Security Target and the Platform**  
 1886 **Security Target shall show (e.g. in form of a mapping) that the Security Targets of the composite**  
 1887 **product and of the underlying platform match, i.e. that there is no conflict between security**  
 1888 **environments, security objectives, and security requirements of the Composite Security Target**  
 1889 **and the Platform Security Target. It can be provided by indicating of the concerned elements**  
 1890 **directly in the Security Target for the composite product followed by explanatory text, if**  
 1891 **necessary.**

1892 **9.9.2.4 Evaluator action elements**1893 **9.9.2.4.1 ASE\_COMP.1.1E**

1894 **The evaluator shall confirm that the information provided meets all requirements for content**  
 1895 **and presentation of evidence.**

1896 **10 Class ADV: Development**1897 **10.1 Introduction**

1898 The requirements of the Development class provide information about the TOE. The knowledge  
 1899 obtained by this information is used as the basis for conducting vulnerability analysis and testing upon  
 1900 the TOE, as described in the AVA and ATE classes.

1901 The Development class encompasses seven families of requirements for structuring and representing  
 1902 the TSF at various levels and varying forms of abstraction. These families include:

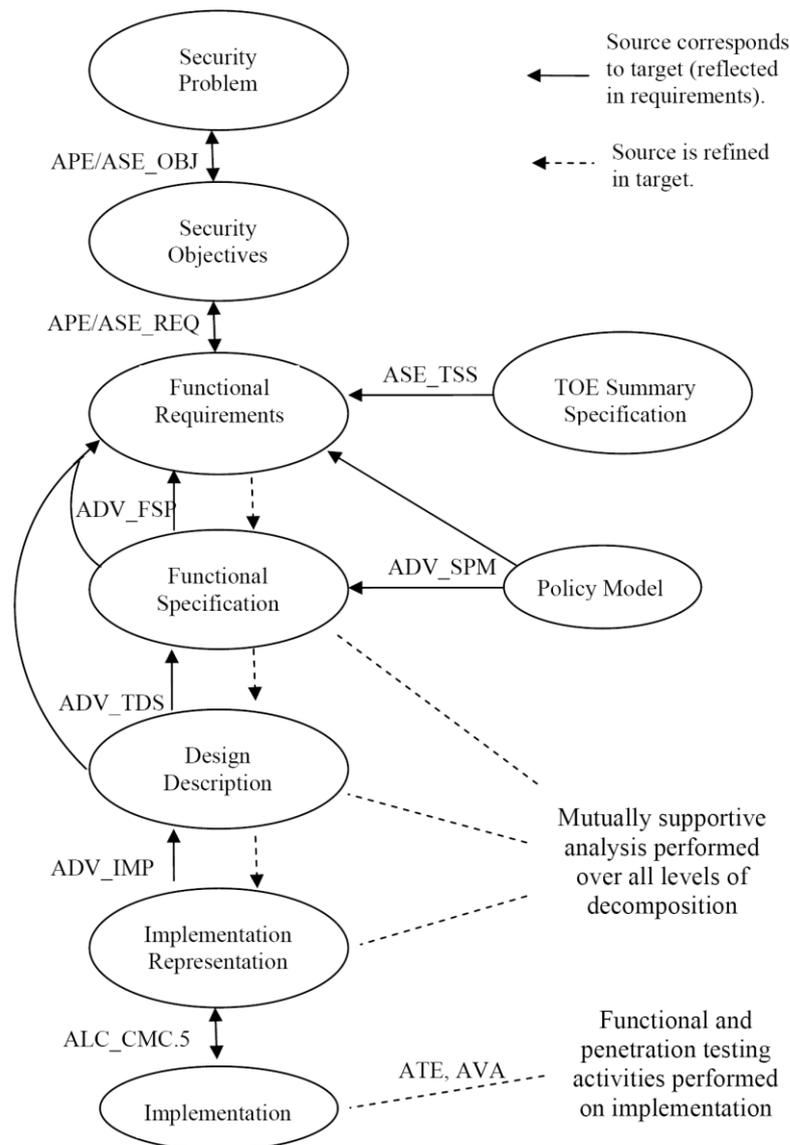
- 1903 • requirements for the description (at the various levels of abstraction) of the design and  
 1904 implementation of the SFRs (ADV\_FSP, ADV\_TDS, ADV\_IMP and ADV\_COMP)
- 1905 • requirements for the description of the architecture-oriented features of domain separation, TSF  
 1906 self-protection and non-bypassability of the security functionality (ADV\_ARC)
- 1907 • requirements for a security policy model and for correspondence mappings between security  
 1908 policy model and the functional specification (ADV\_SPM)
- 1909 • requirements on the internal structure of the TSF, which covers aspects such as modularity,  
 1910 layering, and minimisation of complexity (ADV\_INT)

1911 When documenting the security functionality of a TOE, there are two properties that need to be  
 1912 demonstrated. The first property is that the security functionality works correctly; that is, it performs  
 1913 as specified. The second property, and one that is arguably harder to demonstrate, is that the TOE  
 1914 cannot be used in a way such that the security functionality can be corrupted or bypassed. These two  
 1915 properties require somewhat different approaches in analysis, and so the families in ADV are  
 1916 structured to support these different approaches. The families Functional specification (ADV\_FSP),  
 1917 TOE design (ADV\_TDS), Implementation representation (ADV\_IMP), and Security policy modelling

1918 (ADV\_SPM) deal with the first property: the specification of the security functionality. The families  
 1919 Security Architecture (ADV\_ARC) and TSF internals (ADV\_INT) deal with the second property: the  
 1920 specification of the design of the TOE demonstrating the security functionality cannot be corrupted or  
 1921 bypassed. It should be noted that both properties need to be realised: the more confidence one has  
 1922 that the properties are satisfied, the more trustworthy the TOE is. The TSF of a composite product are  
 1923 represented at various levels of abstraction in the families of the development class ADV. The family  
 1924 Composite design compliance (ADV\_COMP) determines whether the requirements on the application,  
 1925 imposed by the underlying platform, are fulfilled in a composite product. Due to the distribution of the  
 1926 TSF of a composite product to various levels in the families of the class ADV, this family is not  
 1927 represented in Figure 8. The components in the families are designed so that more assurance can be  
 1928 gained as the components hierarchically increase.

1929 The paradigm for the families targeted at the first property is one of design decomposition. At the  
 1930 highest level, there is a functional specification of the TSF in terms of its interfaces (describing *what*  
 1931 the TSF does in terms of requests to the TSF for services and resulting responses), decomposing the  
 1932 TSF into smaller units (dependent on the assurance desired and the complexity of the TOE) and  
 1933 describing *how* the TSF accomplishes its functions (to a level of detail commensurate with the  
 1934 assurance level), and showing the implementation of the TSF. A formal model of the security  
 1935 behaviour also may be given. All levels of decomposition are used in determining the completeness  
 1936 and accuracy of all other levels, ensuring that the levels are mutually supportive. The requirements for  
 1937 the various TSF representations are separated into different families, to allow the PP/ST author to  
 1938 specify which TSF representations are required. The level chosen will dictate the assurance  
 1939 desired/gained.

1940 Figure 8 indicates the relationships among the various TSF representations of the ADV class, as well as  
 1941 their relationships with other classes. As the figure indicates, the APE and ASE classes define the  
 1942 requirements for the correspondence between the SFRs and the security objectives for the TOE. Class  
 1943 ASE also defines requirements for the correspondence between both the security objectives and SFRs,  
 1944 and for the TOE summary specification which explains how the TOE meets its SFRs. The activities of  
 1945 ALC\_CMC.5.2E include the verification that the TSF that is tested under the ATE and AVA classes is in  
 1946 fact the one described by all of the ADV decomposition levels.



1947

1948

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

1949 The requirements for all other correspondence shown in Figure 8 are defined in the ADV class for the  
 1950 TOE. The Security policy modelling (ADV\_SPM) family defines the requirements for formally modelling  
 1951 selected SFRs, and providing correspondence between the functional specification and the formal  
 1952 model. Each assurance family specific to a TSF representation (i.e., Functional specification (ADV\_FSP),  
 1953 TOE design (ADV\_TDS) and Implementation representation (ADV\_IMP)) defines requirements relating  
 1954 that TSF representation to the SFRs. All decompositions must accurately reflect all other  
 1955 decompositions (i.e., be mutually supportive); the developer supplies the tracings in the last .C  
 1956 elements of the components. Assurance relating to this factor is obtained during the analysis for each  
 1957 of the levels of decomposition by referring to other levels of decomposition (in a recursive fashion)  
 1958 while the analysis of a particular level of decomposition is being performed; the evaluator verifies the  
 1959 correspondence as part of the second E element. The understanding gained from these levels of  
 1960 decomposition form the basis of the functional and penetration testing efforts.

1961 The ADV\_INT family is not represented in this figure, as it is related to the internal structure of the TSF,  
 1962 and is only indirectly related to the process of refinement of the TSF representations. Similarly, the

- 1963 ADV\_ARC family is not represented in the figure because it relates to the architectural soundness,  
 1964 rather than representation, of the TSF. Both ADV\_INT and ADV\_ARC relate to the analysis of the  
 1965 property that the TOE cannot be made to circumvent or corrupt its security functionality.
- 1966 The TOE security functionality (TSF) consists of all parts of the TOE that have to be relied upon for  
 1967 enforcement of the SFRs. The TSF includes both functionality that directly enforces the SFRs, as well as  
 1968 functionality that, while not directly enforcing the SFRs, contributes to their enforcement in a more  
 1969 indirect manner, including functionality with the capability to cause the SFRs to be violated. This  
 1970 includes portions of the TOE that are invoked on start-up that are responsible for putting the TSF into  
 1971 its initial secure state.
- 1972 Several important concepts were used in the development of the components of the ADV families.  
 1973 These concepts, while introduced briefly here, are explained more fully in the application notes for the  
 1974 families.
- 1975 One over-riding notion is that, as more information becomes available, greater assurance can be  
 1976 obtained that the security functionality 1) is correctly implemented; 2) cannot be corrupted; and 3)  
 1977 cannot be bypassed. This is done through the verification that the documentation is correct and  
 1978 consistent with other documentation, and by providing information that can be used to ensure that the  
 1979 testing activities (both functional and penetration testing) are comprehensive. This is reflected in the  
 1980 levelling of the components of the families. In general, components are levelled based on the amount of  
 1981 information that is to be provided (and subsequently analysed).
- 1982 While not true for all TOEs, it is generally the case that the TSF is sufficiently complex that there are  
 1983 portions of the TSF that deserve more intense examination than other portions of the TSF.  
 1984 Determining those portions is unfortunately somewhat subjective, thus terminology and components  
 1985 have been defined such that as the level of assurance increases, the responsibility for determining  
 1986 what portions of the TSF need to be examined in detail shifts from the developer to the evaluator. To  
 1987 aid in expressing this concept, the following terminology is introduced. It should be noted that in the  
 1988 families of the class, this terminology is used when expressing SFR-related portions of the TOE (that is,  
 1989 elements and work units embodied in the Functional specification (ADV\_FSP), TOE design (ADV\_TDS),  
 1990 and Implementation representation (ADV\_IMP) families). While the general concept (that some  
 1991 portions of the TOE are more *interesting* than others) applies to other families, the criteria are  
 1992 expressed differently in order to obtain the assurance required.
- 1993 All portions of the TSF are *security relevant*, meaning that they must preserve the security of the TOE  
 1994 as expressed by the SFRs and requirements for domain separation and non-bypassability. One aspect  
 1995 of security relevance is the degree to which a portion of the TSF enforces a security requirement. Since  
 1996 different portions of the TOE play different roles (or no apparent role at all) in enforcing security  
 1997 requirements, this creates a continuum of SFR relevance: at one end of this continuum are portions of  
 1998 the TOE that are termed *SFR-enforcing*. Such portions play a direct role in implementing any SFR on  
 1999 the TOE. Such SFRs refer to any functionality provided by one of the SFRs contained in the ST. It should  
 2000 be noted that the definition of *plays a role in* for SFR-enforcing functionality is impossible to express  
 2001 quantitatively. For example, in the implementation of a Discretionary Access Control (DAC)  
 2002 mechanism, a very narrow view of *SFR-enforcing* might be the several lines of code that actually  
 2003 perform the check of a subject's attributes against the object's attributes. A broader view would  
 2004 include the software entity (e.g., C function) that contained the several lines of code. A broader view  
 2005 still would include callers of the C function, since they would be responsible for enforcing the decision  
 2006 returned by the attribute check. A still broader view would include any code in the call tree (or  
 2007 programming equivalent for the implementation language used) for that C function (e.g., a sort  
 2008 function that sorted access control list entries in a first-match algorithm implementation). At some  
 2009 point, the component is not so much *enforcing* the security policy but rather plays a *supporting* role;  
 2010 such components are termed *SFR supporting*. One of the characteristics of SFR-supporting

2011 functionality is that it is trusted to preserve the correctness of the SFR implementation by operating  
 2012 without error. Such functionality may be depended on by SFR-enforcing functionality, but the  
 2013 dependence is generally at a functional level; for example, memory management, buffer management,  
 2014 etc. Further down on the security relevance continuum is functionality termed *SFR non-interfering*.  
 2015 Such functionality has no role in implementing the SFRs, and is likely part of the TSF because of its  
 2016 environment; for example, any code running in a privileged hardware mode on an operating system. It  
 2017 needs to be considered part of the TSF because, if compromised (or replaced by malicious code), it  
 2018 could compromise the correct operation of an SFR by virtue of its operating in the privileged hardware  
 2019 mode. An example of SFR non-interfering functionality might be a set of mathematical floating point  
 2020 operations implemented in kernel mode for speed considerations.

2021 The architecture family (Security Architecture (ADV\_ARC)) provides for requirements and analysis of  
 2022 the TOE based on properties of domain separation, self-protection, and non-bypassability. These  
 2023 properties relate to the SFRs in that, if these properties are not present, it will likely lead to the failure  
 2024 of mechanisms implementing SFRs. Functionality and design relating to these properties *is not*  
 2025 considered a part of the continuum described above, but instead is treated separately due to its  
 2026 fundamentally different nature and analysis requirements.

2027 The difference in analysis of the implementation of SFRs (SFR-enforcing and SFR-supporting  
 2028 functionality) and the implementation of somewhat fundamental security properties of the TOE, which  
 2029 include the initialisation, self-protection, and non-bypassability concerns, is that the SFR-related  
 2030 functionality is more or less directly visible and relatively easy to test, while the above-mentioned  
 2031 properties require varying degrees of analysis on a much broader set of functionality. Further, the  
 2032 depth of analysis for such properties will vary depending on the design of the TOE. The ADV families  
 2033 are constructed to address this by a separate family (Security Architecture (ADV\_ARC)) devoted to  
 2034 analysis of the initialisation, self-protection, and non-bypassability requirements, while the other  
 2035 families are concerned with analysis of the functionality supporting SFRs.

2036 Even in cases where different descriptions are necessary for the multiple levels of abstraction, it is not  
 2037 absolutely necessary for each and every TSF representation to be in a separate document. Indeed, it  
 2038 may be the case that a single document meets the documentation requirements for more than one TSF  
 2039 representation, since it is the information about each of these TSF representations that is required,  
 2040 rather than the resulting document structure. In cases where multiple TSF representations are  
 2041 combined within a single document, the developer should indicate which portions of the documents  
 2042 meet which requirements.

2043 Three types of specification style are mandated by this class: informal, semiformal and formal. The  
 2044 functional specification and TOE design documentation are always written in either informal or  
 2045 semiformal style. A semiformal style reduces the ambiguity in these documents over an informal  
 2046 presentation. A formal specification may also be required *in addition to* the semi-formal presentation;  
 2047 the value is that a description of the TSF in more than one way will add increased assurance that the  
 2048 TSF has been completely and accurately specified.

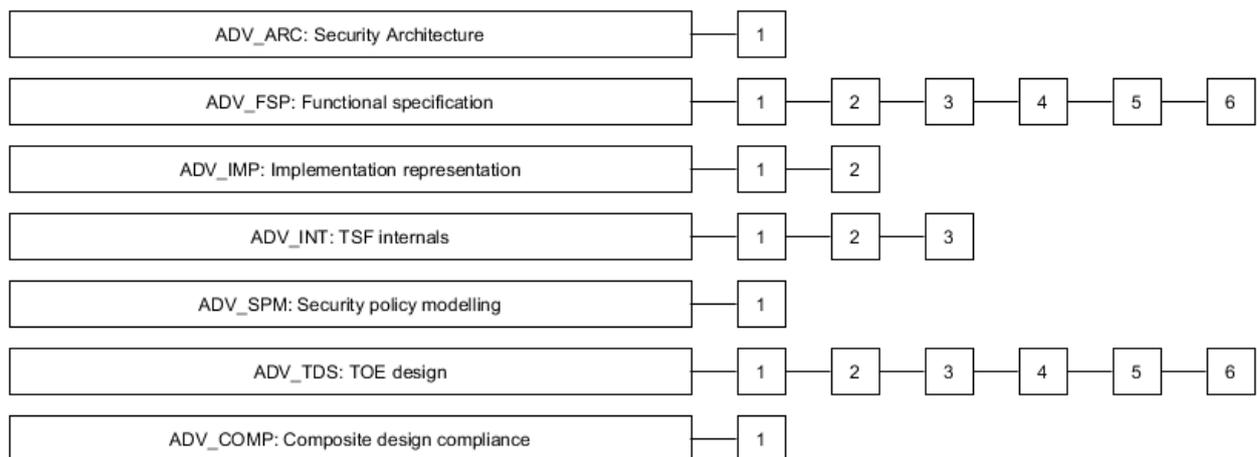
2049 An informal specification is written as prose in natural language. Natural language is used here as  
 2050 meaning communication in any commonly spoken tongue (e.g. Spanish, German, French, English,  
 2051 Dutch). An informal specification is not subject to any notational or special restrictions other than  
 2052 those required as ordinary conventions for that language (e.g. grammar and syntax). While no  
 2053 notational restrictions apply, the informal specification is also required to provide defined meanings  
 2054 for terms that are used in a context other than that accepted by normal usage.

2055 The difference between semiformal and informal documents is only a matter of formatting or  
 2056 presentation: a semiformal notation includes such things as an explicit glossary of terms, a  
 2057 standardised presentation format, etc. A semiformal specification is written to a standard presentation

2058 template. The presentation should use terms consistently if written in a natural language. The  
 2059 presentation may also use more structured languages/diagrams (e.g. data-flow diagrams, state  
 2060 transition diagrams, entity-relationship diagrams, data structure diagrams, and process or program  
 2061 structure diagrams). Whether based on diagrams or natural language, a set of conventions must be  
 2062 used in the presentation. The glossary explicitly identifies the words that are being used in a precise  
 2063 and constant manner; similarly, the standardised format implies that extreme care has been taken in  
 2064 methodically preparing the document in a manner that maximises clarity. It should be noted that  
 2065 fundamentally different portions of the TSF may have different semiformal notation conventions and  
 2066 presentation styles (as long as the number of different “semiformal notations” is small); this still  
 2067 conforms to the concept of a *semiformal presentation*.

2068 A formal specification is written in a notation based upon well-established mathematical concepts, and  
 2069 is typically accompanied by supporting explanatory (informal) prose. These mathematical concepts  
 2070 are used to define the syntax and semantics of the notation and the proof rules that support logical  
 2071 reasoning. The syntactic and semantic rules supporting a formal notation should define how to  
 2072 recognise constructs unambiguously and determine their meaning. There needs to be evidence that it  
 2073 is impossible to derive contradictions, and all rules supporting the notation need to be defined or  
 2074 referenced.

2075 Figure 9 shows the families within this class, and the hierarchy of components within the families.



2076

2077

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

2078 In case of a **multi-assurance evaluation** the requirements for the description (at the various levels of  
 2079 abstraction) of the design and implementation of the SFRs (ADV\_FSP, ADV\_TDS, ADV\_IMP and  
 2080 ADV\_COMP) will be presented for the **sub-TSF** of the TOE. The architecture family (Security  
 2081 Architecture (ADV\_ARC)) provides for requirements and analysis of the TOE based on properties of  
 2082 domain separation, self-protection, and non-bypassability which also may hold for boundaries  
 2083 between the **sub-TSF**.

## 2084 10.2 Security Architecture (ADV\_ARC)

### 2085 10.2.1 Objectives

2086 The objective of this family is for the developer to provide a description of the security architecture of  
 2087 the TSF. This will allow analysis of the information that, when coupled with the other evidence  
 2088 presented for the TSF, will confirm the TSF achieves the desired properties. The security architecture

2089 descriptions supports the implicit claim that security analysis of the TOE can be achieved by  
 2090 examining the TSF; without a sound architecture, the entire TOE functionality would have to be  
 2091 examined.

## 2092 **10.2.2 Component levelling**

2093 This family contains only one component.

## 2094 **10.2.3 Application notes**

2095 The properties of self-protection, domain separation, and non-bypassability are distinct from security  
 2096 functionality expressed by **ISO/IEC 15408-2** SFRs because self-protection and non-bypassability  
 2097 largely have no directly observable interface at the TSF. Rather, they are properties of the TSF that are  
 2098 achieved through the design of the TOE and TSF, and enforced by the correct implementation of that  
 2099 design.

2100 The approach used in this family is for the developer to design and provide a TSF that exhibits the  
 2101 above-mentioned properties, and to provide evidence (in the form of documentation) that explains  
 2102 these properties of the TSF. This explanation is provided at the same level of detail as the description  
 2103 of the SFR-enforcing elements of the TOE in the TOE design document. The evaluator has the  
 2104 responsibility for looking at the evidence and, coupled with other evidence delivered for the TOE and  
 2105 TSF, determining that the properties are achieved.

2106 Specification of security functionality implementing the SFRs (in the Functional specification  
 2107 (ADV\_FSP) and TOE design (ADV\_TDS)) will not necessarily describe mechanisms employed in  
 2108 implementing self-protection and non-bypassability (e.g. memory management mechanisms).  
 2109 Therefore, the material needed to provide the assurance that these requirements are being achieved is  
 2110 better suited to a presentation separate from the design decomposition of the TSF as embodied in  
 2111 ADV\_FSP and ADV\_TDS. This is not to imply that the security architecture description called for by this  
 2112 component cannot reference or make use of the design decomposition material; but it is likely that  
 2113 much of the detail present in the decomposition documentation will not be relevant to the argument  
 2114 being provided for the security architecture description document.

2115 The description of architectural soundness can be thought of as a developer's vulnerability analysis, in  
 2116 that it provides the justification for why the TSF is sound and enforces all of its SFRs. Where the  
 2117 soundness is achieved through specific security mechanisms, these will be tested as part of the Depth  
 2118 (ATE\_DPT) requirements; where the soundness is achieved solely through the architecture, the  
 2119 behaviour will be tested as part of the AVA: Vulnerability assessment requirements.

2120 This family consists of requirements for a security architecture description that describes the self-  
 2121 protection, domain separation, non-bypassability principles, including a description of how these  
 2122 principles are supported by the parts of the TOE that are used for TSF initialisation.

2123 **In case of a multi-assurance evaluation the properties of self-protection, domain separation, and**  
 2124 **non-bypassability may also be described for boundaries between the sub-TSF.**

2125 Additional information on the security architecture properties of self-protection, domain separation,  
 2126 and non-bypassability can be found in Annex A.1, ADV\_ARC: Supplementary material on security  
 2127 architectures.

## 2128 **10.2.4 ADV\_ARC.1 Security architecture description**

2129 Dependencies: ADV\_FSP.1 Basic functional specification

- 2130                   ADV\_TDS.1 Basic design
- 2131   **10.2.4.1 Developer action elements**
- 2132   **10.2.4.1.1 ADV\_ARC.1.1D**
- 2133   **The developer shall design and implement the TOE so that the security features of the TSF**  
2134   **cannot be bypassed.**
- 2135   **10.2.4.1.2 ADV\_ARC.1.2D**
- 2136   **The developer shall design and implement the TSF so that it is able to protect itself from**  
2137   **tampering by untrusted active entities.**
- 2138   **10.2.4.1.3 ADV\_ARC.1.3D**
- 2139   **The developer shall provide a security architecture description of the TSF.**
- 2140   **10.2.4.2 Content and presentation elements**
- 2141   **10.2.4.2.1 ADV\_ARC.1.1C**
- 2142   **The security architecture description shall be at a level of detail commensurate with the**  
2143   **description of the SFR-enforcing abstractions described in the TOE design document.**
- 2144   **10.2.4.2.2 ADV\_ARC.1.2C**
- 2145   **The security architecture description shall describe the security domains maintained by the**  
2146   **TSF consistently with the SFRs.**
- 2147   **10.2.4.2.3 ADV\_ARC.1.3C**
- 2148   **The security architecture description shall describe how the TSF initialisation process is**  
2149   **secure.**
- 2150   **10.2.4.2.4 ADV\_ARC.1.4C**
- 2151   **The security architecture description shall demonstrate that the TSF protects itself from**  
2152   **tampering.**
- 2153   **10.2.4.2.5 ADV\_ARC.1.5C**
- 2154   **The security architecture description shall demonstrate that the TSF prevents bypass of the**  
2155   **SFR-enforcing functionality.**
- 2156   **10.2.4.3 Evaluator action elements**
- 2157   **10.2.4.3.1 ADV\_ARC.1.1E**
- 2158   **The evaluator shall confirm that the information provided meets all requirements for content**  
2159   **and presentation of evidence.**

## 2160 10.3 Functional specification (ADV\_FSP)

### 2161 10.3.1 Objectives

2162 This family levies requirements upon the functional specification, which describes the TSF interfaces  
 2163 (TSFIs). The TSFI consists of all means by which external entities (or subjects in the TOE but outside of  
 2164 the TSF) supply data to the TSF, receive data from the TSF and invoke services from the TSF. It does  
 2165 *not* describe how the TSF processes those service requests, nor does it describe the communication  
 2166 when the TSF invokes services from its operational environment; this information is addressed by the  
 2167 TOE design (ADV\_TDS) and Reliance of dependent component (ACO\_REL) families, respectively.

2168 This family provides assurance directly by allowing the evaluator to understand how the TSF meets  
 2169 the claimed SFRs. It also provides assurance indirectly, as input to other assurance families and  
 2170 classes:

- 2171 • ADV\_ARC, where the description of the TSFIs may be used to gain better understanding of how the  
 2172 TSF is protected against corruption (i.e. subversion of self-protection or domain separation)  
 2173 and/or bypass;
- 2174 • ATE, where the description of the TSFIs is an important input for both developer and evaluator  
 2175 testing;
- 2176 • AVA, where the description of the TSFIs is used to search for vulnerabilities.

### 2177 10.3.2 Component levelling

2178 The components in this family are levelled on the degree of detail required of the description of the  
 2179 TSFIs, and the degree of formalism required of the description of the TSFIs.

### 2180 10.3.3 Application notes

2181 Once the TSFIs are determined (see **XXX** for guidance and examples of determining TSFI), they are  
 2182 described. At lower-level components, developers focus their documentation (and evaluators focus  
 2183 their analysis) on the more security-relevant aspects of the TOE. Three categories of TSFIs are defined,  
 2184 based upon the relevance the services available through them have to the SFRs being claimed:

- 2185 • If a service available through an interface can be traced to one of the SFRs levied on the TSF,  
 2186 then that interface is termed *SFR-enforcing*. Note that it is possible that an interface may have  
 2187 various services and results, some of which may be SFR-enforcing and some of which may not.
- 2188 • Interfaces to (or services available through an interface relating to) services that SFR-enforcing  
 2189 functionality depends upon, but need only to function correctly in order for the security  
 2190 policies of the TOE to be preserved, are termed *SFR-supporting*.
- 2191 • Interfaces to services on which SFR-enforcing functionality has no dependence are termed *SFR*  
 2192 *non-interfering*.

2193 It should be noted that in order for an interface to be SFR-supporting or SFR non-interfering it must  
 2194 have *no* SFR-enforcing services or results. In contrast, an SFR-enforcing interface may have SFR-  
 2195 supporting services (for example, the ability to set the system clock may be an SFR-enforcing service of  
 2196 an interface, but if that same interface is used to display the system date that service may be only SFR-  
 2197 supporting). An example of a purely SFR-supporting interface is a system call interface that is used  
 2198 both by users and by a portion of the TSF that is running on behalf of users.

2199 As more information about the TSFIs becomes available, the greater the assurance that can be gained  
 2200 that the interfaces are correctly categorised/analysed. The requirements are structured such that, at  
 2201 the lowest level, the information required for SFR non-interfering interfaces is the minimum necessary  
 2202 in order for the evaluator to make this determination in an effective manner. At higher levels, more  
 2203 information becomes available so that the evaluator has greater confidence in the designation.

2204 The purpose in defining these labels (SFR-enforcing, SFR-supporting, and SFR-non-interfering) and for  
 2205 levying different requirements upon each (at the lower assurance components) is to provide a first  
 2206 approximation of where to focus the analysis and the evidence upon which that analysis is performed.  
 2207 If the developer's documentation of the TSF interfaces describes all of the interfaces to the degree  
 2208 specified in the requirements for the SFR-enforcing interfaces (that is, if the documentation exceeds  
 2209 the requirements), there is no need for the developer to create new evidence to match the  
 2210 requirements. Similarly, because the labels are merely a means of differentiating the interface types  
 2211 within the requirements, there is no need for the developer to update the evidence solely to label the  
 2212 interfaces as SFR-enforcing, SFR-supporting, and SFR-non-interfering. The primary purpose of this  
 2213 labelling is to allow developers with less mature development methodologies (and associated  
 2214 artefacts, such as detailed interface and design documentation) to provide only the necessary evidence  
 2215 without undue cost.

2216 The last C element of each component within this family provides a direct correspondence between  
 2217 the SFRs and the functional specification; that is, an indication of which interfaces are used to invoke  
 2218 each of the claimed SFRs. In the cases where the ST contains such functional requirements as 15408-2,  
 2219 whose functionality may not manifest itself at the TSFIs, the functional specification and/or the tracing  
 2220 is expected to identify these SFRs; including them in the functional specification helps to ensure that  
 2221 they are not lost at lower levels of decomposition, where they will be relevant.

### 2222 **10.3.3.1 Detail about the Interfaces**

2223 The requirements define collections of details about TSFI to be provided. For the purposes of the  
 2224 requirements, interfaces are specified (in varying degrees of detail) in terms of their purpose, method  
 2225 of use, parameters, parameter descriptions, and error messages.

2226 The *purpose* of an interface is a high-level description of the general goal of the interface (e.g. process  
 2227 GUI commands, receive network packets, provide printer output, etc.).

2228 The interface's *method of use* describes how the interface is supposed to be used. This description  
 2229 should be built around the various interactions available at that interface. For instance, if the interface  
 2230 were a Unix command shell, *ls*, *mv* and *cp* would be interactions for that interface. For each interaction  
 2231 the method of use describes what the interaction does, both for behaviour seen at the interface (e.g.  
 2232 the programmer calling the API, the Windows users changing a setting in the registry, etc.) as well as  
 2233 behaviour at other interfaces (e.g. generating an audit record).

2234 *Parameters* are explicit inputs to and outputs from an interface that control the behaviour of that  
 2235 interface. For example, parameters are the arguments supplied to an API; the various fields in a packet  
 2236 for a given network protocol; the individual key values in the Windows Registry; the signals across a  
 2237 set of pins on a chip; the flags that can be set for the *ls*, etc. The parameters are "identified" with a  
 2238 simple list of what they are.

2239 A *parameter description* tells what the parameter is in some meaningful way. For instance, an  
 2240 acceptable parameter description for interface *foo(i)* would be "parameter *i* is an integer that indicates  
 2241 the number of users currently logged in to the system". A description such as "parameter *i* is an  
 2242 integer" is not an acceptable.

2243 The description of an interface's *actions* describes what the interface does. This is more detailed than  
 2244 the purpose in that, while the “purpose” reveals why one might want to use it, the “actions” reveals  
 2245 everything that it does. These actions might be related to the SFRs or not. In cases where the  
 2246 interface's action is not related to SFRs, its description is said to be *summarised*, meaning the  
 2247 description merely makes clear that it is indeed not SFR-related.

2248 The *error message description* identifies the condition that generated it, what the message is, and the  
 2249 meaning of any error codes. An error message is generated by the TSF to signify that a problem or  
 2250 irregularity of some degree has been encountered. The requirements in this family refer to different  
 2251 kinds of error messages:

- 2252 • a “direct” error message is a security-relevant response through a specific TSFI invocation.
- 2253 • an “indirect” error cannot be tied to a specific TSFI invocation because it results from system-wide  
 2254 conditions (e.g. resource exhaustion, connectivity interruptions, etc.). Error messages that are not  
 2255 security-relevant are also considered “indirect”.
- 2256 • “remaining” errors are any other errors, such as those that might be referenced within the code.  
 2257 For example, the use of condition-checking code that checks for conditions that would not logically  
 2258 occur (e.g. a final “else” after a list of “case” statements), would provide for generating a catch-all  
 2259 error message; in an operational TOE, these error messages should never be seen.

2260 An example functional specification is provided in A.2.4.

### 2261 10.3.3.2 Components of this Family

2262 Increasing assurance through increased completeness and accuracy in the interface specification is  
 2263 reflected in the documentation required from the developer as detailed in the various hierarchical  
 2264 components of this family.

2265 At ADV\_FSP.1 Basic functional specification, the only documentation required is a characterisation of  
 2266 all TSFIs and a high level description of SFR-enforcing and SFR-supporting TSFIs. To provide some  
 2267 assurance that the “important” aspects of the TSF have been correctly characterised at the TSFIs, the  
 2268 developer is required to provide the purpose and method of use, parameters for the SFR-enforcing and  
 2269 SFR-supporting TSFIs.

2270 At ADV\_FSP.2 Security-enforcing functional specification, the developer is required to provide the  
 2271 purpose, method of use, parameters, and parameter descriptions for all TSFIs. Additionally, for the  
 2272 SFR-enforcing TSFIs the developer has to describe the SFR-enforcing actions and direct error  
 2273 messages.

2274 At ADV\_FSP.3 Functional specification with complete summary, the developer must now, in addition to  
 2275 the information required at ADV\_FSP.2, provide enough information about the SFR-supporting and  
 2276 SFR-non-interfering actions to show that they are not SFR-enforcing. Further, the developer must now  
 2277 document all of the direct error messages resulting from the invocation of SFR-enforcing TSFIs.

2278 At ADV\_FSP.4 Complete functional specification, all TSFIs - whether SFR-enforcing, SFR-supporting or  
 2279 SFR-non-interfering - must be described to the same degree, including all of the direct error messages.

2280 At ADV\_FSP.5 Complete semi-formal functional specification with additional error information, the  
 2281 TSFIs descriptions also include error messages that do not result from an invocation of a TSFI.

2282 At ADV\_FSP.6 Complete semi-formal functional specification with additional formal specification, in  
 2283 addition to the information required by ADV\_FSP.5, all remaining error messages are included. The  
 2284 developer must also provide a formal description of the TSFI. This provides an alternative view of the  
 2285 TSFI that may expose inconsistencies or incomplete specification.

#### 2286 **10.3.4 ADV\_FSP.1 Basic functional specification**

2287 Dependencies: No dependencies.

##### 2288 **10.3.4.1 Developer action elements**

###### 2289 **10.3.4.1.1 ADV\_FSP.1.1D**

2290 **The developer shall provide a functional specification.**

###### 2291 **10.3.4.1.2 ADV\_FSP.1.2D**

2292 **The developer shall provide a tracing from the functional specification to the SFRs.**

##### 2293 **10.3.4.2 Content and presentation elements**

###### 2294 **10.3.4.2.1 ADV\_FSP.1.1C**

2295 **The functional specification shall describe the purpose and method of use for each SFR-**  
 2296 **enforcing and SFR-supporting TSFI.**

###### 2297 **10.3.4.2.2 ADV\_FSP.1.2C**

2298 **The functional specification shall identify all parameters associated with each SFR-enforcing**  
 2299 **and SFR-supporting TSFI.**

###### 2300 **10.3.4.2.3 ADV\_FSP.1.3C**

2301 **The functional specification shall provide rationale for the implicit categorisation of interfaces**  
 2302 **as SFR-non-interfering.**

###### 2303 **10.3.4.2.4 ADV\_FSP.1.4C**

2304 **The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.**

##### 2305 **10.3.4.3 Evaluator action elements**

###### 2306 **10.3.4.3.1 ADV\_FSP.1.1E**

2307 **The evaluator shall confirm that the information provided meets all requirements for content**  
 2308 **and presentation of evidence.**

###### 2309 **10.3.4.3.2 ADV\_FSP.1.2E**

2310 **The evaluator shall determine that the functional specification is an accurate and complete**  
 2311 **instantiation of the SFRs.**

#### 2312 **10.3.5 ADV\_FSP.2 Security-enforcing functional specification**

2313 Dependencies: ADV\_TDS.1 Basic design

- 2314 **10.3.5.1 Developer action elements**
- 2315 **10.3.5.1.1 ADV\_FSP.2.1D**
- 2316 The developer shall provide a functional specification.
- 2317 **10.3.5.1.2 ADV\_FSP.2.2D**
- 2318 The developer shall provide a tracing from the functional specification to the SFRs.
- 2319 **10.3.5.2 Content and presentation elements**
- 2320 **10.3.5.2.1 ADV\_FSP.2.1C**
- 2321 **The functional specification shall completely represent the TSF.**
- 2322 **10.3.5.2.2 ADV\_FSP.2.2C**
- 2323 The functional specification shall describe the purpose and method of use for **all** TSFI.
- 2324 **10.3.5.2.3 ADV\_FSP.2.3C**
- 2325 The functional specification shall identify **and describe** all parameters associated with each TSFI.
- 2326 **10.3.5.2.4 ADV\_FSP.2.4C**
- 2327 **For each SFR-enforcing TSFI, the functional specification shall describe the SFR-enforcing actions**  
 2328 **associated with the TSFI.**
- 2329 **10.3.5.2.5 ADV\_FSP.2.5C**
- 2330 **For each SFR-enforcing TSFI, the functional specification shall describe direct error messages**  
 2331 **resulting from processing associated with the SFR-enforcing actions.**
- 2332 **10.3.5.2.6 ADV\_FSP.2.6C**
- 2333 The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.
- 2334 **10.3.5.3 Evaluator action elements**
- 2335 **10.3.5.3.1 ADV\_FSP.2.1E**
- 2336 The evaluator shall confirm that the information provided meets all requirements for content and  
 2337 presentation of evidence.
- 2338 **10.3.5.3.2 ADV\_FSP.2.2E**
- 2339 The evaluator shall determine that the functional specification is an accurate and complete  
 2340 instantiation of the SFRs.
- 2341 **10.3.6 ADV\_FSP.3 Functional specification with complete summary**
- 2342 Dependencies: ADV\_TDS.1 Basic design

2343 **10.3.6.1 Developer action elements**

2344 **10.3.6.1.1 ADV\_FSP.3.1D**

2345 The developer shall provide a functional specification.

2346 **10.3.6.1.2 ADV\_FSP.3.2D**

2347 The developer shall provide a tracing from the functional specification to the SFRs.

2348 **10.3.6.2 Content and presentation elements**

2349 **10.3.6.2.1 ADV\_FSP.3.1C**

2350 The functional specification shall completely represent the TSF.

2351 **10.3.6.2.2 ADV\_FSP.3.2C**

2352 The functional specification shall describe the purpose and method of use for all TSFI.

2353 **10.3.6.2.3 ADV\_FSP.3.3C**

2354 The functional specification shall identify and describe all parameters associated with each TSFI.

2355 **10.3.6.2.4 ADV\_FSP.3.4C**

2356 For each SFR-enforcing TSFI, the functional specification shall describe the SFR-enforcing actions  
2357 associated with the TSFI.

2358 **10.3.6.2.5 ADV\_FSP.3.5C**

2359 For each SFR-enforcing TSFI, the functional specification shall describe direct error messages resulting  
2360 from **SFR-enforcing actions and exceptions** associated with **invocation of the TSFI**.

2361 **10.3.6.2.6 ADV\_FSP.3.6C**

2362 **The functional specification shall summarise the SFR-supporting and SFR-non-interfering**  
2363 **actions associated with each TSFI.**

2364 **10.3.6.2.7 ADV\_FSP.3.7C**

2365 The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

2366 **10.3.6.3 Evaluator action elements**

2367 **10.3.6.3.1 ADV\_FSP.3.1E**

2368 The evaluator shall confirm that the information provided meets all requirements for content and  
2369 presentation of evidence.

2370 **10.3.6.3.2 ADV\_FSP.3.2E**

2371 The evaluator shall determine that the functional specification is an accurate and complete  
2372 instantiation of the SFRs.

- 2373            **10.3.7 ADV\_FSP.4 Complete functional specification**
- 2374    Dependencies:    ADV\_TDS.1 Basic design
- 2375    **10.3.7.1 Developer action elements**
- 2376    **10.3.7.1.1 ADV\_FSP.4.1D**
- 2377    The developer shall provide a functional specification.
- 2378    **10.3.7.1.2 ADV\_FSP.4.2D**
- 2379    The developer shall provide a tracing from the functional specification to the SFRs.
- 2380    **10.3.7.2 Content and presentation elements**
- 2381    **10.3.7.2.1 ADV\_FSP.4.1C**
- 2382    The functional specification shall completely represent the TSF.
- 2383    **10.3.7.2.2 ADV\_FSP.4.2C**
- 2384    The functional specification shall describe the purpose and method of use for all TSFI.
- 2385    **10.3.7.2.3 ADV\_FSP.4.3C**
- 2386    The functional specification shall identify and describe all parameters associated with each TSFI.
- 2387    **10.3.7.2.4 ADV\_FSP.4.4C**
- 2388    The functional specification shall **describe all** actions associated with each TSFI.
- 2389    **10.3.7.2.5 ADV\_FSP.4.5C**
- 2390    **The functional specification shall describe all direct error messages that may result from an**  
 2391    **invocation of each TSFI.**
- 2392    **10.3.7.2.6 ADV\_FSP.4.6C**
- 2393    The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.
- 2394    **10.3.7.3 Evaluator action elements**
- 2395    **10.3.7.3.1 ADV\_FSP.4.1E**
- 2396    The evaluator shall confirm that the information provided meets all requirements for content and  
 2397    presentation of evidence.
- 2398    **10.3.7.3.2 ADV\_FSP.4.2E**
- 2399    The evaluator shall determine that the functional specification is an accurate and complete  
 2400    instantiation of the SFRs.

2401	<b>10.3.8 ADV_FSP.5 Complete semi-formal functional specification with additional error</b>
2402	<b>information</b>
2403	Dependencies: ADV_TDS.1 Basic design
2404	ADV_IMP.1 Implementation representation of the TSF
2405	<b>10.3.8.1 Developer action elements</b>
2406	<b>10.3.8.1.1 ADV_FSP.5.1D</b>
2407	The developer shall provide a functional specification.
2408	<b>10.3.8.1.2 ADV_FSP.5.2D</b>
2409	The developer shall provide a tracing from the functional specification to the SFRs.
2410	<b>10.3.8.2 Content and presentation elements</b>
2411	<b>10.3.8.2.1 ADV_FSP.5.1C</b>
2412	The functional specification shall completely represent the TSF.
2413	<b>10.3.8.2.2 ADV_FSP.5.2C</b>
2414	<b>The functional specification shall describe the TSFI using a semi-formal style.</b>
2415	<b>10.3.8.2.3 ADV_FSP.5.3C</b>
2416	The functional specification shall describe the purpose and method of use for all TSFI.
2417	<b>10.3.8.2.4 ADV_FSP.5.4C</b>
2418	The functional specification shall identify and describe all parameters associated with each TSFI.
2419	<b>10.3.8.2.5 ADV_FSP.5.5C</b>
2420	The functional specification shall describe all actions associated with each TSFI.
2421	<b>10.3.8.2.6 ADV_FSP.5.6C</b>
2422	The functional specification shall describe all direct error messages that may result from an invocation
2423	of each TSFI.
2424	<b>10.3.8.2.7 ADV_FSP.5.7C</b>
2425	<b>The functional specification shall describe all error messages that do not result from an</b>
2426	<b>invocation of a TSFI.</b>
2427	<b>10.3.8.2.8 ADV_FSP.5.8C</b>
2428	<b>The functional specification shall provide a rationale for each error message contained in the</b>
2429	<b>TSF implementation yet does not result from an invocation of a TSFI.</b>

- 2430 **10.3.8.2.9 ADV\_FSP.5.9C**
- 2431 The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.
- 2432 **10.3.8.3 Evaluator action elements**
- 2433 **10.3.8.3.1 ADV\_FSP.5.1E**
- 2434 The evaluator shall confirm that the information provided meets all requirements for content and  
2435 presentation of evidence.
- 2436 **10.3.8.3.2 ADV\_FSP.5.2E**
- 2437 The evaluator shall determine that the functional specification is an accurate and complete  
2438 instantiation of the SFRs.
- 2439 **10.3.9 ADV\_FSP.6 Complete semi-formal functional specification with additional formal**  
2440 **specification**
- 2441 Dependencies: ADV\_TDS.1 Basic design
- 2442 ADV\_IMP.1 Implementation representation of the TSF
- 2443 **10.3.9.1 Developer action elements**
- 2444 **10.3.9.1.1 ADV\_FSP.6.1D**
- 2445 The developer shall provide a functional specification.
- 2446 **10.3.9.1.2 ADV\_FSP.6.2D**
- 2447 **The developer shall provide a formal presentation of the functional specification of the TSF.**
- 2448 **10.3.9.1.3 ADV\_FSP.6.3D**
- 2449 The developer shall provide a tracing from the functional specification to the SFRs.
- 2450 **10.3.9.2 Content and presentation elements**
- 2451 **10.3.9.2.1 ADV\_FSP.6.1C**
- 2452 The functional specification shall completely represent the TSF.
- 2453 **10.3.9.2.2 ADV\_FSP.6.2C**
- 2454 The functional specification shall describe the TSFI using a **formal** style.
- 2455 **10.3.9.2.3 ADV\_FSP.6.3C**
- 2456 The functional specification shall describe the purpose and method of use for all TSFI.
- 2457 **10.3.9.2.4 ADV\_FSP.6.4C**
- 2458 The functional specification shall identify and describe all parameters associated with each TSFI.

2459 **10.3.9.2.5 ADV\_FSP.6.5C**

2460 The functional specification shall describe all actions associated with each TSFI.

2461 **10.3.9.2.6 ADV\_FSP.6.6C**

2462 The functional specification shall describe all direct error messages that may result from an invocation  
2463 of each TSFI.

2464 **10.3.9.2.7 ADV\_FSP.6.7C**

2465 The functional specification shall describe all error messages **contained in the TSF implementation**  
2466 **representation.**

2467 **10.3.9.2.8 ADV\_FSP.6.8C**

2468 The functional specification shall provide a rationale for each error message contained in the TSF  
2469 implementation **that is not otherwise described in the functional specification justifying why it is**  
2470 **not associated with** a TSFI.

2471 **10.3.9.2.9 ADV\_FSP.6.9C**

2472 **The formal presentation of the functional specification of the TSF shall describe the TSFI using**  
2473 **a formal style, supported by informal, explanatory text where appropriate.**

2474 **10.3.9.2.10 ADV\_FSP.6.10C**

2475 The tracing shall demonstrate that the SFRs trace to TSFIs in the functional specification.

2476 **10.3.9.3 Evaluator action elements**2477 **10.3.9.3.1 ADV\_FSP.6.1E**

2478 The evaluator shall confirm that the information provided meets all requirements for content and  
2479 presentation of evidence.

2480 **10.3.9.3.2 ADV\_FSP.6.2E**

2481 The evaluator shall determine that the functional specification is an accurate and complete  
2482 instantiation of the SFRs.

2483 **10.4 Implementation representation (ADV\_IMP)**2484 **10.4.1 Objectives**

2485 The function of the Implementation representation (ADV\_IMP) family is for the developer to make  
2486 available the implementation representation (and, at higher levels, the implementation itself) of the  
2487 TOE in a form that can be analysed by the evaluator. The implementation representation is used in  
2488 analysis activities for other families (analysing the TOE design, for instance) to demonstrate that the  
2489 TOE conforms its design and to provide a basis for analysis in other areas of the evaluation (e.g., the  
2490 search for vulnerabilities). The implementation representation is expected to be in a form that  
2491 captures the detailed internal workings of the TSF. This may be software source code, firmware source  
2492 code, hardware diagrams and/or IC hardware design language code or layout data.

2493 **10.4.2 Component levelling**

2494 The components in this family are levelled on the amount of implementation that is mapped to the  
2495 TOE design description.

2496 **10.4.3 Application notes**

2497 Source code or hardware diagrams and/or IC hardware design language code or layout data that are  
2498 used to build the actual hardware are examples of parts of an implementation representation. It is  
2499 important to note that while the implementation representation must be made available to the  
2500 evaluator, this does not imply that the evaluator needs to possess that representation. For instance,  
2501 the developer may require that the evaluator review the implementation representation at a site of the  
2502 developer's choosing.

2503 The entire implementation representation is made available to ensure that analysis activities are not  
2504 curtailed due to lack of information. This does not, however, imply that all of the representation is  
2505 examined when the analysis activities are being performed. This is likely impractical in almost all  
2506 cases, in addition to the fact that it most likely will not result in a higher-assurance TOE vs. targeted  
2507 sampling of the implementation representation. The implementation representation is made available  
2508 to allow analysis of other TOE design decompositions (e.g., functional specification, TOE design), and  
2509 to gain confidence that the security functionality described at a higher level in the design actually  
2510 appear to be implemented in the TOE. Conventions in some forms of the implementation  
2511 representation may make it difficult or impossible to determine from just the implementation  
2512 representation itself what the actual result of the compilation or run-time interpretation will be. For  
2513 example, compiler directives for C language compilers will cause the compiler to exclude or include  
2514 entire portions of the code. For this reason, it is important that such "extra" information or related  
2515 tools (scripts, compilers, etc.) be provided so that the implementation representation can be  
2516 accurately determined.

2517 The purpose of the mapping between the implementation representation and the TOE design  
2518 description is to aid the evaluator's analysis. The internal workings of the TOE may be better  
2519 understood when the TOE design is analysed with corresponding portions of the implementation  
2520 representation. The mapping serves as an index into the implementation representation. At the lower  
2521 component, only a subset of the implementation representation is mapped to the TOE design  
2522 description. Because of the uncertainty of which portions of the implementation representation will  
2523 need such a mapping, the developer may choose either to map the entire implementation  
2524 representation beforehand, or to wait to see which portions of the implementation representation the  
2525 evaluator requires to be mapped.

2526 The implementation representation is manipulated by the developer in a form that is suitable for  
2527 transformation to the actual implementation. For instance, the developer may work with files  
2528 containing source code, which is eventually compiled to become part of the TSF. The developer makes  
2529 available the implementation representation in the form used by the developer, so that the evaluator  
2530 may use automated techniques in the analysis. This also increases the confidence that the  
2531 implementation representation examined is actually the one used in the production of the TSF (as  
2532 opposed to the case where it is supplied in an alternate presentation format, such as a word processor  
2533 document). It should be noted that other forms of the implementation representation may also be  
2534 used by the developer; these forms are supplied as well. The overall goal is to supply the evaluator  
2535 with the information that will maximise the effectiveness of the evaluator's analysis efforts.

2536 Some forms of the implementation representation may require additional information because they  
2537 introduce significant barriers to understanding and analysis. Examples include "shrouded" source  
2538 code or source code that has been obfuscated in other ways such that it prevents understanding

2539 and/or analysis. These forms of implementation representation typically result from the TOE  
 2540 developer taking a version of the implementation representation and running a shrouding or  
 2541 obfuscation program on it. While the shrouded representation is what is compiled and may be closer  
 2542 to the implementation (in terms of structure) than the original, un-shrouded representation, supplying  
 2543 such obfuscated code may cause significantly more time to be spent in analysis tasks involving the  
 2544 representation. When such forms of representation are created, the components require details on the  
 2545 shrouding tools/algorithms used so that the un-shrouded representation can be supplied, and the  
 2546 additional information can be used to gain confidence that the shrouding process does not  
 2547 compromise any security functionality.

#### 2548 **10.4.4 ADV\_IMP.1 Implementation representation of the TSF**

2549 Dependencies: ADV\_TDS.3 Basic modular design

2550 ALC\_TAT.1 Well-defined development tools

##### 2551 **10.4.4.1 Developer action elements**

###### 2552 **10.4.4.1.1 ADV\_IMP.1.1D**

2553 **The developer shall make available the implementation representation for the entire TSF.**

###### 2554 **10.4.4.1.2 ADV\_IMP.1.2D**

2555 **The developer shall provide a mapping between the TOE design description and the sample of**  
 2556 **the implementation representation.**

##### 2557 **10.4.4.2 Content and presentation elements**

###### 2558 **10.4.4.2.1 ADV\_IMP.1.1C**

2559 **The implementation representation shall define the TSF to a level of detail such that the TSF**  
 2560 **can be generated without further design decisions.**

###### 2561 **10.4.4.2.2 ADV\_IMP.1.2C**

2562 **The implementation representation shall be in the form used by the development personnel.**

###### 2563 **10.4.4.2.3 ADV\_IMP.1.3C**

2564 **The mapping between the TOE design description and the sample of the implementation**  
 2565 **representation shall demonstrate their correspondence.**

##### 2566 **10.4.4.3 Evaluator action elements**

###### 2567 **10.4.4.3.1 ADV\_IMP.1.1E**

2568 **The evaluator shall confirm that, for the selected sample of the implementation representation,**  
 2569 **the information provided meets all requirements for content and presentation of evidence.**

#### 2570 **10.4.5 ADV\_IMP.2 Complete mapping of the implementation representation of the TSF**

2571 Dependencies: ADV\_TDS.3 Basic modular design

2572 ALC\_TAT.1 Well-defined development tools

- 2573 ALC\_CMC.5 Advanced support
- 2574 **10.4.5.1 Developer action elements**
- 2575 **10.4.5.1.1 ADV\_IMP.2.1D**
- 2576 The developer shall make available the implementation representation for the entire TSF.
- 2577 **10.4.5.1.2 ADV\_IMP.2.2D**
- 2578 The developer shall provide a mapping between the TOE design description and the **entire**  
2579 implementation representation.
- 2580 **10.4.5.2 Content and presentation elements**
- 2581 **10.4.5.2.1 ADV\_IMP.2.1C**
- 2582 The implementation representation shall define the TSF to a level of detail such that the TSF can be  
2583 generated without further design decisions.
- 2584 **10.4.5.2.2 ADV\_IMP.2.2C**
- 2585 The implementation representation shall be in the form used by the development personnel.
- 2586 **10.4.5.2.3 ADV\_IMP.2.3C**
- 2587 The mapping between the TOE design description and the **entire** implementation representation shall  
2588 demonstrate their correspondence.
- 2589 **10.4.5.3 Evaluator action elements**
- 2590 **10.4.5.3.1 ADV\_IMP.2.1E**
- 2591 The evaluator shall confirm that the information provided meets all requirements for content and  
2592 presentation of evidence.
- 2593 **10.5 TSF internals (ADV\_INT)**
- 2594 **10.5.1 Objectives**
- 2595 This family addresses the assessment of the internal structure of the TSF. A TSF whose internals are  
2596 well-structured is easier to implement and less likely to contain flaws that could lead to  
2597 vulnerabilities; it is also easier to maintain without the introduction of flaws.
- 2598 **10.5.2 Component levelling**
- 2599 The components in this family are levelled on the basis of the amount of structure and minimisation of  
2600 complexity required. ADV\_INT.1 Well-structured subset of TSF internals places requirements for well-  
2601 structured internals on only selected parts of the TSF. This component is not included in an EAL  
2602 because this component is viewed for use in special circumstances (e.g., the sponsor has a specific  
2603 concern regarding a cryptographic module, which is isolated from the rest of the TSF) and would not  
2604 be widely applicable.
- 2605 At the next level, the requirements for well-structured internals are placed on the entire TSF. Finally,  
2606 minimisation of complexity is introduced in the highest component.

2607 **10.5.3 Application notes**

2608 These requirements, when applied to the internal structure of the TSF, typically result in  
 2609 improvements that aid both the developer and the evaluator in understanding the TSF, and also  
 2610 provide the basis for designing and evaluating test suites. Further, improving understandability of the  
 2611 TSF should assist the developer in simplifying its maintainability.

2612 The requirements in this family are presented at a fairly abstract level. The wide variety of TOEs  
 2613 makes it impossible to codify anything more specific than “well-structured” or “minimum complexity”.  
 2614 Judgements on structure and complexity are expected to be derived from the specific technologies  
 2615 used in the TOE. For example, software is likely to be considered well-structured if it exhibits the  
 2616 characteristics cited in the software engineering disciplines. The components within this family call for  
 2617 identifying the standards for measuring the characteristic of being well-structured and not overly-  
 2618 complex.

2619 **10.5.4 ADV\_INT.1 Well-structured subset of TSF internals**

2620 Dependencies: ADV\_IMP.1 Implementation representation of the TSF

2621 ADV\_TDS.3 Basic modular design

2622 ALC\_TAT.1 Well-defined development tools

2623 **10.5.4.1 Objectives**

2624 The objective of this component is to provide a means for requiring specific portions of the TSF to be  
 2625 well-structured. The intent is that the entire TSF has been designed and implemented using sound  
 2626 engineering principles, but the analysis is performed upon only a specific subset.

2627 **10.5.4.2 Application notes**

2628 This component requires the PP or ST author to fill in an assignment with the subset of the TSF. This  
 2629 subset may be identified in terms of the internals of the TSF at any layer of abstraction. For example:

2630 a) the structural elements of the TSF as identified in the TOE design (e.g. “The developer shall design  
 2631 and implement *the audit subsystem* such that it has well-structured internals.”)

2632 b) the implementation (e.g. “The developer shall design and implement *the encrypt.c and decrypt.c*  
 2633 *files* such that it has well-structured internals.” or “The developer shall design and implement *the*  
 2634 *6227 IC chip* such that it has well-structured internals.”)

2635 It is likely this would not be readily accomplished by referencing the claimed SFRs (e.g. “The developer  
 2636 shall design and implement *the portion of the TSF that provide anonymity as defined in FPR\_ANO.2* such  
 2637 that it has well-structured internals.”) because this does not indicate where to focus the analysis.

2638 This component has limited value and would be suitable in cases where potentially-malicious  
 2639 users/subjects have limited or strictly controlled access to the TSFIs or where there is another means  
 2640 of protection (e.g., domain separation) that ensures the chosen subset of the TSF cannot be adversely  
 2641 affected by the rest of the TSF (e.g., the cryptographic functionality, which is isolated from the rest of  
 2642 the TSF, is well-structured).

2643 **10.5.4.3 Developer action elements**

2644 **10.5.4.3.1 ADV\_INT.1.1D**

2645 **The developer shall design and implement [assignment: *subset of the TSF*] such that it has well-**  
 2646 **structured internals.**

2647 **10.5.4.3.2 ADV\_INT.1.2D**

2648 **The developer shall provide an internals description and justification.**

2649 **10.5.4.4 Content and presentation elements**

2650 **10.5.4.4.1 ADV\_INT.1.1C**

2651 **The justification shall explain the characteristics used to judge the meaning of “well-**  
 2652 **structured”.**

2653 **10.5.4.4.2 ADV\_INT.1.2C**

2654 **The TSF internals description shall demonstrate that the assigned subset of the TSF is well-**  
 2655 **structured.**

2656 **10.5.4.5 Evaluator action elements**

2657 **10.5.4.5.1 ADV\_INT.1.1E**

2658 **The evaluator shall confirm that the information provided meets all requirements for content**  
 2659 **and presentation of evidence.**

2660 **10.5.4.5.2 ADV\_INT.1.2E**

2661 **The evaluator shall perform an internals analysis on the assigned subset of the TSF.**

2662 **10.5.5 ADV\_INT.2 Well-structured internals**

2663 Dependencies: ADV\_IMP.1 Implementation representation of the TSF

2664 ADV\_TDS.3 Basic modular design

2665 ALC\_TAT.1 Well-defined development tools

2666 **10.5.5.1 Objectives**

2667 The objective of this component is to provide a means for requiring the TSF to be well-structured. The  
 2668 intent is that the entire TSF has been designed and implemented using sound engineering principles.

2669 **10.5.5.2 Application notes**

2670 Judgements on the adequacy of the structure are expected to be derived from the specific technologies  
 2671 used in the TOE. This component calls for identifying the standards for measuring the characteristic of  
 2672 being well-structured.

2673 **10.5.5.3 Developer action elements**2674 **10.5.5.3.1 ADV\_INT.2.1D**

2675 The developer shall design and implement the **entire TSF** such that it has well-structured internals.

2676 **10.5.5.3.2 ADV\_INT.2.2D**

2677 The developer shall provide an internals description and justification.

2678 **10.5.5.4 Content and presentation elements**2679 **10.5.5.4.1 ADV\_INT.2.1C**

2680 The justification shall **describe** the characteristics used to judge the meaning of “well-structured”.

2681 **10.5.5.4.2 ADV\_INT.2.2C**

2682 The TSF internals description shall demonstrate that the **entire** TSF is well-structured.

2683 **10.5.5.5 Evaluator action elements**2684 **10.5.5.5.1 ADV\_INT.2.1E**

2685 The evaluator shall confirm that the information provided meets all requirements for content and  
2686 presentation of evidence.

2687 **10.5.5.5.2 ADV\_INT.2.2E**

2688 The evaluator shall perform an internals analysis on the TSF.

2689 **10.5.6 ADV\_INT.3 Minimally complex internals**

2690 Dependencies: ADV\_IMP.1 Implementation representation of the TSF

2691 ADV\_TDS.3 Basic modular design

2692 ALC\_TAT.1 Well-defined development tools

2693 **10.5.6.1 Objectives**

2694 The objective of this component is to provide a means for requiring the TSF to be well-structured and  
2695 of minimal complexity. The intent is that the entire TSF has been designed and implemented using  
2696 sound engineering principles.

2697 **10.5.6.2 Application notes**

2698 Judgements on the adequacy of the structure and complexity are expected to be derived from the  
2699 specific technologies used in the TOE. This component calls for identifying the standards for  
2700 measuring the structure and complexity.

2701 **10.5.6.3 Developer action elements**2702 **10.5.6.3.1 ADV\_INT.3.1D**

2703 The developer shall design and implement the entire TSF such that it has well-structured internals.

2704 **10.5.6.3.2 ADV\_INT.3.2D**

2705 The developer shall provide an internals description and justification.

2706 **10.5.6.4 Content and presentation elements**2707 **10.5.6.4.1 ADV\_INT.3.1C**

2708 The justification shall describe the characteristics used to judge the meaning of “well-structured” **and**  
2709 **“complex”**.

2710 **10.5.6.4.2 ADV\_INT.3.2C**

2711 The TSF internals description shall demonstrate that the entire TSF is well-structured **and is not**  
2712 **overly complex**.

2713 **10.5.6.5 Evaluator action elements**2714 **10.5.6.5.1 ADV\_INT.3.1E**

2715 The evaluator shall confirm that the information provided meets all requirements for content and  
2716 presentation of evidence.

2717 **10.5.6.5.2 ADV\_INT.3.2E**

2718 The evaluator shall perform an internals analysis on the **entire** TSF.

2719 **10.6 Security policy modelling (ADV\_SPM)**2720 **10.6.1 Objectives**

2721 It is the objective of this family to provide additional assurance from the development of a formal  
2722 *security policy model* of the TSFI behaviour of the TSF, and establishing a correspondence between the  
2723 functional specification and this security policy model. Preserving internal consistency the security  
2724 policy model is expected to formally establish the security principles from its characteristics by means  
2725 of a mathematical proof.

2726 **10.6.2 Component levelling**

2727 This family contains only one component.

2728 **10.6.3 Application notes**

2729 Inadequacies in a TOE can result either from a failure in understanding the security requirements or  
2730 from a flawed implementation of those security requirements. Defining the security requirements  
2731 adequately to ensure their understanding may be problematic because the definition must be  
2732 sufficiently precise to prevent undesired results or subtle flaws during implementation of the TOE.  
2733 Throughout the design, implementation, and review processes, the modelled security requirements  
2734 may and should be used as precise design and implementation guidance, thereby providing increased  
2735 assurance that the modelled security requirements modelled via the TSFI behaviour are satisfied by  
2736 the TOE. The precision of the model and resulting guidance is significantly improved by casting the  
2737 model in a formal language and verifying the security requirements by automated formal proof  
2738 techniques.

2739 The creation of a formal security policy model helps to identify and eliminate ambiguous, inconsistent,  
 2740 incomplete, contradictory, or unenforceable security policy elements. Once the TOE has been built, the  
 2741 formal model serves the evaluation effort by contributing to the evaluator's judgement of how well the  
 2742 developer has understood the security functionality being implemented and whether there are  
 2743 inconsistencies between the security requirements and the TOE design. The confidence in the model is  
 2744 accompanied by a correspondence analysis for model elements and the functional specification, and a  
 2745 proof that the model contains no inconsistencies.

2746 A formal security policy model is a precise formal presentation of the important aspects of security  
 2747 and their relationship to the behaviour of the TOE; it identifies the set of rules and practises that  
 2748 regulates how the TSF manages, protects, and otherwise controls the system resources. The model  
 2749 includes the set of restrictions and properties that specify how information and computing resources  
 2750 are prevented from being used to violate the SFRs, accompanied by a persuasive set of engineering  
 2751 arguments showing that these restrictions and properties play a key role in the enforcement of the  
 2752 SFRs. It consists both of the formalisms that express the security functionality, as well as ancillary text  
 2753 to explain the model and to provide it with context. The security behaviour of the TSF is modelled both  
 2754 in terms of external behaviour (i.e. how the TSF interacts with the rest of the TOE and with its  
 2755 operational environment), as well as its internal behaviour.

2756 The security policy model of the TOE is informally abstracted from its realisation by considering the  
 2757 TSFI behaviour defined in the functional specification, which is strongly connected to the SFRs and  
 2758 security policies expressed in the ST. The purpose of formal methods lies within the enhancement of  
 2759 the rigour of enforcement. Informal arguments are always prone to fallacies; especially if relationships  
 2760 among subjects, objects and operations get more and more involved. In order to minimise the risk of  
 2761 insecure state reachability the rules and characteristics of the security policy model are mapped to  
 2762 respective properties and features within some formal system, whose rigour and strength can  
 2763 afterwards be used to obtain the security properties by means of theorems and formal proof.

2764 While the term "formal security policy model" is used in academic circles, ISO/IEC 15408's approach  
 2765 has no fixed definition of "security"; it would equate to whatever SFRs are being claimed. Therefore,  
 2766 the formal security policy model is merely a formal representation of the set of SFRs being claimed by  
 2767 the TOE.

2768 The term *security policy* has traditionally been associated with only access control policies, whether  
 2769 label-based (mandatory access control) or user-based (discretionary access control). However, a  
 2770 security policy is not limited to access control; there are also audit policies, identification policies,  
 2771 authentication policies, encryption policies, management policies, and any other security policies that  
 2772 are enforced by the TOE, as described in the PP/ST.

#### 2773 **10.6.4 ADV\_SPM.1 Formal TOE security policy model**

2774 Dependencies: ADV\_FSP.5 Complete semi-formal functional specification with additional error  
 2775 information

2776 ADV\_FSP.6 Complete semi-formal functional specification with additional formal  
 2777 specification

##### 2778 **10.6.4.1 Developer action elements**

###### 2779 **10.6.4.1.1 ADV\_SPM.1.1D**

2780 **The developer shall provide a formal security policy model for the TSFI behaviour of the TOE.**

2781 **10.6.4.1.2 ADV\_SPM.1.2D**

2782 **The developer shall determine all TSFIs and analyze for each TSFI whether its behaviour can be**  
 2783 **modelled by the formal security policy model. If a TSFI cannot be modelled, for example caused**  
 2784 **by technical limitations, the developer shall analyze the impact of not modelling the TSFI**  
 2785 **behaviour on the security of the TOE. If parts of the TSFI behaviour cannot be modelled due to**  
 2786 **technical limitations, the remaining parts shall nevertheless be covered by the formal model.**

2787 **10.6.4.1.3 ADV\_SPM.1.3D**

2788 **The formal security policy model shall identify the modelled TSFIs. For each TSFI covered by**  
 2789 **the formal security policy model, the model shall identify the related SFRs and security policies**  
 2790 **in the ST. For each SFR covered by the formal security policy model, the model shall identify the**  
 2791 **relevant portions of the statement of SFRs.**

2792 **10.6.4.1.4 ADV\_SPM.1.4D**

2793 **For all TSFIs that are not modelled by the formal security policy model, the developer shall**  
 2794 **identify the affected SFRs and security policies in the ST.**

2795 **10.6.4.1.5 ADV\_SPM.1.5D**

2796 **The developer shall provide a formal proof of correspondence between the model and any**  
 2797 **formal functional specification. The proof of correspondence shall relate model elements and**  
 2798 **TSFIs. With ADV\_SPM.1.3D, the given proof of correspondence thereby implicitly provides a**  
 2799 **correspondence between model elements and SFRs, as well as model elements and security**  
 2800 **policies. The developer defines a structured process for identifying and presenting**  
 2801 **corresponding items formally.**

2802 **10.6.4.1.6 ADV\_SPM.1.6D**

2803 **The developer shall provide a demonstration of correspondence between the model and the**  
 2804 **functional specification. This item shall demonstrate the correspondence between model**  
 2805 **elements and TSFIs.**

2806 **10.6.4.2 Content and presentation elements**2807 **10.6.4.2.1 ADV\_SPM.1.1C**

2808 **The model shall define security for the TOE and provide a formal proof that the TOE cannot**  
 2809 **reach a state that is not secure.**

2810 **10.6.4.2.2 ADV\_SPM.1.2C**

2811 **The developer shall provide an analysis why the chosen modelling formalism is appropriate.**

2812 **10.6.4.2.3 ADV\_SPM.1.3C**

2813 **If tool support is used, the developer shall identify the tool chain used to verify the formal**  
 2814 **security policy model, including environments and version numbers. The developer shall**  
 2815 **provide arguments why the tool chain is suited and trustworthy.**

2816 **10.6.4.2.4 ADV\_SPM.1.4C**

2817 **The developer shall define how the formal analysis of the formal security policy model can be**  
 2818 **reproduced (for example, applying an interactive theorem prover to prove correctness of the**  
 2819 **formal security policy model).**

2820 **10.6.4.2.5 ADV\_SPM.1.5C**

2821 **The model shall be in a formal style, supported by explanatory text as required, and identify**  
 2822 **the TSFIs that are modelled. Additionally, the SFRs and security policies of the TSF that are**  
 2823 **modelled via the TSFI behaviour shall be presented. The model shall identify all TSFIs that are**  
 2824 **not modelled (compare ADV\_SPM.1.2D) and present the affected SFRs and security policies. The**  
 2825 **model shall explain the reason for not modelling TSFIs and provide an impact analysis which**  
 2826 **shows that correctness of the formal model is not affected.**

2827 **10.6.4.2.6 ADV\_SPM.1.6C**

2828 **The correspondence between the model and the functional specification shall be at the correct**  
 2829 **level of formality. The developer shall describe the correspondence analysis process and define**  
 2830 **the applied understanding of correspondence. If a semi-formal functional specification is**  
 2831 **provided, the correspondence must be shown semi-formally. If a formal functional**  
 2832 **specification is provided, the correspondence must be shown formally.**

2833 **10.6.4.2.7 ADV\_SPM.1.7C**

2834 **The correspondence shall show that the model is consistent and complete with respect to the**  
 2835 **functional specification.**

2836 **10.6.4.3 Evaluator action elements**2837 **10.6.4.3.1 ADV\_SPM.1.1E**

2838 **The evaluator shall confirm that the information provided meets all requirements for content**  
 2839 **and presentation of evidence.**

2840 **10.7 TOE design (ADV\_TDS)**2841 **10.7.1 Objectives**

2842 The design description of a TOE provides both context for a description of the TSF, and a thorough  
 2843 description of the TSF. As assurance needs increase, the level of detail provided in the description also  
 2844 increases. As the size and complexity of the TSF increase, multiple levels of decomposition are  
 2845 appropriate. The design requirements are intended to provide information (commensurate with the  
 2846 given assurance level) so that a determination can be made that the security functional requirements  
 2847 are realised.

2848 **10.7.2 Component levelling**

2849 The components in this family are levelled on the basis of the amount of information that is required  
 2850 to be presented with respect to the TSF, and on the degree of formalism required of the design  
 2851 description.

2852 **10.7.3 Application notes**

2853 The goal of design documentation is to provide sufficient information to determine the TSF boundary,  
 2854 and to describe *how* the TSF implements the Security Functional Requirements. The amount and  
 2855 structure of the design documentation will depend on the complexity of the TOE and the number of  
 2856 SFRs; in general, a very complex TOE with a large number of SFRs will require more design  
 2857 documentation than a very simple TOE implementing only a few SFRs. Very complex TOEs will benefit  
 2858 (in terms of the assurance provided) from the production of differing levels of decomposition in  
 2859 describing the design, while very simple TOEs do not require both high-level and low-level  
 2860 descriptions of its implementation.

2861 This family uses two levels of decomposition: the *subsystem* and the *module*. A module is the most  
 2862 specific description of functionality: it is a description of the implementation. A developer should be  
 2863 able to implement the part of the TOE described by the module with no further design decisions. A  
 2864 subsystem is a description of the design of the TOE; it helps to provide a high-level description of what  
 2865 a portion of the TOE is doing and how. As such, a subsystem may be further divided into lower-level  
 2866 subsystems, or into modules. Very complex TOEs might require several levels of subsystems in order  
 2867 to adequately convey a useful description of how the TOE works. Very simple TOEs, in contrast, might  
 2868 not require a subsystem level of description; the module might clearly describe how the TOE works.

2869 The general approach adopted for design documentation is that, as the level of assurance increases,  
 2870 the emphasis of description shifts from the general (subsystem level) to more (module level) detail. In  
 2871 cases where a module-level of abstraction is appropriate because the TOE is simple enough to be  
 2872 described at the module level, yet the level of assurance calls for a subsystem level of description, the  
 2873 module-level description alone will suffice. For complex TOEs, however, this is not the case: an  
 2874 enormous amount of (module-level) detail would be incomprehensible without an accompanying  
 2875 subsystem level of description.

2876 This approach follows the general paradigm that providing additional detail about the implementation  
 2877 of the TSF will result in greater assurance that the SFRs are implemented correctly, and provide  
 2878 information that can be used to demonstrate this in testing (ATE: Tests).

2879 In the requirements for this family, the term *interface* is used as the means of communication  
 2880 (between two subsystems or modules). It describes how the communication is invoked; this is similar  
 2881 to the details of TSFI (see Functional specification (ADV\_FSP)). The term *interaction* is used to identify  
 2882 the purpose for communication; it identifies why two subsystems or modules are communicating.

2883 **10.7.3.1 Detail about the Subsystems and Modules**

2884 The requirements define collections of details about subsystems and modules to be provided:

- 2885 a) The subsystems and modules are *identified* with a simple list of what they are.
- 2886 b) Subsystems and modules may be *categorised* (either implicitly or explicitly) as “SFR-enforcing”,  
 2887 “SFR-supporting”, or “SFR-non-interfering”; these terms are used the same as they are used in  
 2888 Functional specification (ADV\_FSP).
- 2889 c) A subsystem's *behaviour* is what it does. The behaviour may also be categorised as SFR-enforcing,  
 2890 SFR-supporting, or SFR-non-interfering. The behaviour of the subsystem is never categorised as  
 2891 more SFR-relevant than the category of the subsystem itself. For example, an SFR-enforcing  
 2892 subsystem can have SFR-enforcing behaviour as well as SFR-supporting or SFR-non-interfering  
 2893 behaviour.

- 2894 d) A *behaviour summary* of a subsystem is an overview of the actions it performs (e.g. “The TCP  
2895 subsystem assembles IP datagrams into reliable byte streams”).
- 2896 e) A *behaviour description* of a subsystem is an explanation of everything it does. This description  
2897 should be at a level of detail that one can readily determine whether the behaviour has any  
2898 relevance to the enforcement of the SFRs.
- 2899 f) A *description of interactions* among or between subsystems or modules identifies the reason that  
2900 subsystems or modules communicate, and characterises the information that is passed. It need not  
2901 define the information to the same level of detail as an interface specification. For example, it  
2902 would be sufficient to say “subsystem X requests a block of memory from the memory manager,  
2903 which responds with the location of the allocated memory.
- 2904 g) A *description of interfaces* provides the details of how the interactions among modules are  
2905 achieved. Rather than describing the reason the modules are communicating or the purpose of  
2906 their communication (that is, the description of interactions), the description of interfaces  
2907 describes the details of how that communication is accomplished, in terms of the structure and  
2908 contents of the messages, semaphores, internal process communications, etc.
- 2909 h) The *purpose* describes how a module provides their functionality. It provides sufficient detail that  
2910 no further design decisions are needed. The correspondence between the implementation  
2911 representation that implements the module, and the purpose of the module should be readily  
2912 apparent.
- 2913 i) A module is otherwise *described* in terms of whatever is identified in the element.

2914 Subsystems and modules, and “SFR-enforcing”, etc. are all further explained in greater detail in A.4,  
2915 ADV\_TDS: Subsystems and Modules.

#### 2916 **10.7.4 ADV\_TDS.1 Basic design**

2917 Dependencies: ADV\_FSP.2 Security-enforcing functional specification

##### 2918 **10.7.4.1 Developer action elements**

###### 2919 **10.7.4.1.1 ADV\_TDS.1.1D**

2920 **The developer shall provide the design of the TOE.**

###### 2921 **10.7.4.1.2 ADV\_TDS.1.2D**

2922 **The developer shall provide a mapping from the TSFI of the functional specification to the**  
2923 **lowest level of decomposition available in the TOE design.**

##### 2924 **10.7.4.2 Content and presentation elements**

###### 2925 **10.7.4.2.1 ADV\_TDS.1.1C**

2926 **The design shall describe the structure of the TOE in terms of subsystems.**

###### 2927 **10.7.4.2.2 ADV\_TDS.1.2C**

2928 **The design shall identify all subsystems of the TSF.**

2929 **10.7.4.2.3 ADV\_TDS.1.3C**

2930 **The design shall provide the behaviour summary of each SFR-supporting or SFR-non-**  
 2931 **interfering TSF subsystem.**

2932 **10.7.4.2.4 ADV\_TDS.1.4C**

2933 **The design shall summarise the SFR-enforcing behaviour of the SFR-enforcing subsystems.**

2934 **10.7.4.2.5 ADV\_TDS.1.5C**

2935 **The design shall provide a description of the interactions among SFR-enforcing subsystems of**  
 2936 **the TSF, and between the SFR-enforcing subsystems of the TSF and other subsystems of the TSF.**

2937 **10.7.4.2.6 ADV\_TDS.1.6C**

2938 **The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE**  
 2939 **design that they invoke.**

2940 **10.7.4.3 Evaluator action elements**2941 **10.7.4.3.1 ADV\_TDS.1.1E**

2942 **The evaluator shall confirm that the information provided meets all requirements for content**  
 2943 **and presentation of evidence.**

2944 **10.7.4.3.2 ADV\_TDS.1.2E**

2945 **The evaluator shall determine that the design is an accurate and complete instantiation of all**  
 2946 **security functional requirements.**

2947 **10.7.5 ADV\_TDS.2 Architectural design**

2948 Dependencies: ADV\_FSP.3 Functional specification with complete summary

2949 **10.7.5.1 Developer action elements**2950 **10.7.5.1.1 ADV\_TDS.2.1D**

2951 The developer shall provide the design of the TOE.

2952 **10.7.5.1.2 ADV\_TDS.2.2D**

2953 The developer shall provide a mapping from the TSFI of the functional specification to the lowest level  
 2954 of decomposition available in the TOE design.

2955 **10.7.5.2 Content and presentation elements**2956 **10.7.5.2.1 ADV\_TDS.2.1C**

2957 The design shall describe the structure of the TOE in terms of subsystems.

2958 **10.7.5.2.2 ADV\_TDS.2.2C**

2959 The design shall identify all subsystems of the TSF.

- 2960 **10.7.5.2.3 ADV\_TDS.2.3C**
- 2961 **The design shall provide the behaviour summary of each SFR non-interfering subsystem of the**  
2962 **TSF.**
- 2963 **10.7.5.2.4 ADV\_TDS.2.4C**
- 2964 The design shall **describe** the SFR-enforcing behaviour of the SFR-enforcing subsystems.
- 2965 **10.7.5.2.5 ADV\_TDS.2.5C**
- 2966 The design shall summarise the **SFR-supporting and SFR-non-interfering** behaviour of the SFR-  
2967 enforcing subsystems.
- 2968 **10.7.5.2.6 ADV\_TDS.2.6C**
- 2969 The design shall summarise the behaviour of the **SFR-supporting** subsystems.
- 2970 **10.7.5.2.7 ADV\_TDS.2.7C**
- 2971 **The design shall provide a description of the interactions among all subsystems of the TSF.**
- 2972 **10.7.5.2.8 ADV\_TDS.2.8C**
- 2973 The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE design that  
2974 they invoke.
- 2975 **10.7.5.3 Evaluator action elements**
- 2976 **10.7.5.3.1 ADV\_TDS.2.1E**
- 2977 The evaluator shall confirm that the information provided meets all requirements for content and  
2978 presentation of evidence.
- 2979 **10.7.5.3.2 ADV\_TDS.2.2E**
- 2980 The evaluator shall determine that the design is an accurate and complete instantiation of all security  
2981 functional requirements.
- 2982 **10.7.6 ADV\_TDS.3 Basic modular design**
- 2983 Dependencies: ADV\_FSP.4 Complete functional specification
- 2984 **10.7.6.1 Developer action elements**
- 2985 **10.7.6.1.1 ADV\_TDS.3.1D**
- 2986 The developer shall provide the design of the TOE.
- 2987 **10.7.6.1.2 ADV\_TDS.3.2D**
- 2988 The developer shall provide a mapping from the TSFI of the functional specification to the lowest level  
2989 of decomposition available in the TOE design.

- 2990 **10.7.6.2 Content and presentation elements**
- 2991 **10.7.6.2.1 ADV\_TDS.3.1C**
- 2992 The design shall describe the structure of the TOE in terms of subsystems.
- 2993 **10.7.6.2.2 ADV\_TDS.3.2C**
- 2994 **The design shall describe the TSF in terms of modules.**
- 2995 **10.7.6.2.3 ADV\_TDS.3.3C**
- 2996 The design shall identify all subsystems of the TSF.
- 2997 **10.7.6.2.4 ADV\_TDS.3.4C**
- 2998 The design shall **provide a description of each subsystem of the TSF.**
- 2999 **10.7.6.2.5 ADV\_TDS.3.5C**
- 3000 The design shall provide a description of the interactions among all subsystems of the TSF.
- 3001 **10.7.6.2.6 ADV\_TDS.3.6C**
- 3002 **The design shall provide a mapping from the subsystems of the TSF to the modules of the TSF.**
- 3003 **10.7.6.2.7 ADV\_TDS.3.7C**
- 3004 The design shall describe each **SFR-enforcing module in terms of its purpose and relationship**  
3005 **with other modules.**
- 3006 **10.7.6.2.8 ADV\_TDS.3.8C**
- 3007 **The design shall describe each SFR-enforcing module in terms of its SFR-related interfaces,**  
3008 **return values from those interfaces, interaction with other modules and called SFR-related**  
3009 **interfaces to other SFR-enforcing modules.**
- 3010 **10.7.6.2.9 ADV\_TDS.3.9C**
- 3011 The design shall describe each **SFR-supporting and SFR-non-interfering module in terms of its**  
3012 **purpose and interaction with other modules.**
- 3013 **10.7.6.2.10 ADV\_TDS.3.10C**
- 3014 The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE design that  
3015 they invoke.
- 3016 **10.7.6.3 Evaluator action elements**
- 3017 **10.7.6.3.1 ADV\_TDS.3.1E**
- 3018 The evaluator shall confirm that the information provided meets all requirements for content and  
3019 presentation of evidence.

3020 **10.7.6.3.2 ADV\_TDS.3.2E**

3021 The evaluator shall determine that the design is an accurate and complete instantiation of all security  
3022 functional requirements.

3023 **10.7.7 ADV\_TDS.4 Semiformal modular design**

3024 Dependencies: ADV\_FSP.5 Complete semi-formal functional specification with additional error  
3025 information

3026 **10.7.7.1 Developer action elements**3027 **10.7.7.1.1 ADV\_TDS.4.1D**

3028 The developer shall provide the design of the TOE.

3029 **10.7.7.1.2 ADV\_TDS.4.2D**

3030 The developer shall provide a mapping from the TSFI of the functional specification to the lowest level  
3031 of decomposition available in the TOE design.

3032 **10.7.7.2 Content and presentation elements**3033 **10.7.7.2.1 ADV\_TDS.4.1C**

3034 The design shall describe the structure of the TOE in terms of subsystems.

3035 **10.7.7.2.2 ADV\_TDS.4.2C**

3036 The design shall describe the TSF in terms of modules, **designating each module as SFR-enforcing,**  
3037 **SFR-supporting, or SFR-non-interfering.**

3038 **10.7.7.2.3 ADV\_TDS.4.3C**

3039 The design shall identify all subsystems of the TSF.

3040 **10.7.7.2.4 ADV\_TDS.4.4C**

3041 The design shall provide a **semiformal** description of each subsystem of the TSF, **supported by**  
3042 **informal, explanatory text where appropriate.**

3043 **10.7.7.2.5 ADV\_TDS.4.5C**

3044 The design shall provide a description of the interactions among all subsystems of the TSF.

3045 **10.7.7.2.6 ADV\_TDS.4.6C**

3046 The design shall provide a mapping from the subsystems of the TSF to the modules of the TSF.

3047 **10.7.7.2.7 ADV\_TDS.4.7C**

3048 The design shall describe each SFR-enforcing **and SFR-supporting** module in terms of its purpose and  
3049 relationship with other modules.

3050 **10.7.7.2.8 ADV\_TDS.4.8C**

3051 The design shall describe each SFR-enforcing **and SFR-supporting** module in terms of its SFR-related  
 3052 interfaces, return values from those interfaces, interaction with other modules and called SFR-related  
 3053 interfaces to other SFR-enforcing **or SFR-supporting** modules.

3054 **10.7.7.2.9 ADV\_TDS.4.9C**

3055 The design shall describe each SFR-non-interfering module in terms of its purpose and interaction  
 3056 with other modules.

3057 **10.7.7.2.10 ADV\_TDS.4.10C**

3058 The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE design that  
 3059 they invoke.

3060 **10.7.7.3 Evaluator action elements**

3061 **10.7.7.3.1 ADV\_TDS.4.1E**

3062 The evaluator shall confirm that the information provided meets all requirements for content and  
 3063 presentation of evidence.

3064 **10.7.7.3.2 ADV\_TDS.4.2E**

3065 The evaluator shall determine that the design is an accurate and complete instantiation of all security  
 3066 functional requirements.

3067 **10.7.8 ADV\_TDS.5 Complete semiformal modular design**

3068 Dependencies: ADV\_FSP.5 Complete semi-formal functional specification with additional error  
 3069 information

3070 **10.7.8.1 Developer action elements**

3071 **10.7.8.1.1 ADV\_TDS.5.1D**

3072 The developer shall provide the design of the TOE.

3073 **10.7.8.1.2 ADV\_TDS.5.2D**

3074 The developer shall provide a mapping from the TSFI of the functional specification to the lowest level  
 3075 of decomposition available in the TOE design.

3076 **10.7.8.2 Content and presentation elements**

3077 **10.7.8.2.1 ADV\_TDS.5.1C**

3078 The design shall describe the structure of the TOE in terms of subsystems.

3079 **10.7.8.2.2 ADV\_TDS.5.2C**

3080 The design shall describe the TSF in terms of modules, designating each module as SFR-enforcing, SFR-  
 3081 supporting, or SFR-non-interfering.

3082 **10.7.8.2.3 ADV\_TDS.5.3C**

3083 The design shall identify all subsystems of the TSF.

3084 **10.7.8.2.4 ADV\_TDS.5.4C**

3085 The design shall provide a semiformal description of each subsystem of the TSF, supported by  
3086 informal, explanatory text where appropriate.

3087 **10.7.8.2.5 ADV\_TDS.5.5C**

3088 The design shall provide a description of the interactions among all subsystems of the TSF.

3089 **10.7.8.2.6 ADV\_TDS.5.6C**

3090 The design shall provide a mapping from the subsystems of the TSF to the modules of the TSF.

3091 **10.7.8.2.7 ADV\_TDS.5.7C**

3092 The design shall **provide a semiformal description of** each module in terms of its **purpose,**  
3093 **interaction,** interfaces, return values from those interfaces, and called interfaces to other modules,  
3094 **supported by informal, explanatory text where appropriate.**

3095 **10.7.8.2.8 ADV\_TDS.5.8C**

3096 The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE design that  
3097 they invoke.

3098 **10.7.8.3 Evaluator action elements**3099 **10.7.8.3.1 ADV\_TDS.5.1E**

3100 The evaluator shall confirm that the information provided meets all requirements for content and  
3101 presentation of evidence.

3102 **10.7.8.3.2 ADV\_TDS.5.2E**

3103 The evaluator shall determine that the design is an accurate and complete instantiation of all security  
3104 functional requirements.

3105 **10.7.9 ADV\_TDS.6 Complete semiformal modular design with formal high-level design**  
3106 **presentation**

3107 Dependencies: ADV\_FSP.6 Complete semi-formal functional specification with additional formal  
3108 specification

3109 **10.7.9.1 Developer action elements**3110 **10.7.9.1.1 ADV\_TDS.6.1D**

3111 The developer shall provide the design of the TOE.

3112 **10.7.9.1.2 ADV\_TDS.6.2D**

3113 The developer shall provide a mapping from the TSFI of the functional specification to the lowest level  
3114 of decomposition available in the TOE design.

3115 **10.7.9.1.3 ADV\_TDS.6.3D**

3116 **The developer shall provide a formal specification of the TSF subsystems.**

- 3117 **10.7.9.1.4 ADV\_TDS.6.4D**
- 3118 **The developer shall provide a proof of correspondence between the formal specifications of the**  
 3119 **TSF subsystems and of the functional specification.**
- 3120 **10.7.9.2 Content and presentation elements**
- 3121 **10.7.9.2.1 ADV\_TDS.6.1C**
- 3122 The design shall describe the structure of the TOE in terms of subsystems.
- 3123 **10.7.9.2.2 ADV\_TDS.6.2C**
- 3124 The design shall describe the TSF in terms of modules, designating each module as SFR-enforcing, SFR-  
 3125 supporting, or SFR-non-interfering.
- 3126 **10.7.9.2.3 ADV\_TDS.6.3C**
- 3127 The design shall identify all subsystems of the TSF.
- 3128 **10.7.9.2.4 ADV\_TDS.6.4C**
- 3129 The design shall provide a semiformal description of each subsystem of the TSF, supported by  
 3130 informal, explanatory text where appropriate.
- 3131 **10.7.9.2.5 ADV\_TDS.6.5C**
- 3132 The design shall provide a description of the interactions among all subsystems of the TSF.
- 3133 **10.7.9.2.6 ADV\_TDS.6.6C**
- 3134 The design shall provide a mapping from the subsystems of the TSF to the modules of the TSF.
- 3135 **10.7.9.2.7 ADV\_TDS.6.7C**
- 3136 The design shall **describe** each module in **semiformal style in** terms of its purpose, interaction,  
 3137 interfaces, return values from those interfaces, and called interfaces to other modules, supported by  
 3138 informal, explanatory text where appropriate.
- 3139 **10.7.9.2.8 ADV\_TDS.6.8C**
- 3140 **The formal specification of the TSF subsystems shall describe the TSF using a formal style,**  
 3141 **supported by informal, explanatory text where appropriate.**
- 3142 **10.7.9.2.9 ADV\_TDS.6.9C**
- 3143 The mapping shall demonstrate that all TSFIs trace to the behaviour described in the TOE design that  
 3144 they invoke.
- 3145 **10.7.9.2.10 ADV\_TDS.6.10C**
- 3146 **The proof of correspondence between the formal specifications of the TSF subsystems and of**  
 3147 **the functional specification shall demonstrate that all behaviour described in the TOE design is**  
 3148 **a correct and complete refinement of the TSFI that invoked it.**

3149 **10.7.9.3 Evaluator action elements**3150 **10.7.9.3.1 ADV\_TDS.6.1E**

3151 The evaluator shall confirm that the information provided meets all requirements for content and  
3152 presentation of evidence.

3153 **10.7.9.3.2 ADV\_TDS.6.2E**

3154 The evaluator shall determine that the design is an accurate and complete instantiation of all security  
3155 functional requirements.

3156 **10.8 Composite design compliance (ADV\_COMP)**3157 **10.8.1 Objectives**

3158 The aim of this activity is to determine whether the requirements on the application, imposed by the  
3159 underlying platform, are fulfilled in the composite product.

3160 **10.8.2 Component levelling**

3161 This family contains only one component.

3162 **10.8.3 Application notes**

3163 The requirements on the application, imposed by the underlying platform, can be formulated in the  
3164 relevant certification report (e.g. in form of constraints and recommendations), user guidance and  
3165 ETR\_COMP (in form of observations and recommendations) for the platform. The developer of the  
3166 composite product shall regard each of these sources, if available (cf. Table D2, chapter Annex A), and  
3167 implement the composite product in such a way that the applicable requirements are fulfilled.

3168 The TSF of the composite product is represented at various levels of abstraction in the families of the  
3169 development class ADV. Experiential, the appropriate levels of design representation for examining,  
3170 whether the requirements of the platform are fulfilled by the composite product, are the TOE design  
3171 (ADV\_TDS), security architecture (ADV\_ARC) and the implementation (ADV\_IMP). In case, these design  
3172 representation levels are not available (e.g. due to the assurance package chosen is EAL1), the current  
3173 activity is not applicable (see the next paragraph for the reason).

3174 Due to the definition of the composite TOE (cf. Clause 13.3.2.2 in ISO/IEC 15408-1) the interface  
3175 between the underlying platform and the application is the internal one, hence, a functional  
3176 specification (ADV\_FSP) as representation level is not appropriate for analysing the design  
3177 compliance.

3178 Security architecture ADV\_ARC as assurance family is dedicated to ensure that integrative security  
3179 services like domain separation, self-protection and non-bypassability properly work. It is impossible  
3180 and not the sense of the composite evaluation to have an insight into the architectural internals of the  
3181 underlying platform (it is a matter of the platform evaluation). What the Composite Evaluator has to  
3182 do in the context of ADV\_ARC is

3183 i. to determine whether the application uses services of the underlying platform within its own  
3184 Composite-ST to provide domain separation, self-protection, non-bypassability and protected  
3185 start-up; if no, there is no further composite activities for ADV\_ARC; if yes, then

3186 ii. the evaluator has to determine, whether the application uses these platform-services in an  
 3187 appropriate/secure way (please refer to the platform user guidance, cf. item #3 in Table D1,  
 3188 chapter Annex A).

3189 Since consistency of the composite product security policy has already been considered in the context  
 3190 of the Security Target in the assurance family ASE\_COMP, there is no necessity to consider non-  
 3191 contradictoriness of the security policy model (ADV\_SPM) of the composite TOE and the security  
 3192 policy model of the underlying platform.

3193 **10.8.4 ADV\_COMP.1 Design compliance with the platform certification report, guidance**  
 3194 **and ETR\_COMP**

3195 Dependencies: No dependencies

3196 **10.8.4.1**

3197 **10.8.4.2 Developer action elements**

3198 **10.8.4.2.1 ADV\_COMP.1.1D**

3199 **The developer shall provide a design compliance justification; cf. item #6 as well as items #3,**  
 3200 **#4, #5 in Table D1, chapter Annex A.**

3201 **10.8.4.3 Content and presentation elements**

3202 **10.8.4.3.1 ADV\_COMP.1.1C**

3203 **The design compliance justification shall provide a rationale for design compliance – on an**  
 3204 **appropriate representation level – of how the requirements on the application, imposed by the**  
 3205 **underlying platform, are fulfilled in the composite product.**

3206 **10.8.4.4 Evaluator action elements**

3207 **10.8.4.4.1 ADV\_COMP.1.1E**

3208 **The evaluator shall confirm that the rationale for design compliance is complete, coherent, and**  
 3209 **internally consistent.**

3210 **11 Class AGD: Guidance documents**

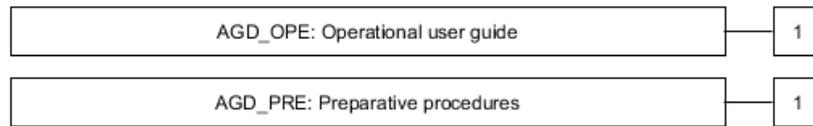
3211 **11.1 Introduction**

3212 The guidance documents class provides the requirements for guidance documentation for all user  
 3213 roles. For the secure preparation and operation of the TOE it is necessary to describe all relevant  
 3214 aspects for the secure handling of the TOE. The class also addresses the possibility of unintended  
 3215 incorrect configuration or handling of the TOE.

3216 In many cases it may be appropriate that guidance is provided in separate documents for preparation  
 3217 and operation of the TOE, or even separate for different user roles as end-users, administrators,  
 3218 application programmers using software or hardware interfaces, etc.

3219 The guidance documents class is subdivided into two families which are concerned with the  
 3220 preparative user guidance (what has to be done to transform the delivered TOE into its evaluated  
 3221 configuration in the operational environment as described in the ST) and with the operational user  
 3222 guidance (what has to be done during the operation of the TOE in its evaluated configuration).

3223 Figure 10 shows the families within this class, and the hierarchy of components within the families.



3224

3225 **Figure 10 — AGD: Guidance documents class decomposition**

## 3226 11.2 Operational user guidance (AGD\_OPE)

### 3227 11.2.1 Objectives

3228 Operational user guidance refers to written material that is intended to be used by all types of users of  
 3229 the TOE in its evaluated configuration: end-users, persons responsible for maintaining and  
 3230 administering the TOE in a correct manner for maximum security, and by others (e.g. programmers)  
 3231 using the TOE's external interfaces. Operational user guidance describes the security functionality  
 3232 provided by the TSF, provides instructions and guidelines (including warnings), helps to understand  
 3233 the TSF and includes the security-critical information, and the security-critical actions required, for its  
 3234 secure use. Misleading and unreasonable guidance should be absent from the guidance  
 3235 documentation, and secure procedures for all modes of operation should be addressed. Insecure states  
 3236 should be easy to detect.

3237 The operational user guidance provides a measure of confidence that non-malicious users,  
 3238 administrators, application providers and others exercising the external interfaces of the TOE will  
 3239 understand the secure operation of the TOE and will use it as intended. The evaluation of the user  
 3240 guidance includes investigating whether the TOE can be used in a manner that is insecure but that the  
 3241 user of the TOE would reasonably believe to be secure. The objective is to minimise the risk of human  
 3242 or other errors in operation that may deactivate, disable, or fail to activate security functionality,  
 3243 resulting in an undetected insecure state.

### 3244 11.2.2 Component levelling

3245 This family contains only one component.

### 3246 11.2.3 Application notes

3247 There may be different user roles or groups that are recognised by the TOE and that can interact with  
 3248 the TSF. These user roles and groups should be taken into consideration by the operational user  
 3249 guidance. They may be roughly grouped into administrators and non-administrative users, or more  
 3250 specifically grouped into persons responsible for receiving, accepting, installing and maintaining the  
 3251 TOE, application programmers, revisors, auditors, daily-management, end-users. Each role can  
 3252 encompass an extensive set of capabilities, or can be a single one.

3253 The requirement AGD\_OPE.1.1C encompasses the aspect that any warnings to the users during  
 3254 operation of a TOE with regard to the security problem definition and the security objectives for the  
 3255 operational environment described in the PP/ST are appropriately covered in the user guidance.

3256 The concept of secure values, as employed in AGD\_OPE.1.3C, has relevance where a user has control  
 3257 over security parameters. Guidance needs to be provided on secure and insecure settings for such  
 3258 parameters.

3259 AGD\_OPE.1.4C requires that the user guidance describes the appropriate reactions to all security-  
 3260 relevant events. Although many security-relevant events are the result of performing functions, this  
 3261 need not always be the case (e.g. the audit log fills up, an intrusion is detected). Furthermore, a  
 3262 security-relevant event may happen as a result of a specific chain of functions or, conversely, several  
 3263 security-relevant events may be triggered by one function.

3264 AGD\_OPE.1.7C requires that the user guidance is clear and reasonable. Misleading or unreasonable  
 3265 guidance may result in a user of the TOE believing that the TOE is secure when it is not.

3266 An example of misleading guidance would be the description of a single guidance instruction that  
 3267 could be parsed in more than one way, one of which may result in an insecure state.

3268 An example of unreasonable guidance would be a recommendation to follow a procedure that is so  
 3269 complicated that it cannot reasonably be expected that users will follow this guidance.

#### 3270 **11.2.4 AGD\_OPE.1 Operational user guidance**

3271 Dependencies: ADV\_FSP.1 Basic functional specification

##### 3272 **11.2.4.1 Developer action elements**

###### 3273 **11.2.4.1.1 AGD\_OPE.1.1D**

3274 **The developer shall provide operational user guidance.**

##### 3275 **11.2.4.2 Content and presentation elements**

###### 3276 **11.2.4.2.1 AGD\_OPE.1.1C**

3277 **The operational user guidance shall describe, for each user role, the user-accessible functions**  
 3278 **and privileges that should be controlled in a secure processing environment, including**  
 3279 **appropriate warnings.**

###### 3280 **11.2.4.2.2 AGD\_OPE.1.2C**

3281 **The operational user guidance shall describe, for each user role, how to use the available**  
 3282 **interfaces provided by the TOE in a secure manner.**

###### 3283 **11.2.4.2.3 AGD\_OPE.1.3C**

3284 **The operational user guidance shall describe, for each user role, the available functions and**  
 3285 **interfaces, in particular all security parameters under the control of the user, indicating secure**  
 3286 **values as appropriate.**

###### 3287 **11.2.4.2.4 AGD\_OPE.1.4C**

3288 **The operational user guidance shall, for each user role, clearly present each type of security-**  
 3289 **relevant event relative to the user-accessible functions that need to be performed, including**  
 3290 **changing the security characteristics of entities under the control of the TSF.**

###### 3291 **11.2.4.2.5 AGD\_OPE.1.5C**

3292 **The operational user guidance shall identify all possible modes of operation of the TOE**  
 3293 **(including operation following failure or operational error), their consequences and**  
 3294 **implications for maintaining secure operation.**

3295 **11.2.4.2.6 AGD\_OPE.1.6C**

3296 **The operational user guidance shall, for each user role, describe the security controls to be**  
 3297 **followed in order to fulfil the security objectives for the operational environment as described**  
 3298 **in the ST.**

3299 **11.2.4.2.7 AGD\_OPE.1.7C**

3300 **The operational user guidance shall be clear and reasonable.**

3301 **11.2.4.3 Evaluator action elements**3302 **11.2.4.3.1 AGD\_OPE.1.1E**

3303 **The evaluator shall confirm that the information provided meets all requirements for content**  
 3304 **and presentation of evidence.**

3305 **11.3 Preparative procedures (AGD\_PRE)**3306 **11.3.1 Objectives**

3307 Preparative procedures are useful for ensuring that the TOE has been received and installed in a  
 3308 secure manner as intended by the developer. The requirements for preparation call for a secure  
 3309 transition from the delivered TOE to its initial operational environment. This includes investigating  
 3310 whether the TOE can be configured or installed in a manner that is insecure but that the user of the  
 3311 TOE would reasonably believe to be secure.

3312 **11.3.2 Component levelling**

3313 This family contains only one component.

3314 **11.3.3 Application notes**

3315 It is recognised that the application of these requirements will vary depending on aspects such as  
 3316 whether the TOE is delivered in an operational state, or whether it has to be installed at the TOE  
 3317 owner's site, etc.

3318 The first process covered by the preparative procedures is the consumer's secure acceptance of the  
 3319 received TOE in accordance with the developer's delivery procedures. If the developer has not defined  
 3320 delivery procedures, security of the acceptance has to be ensured otherwise.

3321 Installation of the TOE includes transforming its operational environment into a state that conforms to  
 3322 the security objectives for the operational environment provided in the ST.

3323 It might also be the case that no installation is necessary, for example a smart card. In this case it may  
 3324 be inappropriate to require and analyse installation procedures.

3325 The requirements in this assurance family are presented separately from those in the Operational user  
 3326 guidance (AGD\_OPE) family, due to the infrequent, possibly one-time use of the preparative  
 3327 procedures.

3328 **11.3.4 AGD\_PRE.1 Preparative procedures**

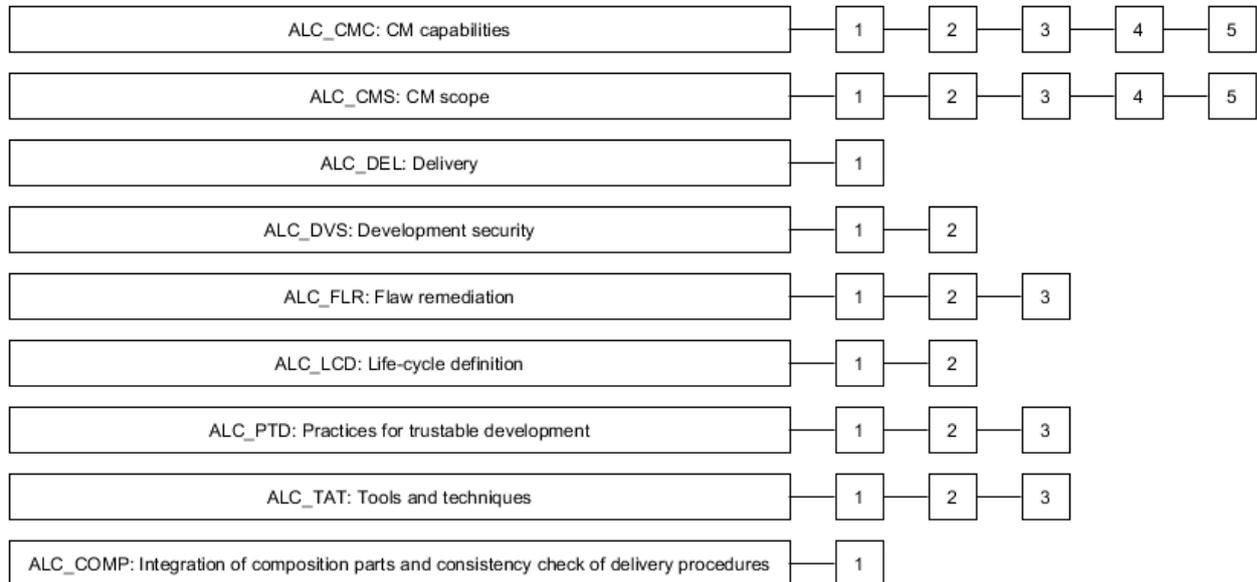
3329 Dependencies: No dependencies.

3330 **11.3.4.1 Developer action elements**3331 **11.3.4.1.1 AGD\_PRE.1.1D**3332 **The developer shall provide the TOE including its preparative procedures.**3333 **11.3.4.2 Content and presentation elements**3334 **11.3.4.2.1 AGD\_PRE.1.1C**3335 **The preparative procedures shall describe all the steps necessary for secure acceptance of the**  
3336 **delivered TOE in accordance with the developer's delivery procedures.**3337 **11.3.4.2.2 AGD\_PRE.1.2C**3338 **The preparative procedures shall describe all the steps necessary for secure installation of the**  
3339 **TOE and for the secure preparation of the operational environment in accordance with the**  
3340 **security objectives for the operational environment as described in the ST.**3341 **11.3.4.3 Evaluator action elements**3342 **11.3.4.3.1 AGD\_PRE.1.1E**3343 **The evaluator shall confirm that the information provided meets all requirements for content**  
3344 **and presentation of evidence.**3345 **11.3.4.3.2 AGD\_PRE.1.2E**3346 **The evaluator shall apply the preparative procedures to confirm that the TOE can be prepared**  
3347 **securely for operation.**3348 **12 Class ALC: Life-cycle support**3349 **12.1 Introduction**3350 Life-cycle support is an aspect of establishing appropriate security controls in the development,  
3351 production, delivery and maintenance of the TOE. Confidence in the correspondence between the TOE  
3352 security requirements and the TOE is greater if security analysis and the production of the evidence  
3353 are done on a regular basis as an integral part of the development, production, delivery and  
3354 maintenance activities.3355 During the life-cycle of the TOE it is distinguished whether the TOE is under the responsibility of the  
3356 TOE developer or the user rather than whether it is located in the development or the user  
3357 environment. The point of transition is when the TOE is accepted by the user. User in this context  
3358 relates to the end-user as well as product- and system integrators.

3359 The ALC class consists of nine families:

- 3360 • Development Life-cycle definition (ALC\_LCD) provides requirements for the developer's  
3361 description of the life-cycle model used in the development, production, delivery and maintenance  
3362 life-cycle of the TOE;
- 3363 • CM capabilities (ALC\_CMC) provides requirements for the management of the configuration items;

- 3364 • CM scope (ALC\_CMS) requires a minimum set of configuration items to be managed in the defined  
3365 way;
- 3366 • Developer environment security (ALC\_DVS) is concerned with the developer's physical, logical,  
3367 procedural, personnel, and other security controls;
- 3368 • Tools and techniques (ALC\_TAT) provides requirements for the development tools and  
3369 implementation standards used by the developer;
- 3370 • Flaw remediation (ALC\_FLR) provides requirements for the handling of security flaws.
- 3371 • Delivery (ALC\_DEL) provides requirements for the procedures used for the delivery of the TOE to  
3372 the downstream user. Delivery processes occurring during the development of the TOE are  
3373 denoted rather as transfers, and are handled in the context of integration and acceptance  
3374 procedures in other families of this class.
- 3375 • ALC\_TDA
- 3376 • ALC\_COMP
- 3377 Throughout this class, development and related terms (developer, develop) are meant in the more  
3378 general sense to comprise development and production, whereas production specifically means the  
3379 process of transforming the implementation representation into the final TOE.
- 3380 Figure 11 shows the families within this class, and the hierarchy of components within the families.



3381

3382 **Figure 11 — ALC: Life-cycle support class decomposition**

3383 **12.2 CM capabilities (ALC\_CMC)**

3384 **12.2.1 Objectives**

3385 Configuration management (CM) techniques, properly defined as part of the development life-cycle  
3386 model, contribute to the assurance argument that the TOE meets the SFRs. A Configuration

3387 Management (CM) system that is managed and operated correctly will help ensure the integrity of the  
 3388 portions of the TOE that are controlled, by providing a method of tracking any changes to the TOE, and  
 3389 to help ensure that all changes to the TOE are authorised.

3390 The objective of this family is to require the TOE developer's CM system to have certain capabilities.  
 3391 These capabilities are intended to reduce the likelihood that accidental or unauthorised modifications  
 3392 of the configuration items will occur. The CM system should support maintaining the integrity of the  
 3393 TOE throughout the part of the TOE's life-cycle that is under the control of the developer.

3394 The objective of introducing automated CM tools is to increase the effectiveness of the CM system.  
 3395 While both automated and manual CM systems can be bypassed, ignored, or proven insufficient to  
 3396 prevent unauthorised modification, automated systems are less susceptible to human error or  
 3397 negligence.

3398 The objectives of this family include the following:

- 3399 a) ensuring that the TOE is identifiable and complete before it is sent to the downstream user;
- 3400 b) ensuring that no configuration items are missed during evaluation;
- 3401 c) preventing unauthorised modification, addition, or deletion of TOE configuration items.

#### 3402 **12.2.2 Component levelling**

3403 The components in this family are levelled on the basis of the CM system capabilities, the scope of the  
 3404 CM documentation and the evidence provided by the developer.

#### 3405 **12.2.3 Application notes**

3406

3407 In the case where the TOE is a subset of a product, the requirements of this family apply only to the  
 3408 TOE configuration items, not to the product as a whole.

3409 For developer organizations that specify more than one CM application, or include different instances  
 3410 of a CM application within the scope of the TOEs design, development, production and maintenance, it  
 3411 is required to document all of them. For evaluation purposes, the set of CM applications should be  
 3412 regarded as parts of an overall CM system, applicable to the TOE, which is addressed in the criteria.

3413 The overall CM system should address any aspects of integration between component CM applications.

3414 Several elements of this family refer to configuration items. These elements identify CM requirements  
 3415 to be imposed on all items identified in the configuration list, but leave the contents of the list to the  
 3416 discretion of the developer. CM scope (ALC\_CMS) can be used to narrow this discretion by identifying  
 3417 specific items that must be included in the configuration list, and hence within the scope of the overall  
 3418 CM system.

3419 ALC\_CMC.2.3C introduces a requirement that the CM system uniquely identify all configuration items.  
 3420 This also requires that modifications to configuration items result in a new, unique identifier being  
 3421 assigned to the configuration item.

3422 ALC\_CMC.3.8C introduces the requirement that the evidence shall demonstrate that the CM system  
 3423 operates in accordance with the CM plan. Examples of such evidence might be documentation such as  
 3424 screen snapshots or audit trail output from the CM system, or a detailed demonstration of the CM

3425 system by the developer. The evaluator is responsible for determining that this evidence is sufficient  
3426 to show that the CM system operates in accordance with the CM plan.

3427 ALC\_CMC.4.5C introduces a requirement that the CM system provide an automated means to support  
3428 the production of the TOE. This requires that the CM system provide an automated means to assist in  
3429 determining that the correct configuration items are used in generating the TOE.

3430 ALC\_CMC.5.10C introduces a requirement that the CM system provide an automated means to  
3431 ascertain the changes between the TOE and its preceding version. If no previous version of the TOE  
3432 exists, the developer still needs to provide an automated means to ascertain the changes between the  
3433 TOE and a future version of the TOE.

#### 3434 **12.2.4 ALC\_CMC.1 Labelling of the TOE**

3435 Dependencies: ALC\_CMS.1 TOE CM coverage

##### 3436 **12.2.4.1 Objectives**

3437 A unique reference is required to ensure that there is no ambiguity in terms of which instance of the  
3438 TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be  
3439 aware of which instance of the TOE they are using.

##### 3440 **12.2.4.2 Developer action elements**

###### 3441 **12.2.4.2.1 ALC\_CMC.1.1D**

3442 **The developer shall provide the TOE and a unique reference for the TOE.**

##### 3443 **12.2.4.3 Content and presentation elements**

###### 3444 **12.2.4.3.1 ALC\_CMC.1.1C**

3445 **The TOE shall be labelled with its unique reference.**

##### 3446 **12.2.4.4 Evaluator action elements**

###### 3447 **12.2.4.4.1 ALC\_CMC.1.1E**

3448 **The evaluator shall confirm that the information provided meets all requirements for content**  
3449 **and presentation of evidence.**

#### 3450 **12.2.5 ALC\_CMC.2 Use of the CM system**

3451 Dependencies: ALC\_CMS.1 TOE CM coverage

##### 3452 **12.2.5.1 Objectives**

3453 A unique reference is required to ensure that there is no ambiguity in terms of which instance of the  
3454 TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be  
3455 aware of which instance of the TOE they are using.

3456 Unique identification of the configuration items leads to a clearer understanding of the composition of  
3457 the TOE, which in turn helps to determine those items which are subject to the evaluation  
3458 requirements for the TOE.

3459 The use of a CM system increases assurance that the configuration items are maintained in a  
3460 controlled manner.

3461 **12.2.5.2 Developer action elements**

3462 **12.2.5.2.1 ALC\_CMC.2.1D**

3463 The developer shall provide the TOE and a unique reference for the TOE.

3464 **12.2.5.2.2 ALC\_CMC.2.2D**

3465 **The developer shall provide the CM documentation.**

3466 **12.2.5.2.3 ALC\_CMC.2.3D**

3467 **The developer shall use a CM system.**

3468 **12.2.5.3 Content and presentation elements**

3469 **12.2.5.3.1 ALC\_CMC.2.1C**

3470 The TOE shall be labelled with its unique reference.

3471 **12.2.5.3.2 ALC\_CMC.2.2C**

3472 **The CM documentation shall describe the method used to uniquely identify the configuration**  
3473 **items.**

3474 **12.2.5.3.3 ALC\_CMC.2.3C**

3475 **The CM system shall uniquely identify all configuration items.**

3476 **12.2.5.4 Evaluator action elements**

3477 **12.2.5.4.1 ALC\_CMC.2.1E**

3478 The evaluator shall confirm that the information provided meets all requirements for content and  
3479 presentation of evidence.

3480 **12.2.6 ALC\_CMC.3 Authorisation controls**

3481 Dependencies: ALC\_CMS.1 TOE CM coverage

3482 ALC\_DVS.1 Identification of security measures

3483 A life-cycle model encompasses the procedures, tools and techniques used to develop  
3484 and maintain the TOE. Aspects of the process that may be covered by such a model include design  
3485 methods, review procedures, project management controls, change control procedures, test methods  
3486 and acceptance procedures. An effective life-cycle model will address these aspects of the  
3487 development and maintenance process within an overall management structure that assigns  
3488 responsibilities and monitors progress.

3489 There are different types of acceptance situations that are dealt with at different locations in the  
3490 criteria:

- 3491       • acceptance of parts delivered by subcontractors (“integration”) should be treated in this family
- 3492       • Development Life-cycle definition (ALC\_LCD),
- 3493       • acceptance subsequent to internal transportations in Developer environment security  
3494       (ALC\_DVS),
- 3495       • acceptance of parts into the CM system in CM capabilities (ALC\_CMC), and
- 3496       • acceptance of the delivered TOE by the consumer in Delivery (ALC\_DEL).
- 3497       The first three types may overlap.
- 3498       Although life-cycle definition deals with the maintenance of the TOE and hence with aspects becoming  
3499       relevant after the completion of the evaluation, its evaluation adds assurance through an analysis of  
3500       the life-cycle information for the TOE provided at the time of the evaluation.
- 3501       A life-cycle model provides for the necessary control over the development and maintenance of the  
3502       TOE, if the model enables sufficient minimisation of the danger that the TOE will not meet its security  
3503       requirement.
- 3504       A measurable life-cycle model is a model using some quantitative valuation (arithmetic parameters  
3505       and/or metrics) of the managed product in order to measure development properties of the product.  
3506       Typical metrics are source code complexity metrics, defect density (errors per size of code) or mean  
3507       time to failure. For the security evaluation all those metrics are of relevance, which are used to  
3508       increase quality by decreasing the probability of faults and thereby in turn increasing assurance in the  
3509       security of the TOE.
- 3510       One should take into account that there exist standardised life-cycle models on the one hand (like the  
3511       waterfall model) and standardised metrics on the other hand (like error density), which may be  
3512       combined. ISO/IEC 15408 does not require the life-cycle to follow exactly one standard defining both  
3513       aspects.
- 3514       ALC\_LCD.1 Developer defined life-cycle processes
- 3515       **12.2.6.1 Objectives**
- 3516       A unique reference is required to ensure that there is no ambiguity in terms of which instance of the  
3517       TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be  
3518       aware of which instance of the TOE they are using.
- 3519       Unique identification of the configuration items leads to a clearer understanding of the composition of  
3520       the TOE, which in turn helps to determine those items which are subject to the evaluation  
3521       requirements for the TOE.
- 3522       The use of a CM system increases assurance that the configuration items are maintained in a  
3523       controlled manner.
- 3524       Providing controls to ensure that unauthorised modifications are not made to the TOE (“CM access  
3525       control”), and ensuring proper functionality and use of the CM system, helps to maintain the integrity  
3526       of the TOE.

- 3527 **12.2.6.2 Developer action elements**
- 3528 **12.2.6.2.1 ALC\_CMC.3.1D**
- 3529 The developer shall provide the TOE and a unique reference for the TOE.
- 3530 **12.2.6.2.2 ALC\_CMC.3.2D**
- 3531 The developer shall provide the CM documentation.
- 3532 **12.2.6.2.3 ALC\_CMC.3.3D**
- 3533 The developer shall use a CM system.
- 3534 **12.2.6.3 Content and presentation elements**
- 3535 **12.2.6.3.1 ALC\_CMC.3.1C**
- 3536 The TOE shall be labelled with its unique reference.
- 3537 **12.2.6.3.2 ALC\_CMC.3.2C**
- 3538 The CM documentation shall describe the method used to uniquely identify the configuration items.
- 3539 **12.2.6.3.3 ALC\_CMC.3.3C**
- 3540 The CM system shall uniquely identify all configuration items.
- 3541 **12.2.6.3.4 ALC\_CMC.3.4C**
- 3542 **The CM system shall provide controls such that only authorised changes are made to the**  
3543 **configuration items.**
- 3544 **12.2.6.3.5 ALC\_CMC.3.5C**
- 3545 **The CM documentation shall include a CM plan.**
- 3546 **12.2.6.3.6 ALC\_CMC.3.6C**
- 3547 **The CM plan shall describe how the CM system is used for the development of the TOE.**
- 3548 **12.2.6.3.7 ALC\_CMC.3.7C**
- 3549 **The evidence shall demonstrate that all configuration items are being maintained under the CM**  
3550 **system.**
- 3551 **12.2.6.3.8 ALC\_CMC.3.8C**
- 3552 **The evidence shall demonstrate that the CM system is being operated in accordance with the**  
3553 **CM plan.**

3554 **12.2.6.4 Evaluator action elements**3555 **12.2.6.4.1 ALC\_CMC.3.1E**

3556 The evaluator shall confirm that the information provided meets all requirements for content and  
3557 presentation of evidence.

3558 **12.2.7 ALC\_CMC.4 Production support, acceptance procedures and automation**

3559 Dependencies: ALC\_CMS.1 TOE CM coverage

3560 ALC\_DVS.1 Identification of security measures

3561 A life-cycle model encompasses the procedures, tools and techniques used to develop  
3562 and maintain the TOE. Aspects of the process that may be covered by such a model include design  
3563 methods, review procedures, project management controls, change control procedures, test methods  
3564 and acceptance procedures. An effective life-cycle model will address these aspects of the  
3565 development and maintenance process within an overall management structure that assigns  
3566 responsibilities and monitors progress.

3567 There are different types of acceptance situations that are dealt with at different locations in the  
3568 criteria:

- 3569 • acceptance of parts delivered by subcontractors (“integration”) should be treated in this family
- 3570 • Development Life-cycle definition (ALC\_LCD),
- 3571 • acceptance subsequent to internal transportations in Developer environment security  
3572 (ALC\_DVS),
- 3573 • acceptance of parts into the CM system in CM capabilities (ALC\_CMC), and
- 3574 • acceptance of the delivered TOE by the consumer in Delivery (ALC\_DEL).

3575 The first three types may overlap.

3576 Although life-cycle definition deals with the maintenance of the TOE and hence with aspects becoming  
3577 relevant after the completion of the evaluation, its evaluation adds assurance through an analysis of  
3578 the life-cycle information for the TOE provided at the time of the evaluation.

3579 A life-cycle model provides for the necessary control over the development and maintenance of the  
3580 TOE, if the model enables sufficient minimisation of the danger that the TOE will not meet its security  
3581 requirement.

3582 A measurable life-cycle model is a model using some quantitative valuation (arithmetic parameters  
3583 and/or metrics) of the managed product in order to measure development properties of the product.  
3584 Typical metrics are source code complexity metrics, defect density (errors per size of code) or mean  
3585 time to failure. For the security evaluation all those metrics are of relevance, which are used to  
3586 increase quality by decreasing the probability of faults and thereby in turn increasing assurance in the  
3587 security of the TOE.

3588 One should take into account that there exist standardised life-cycle models on the one hand (like the  
3589 waterfall model) and standardised metrics on the other hand (like error density), which may be

3590 combined. ISO/IEC 15408 does not require the life-cycle to follow exactly one standard defining both  
3591 aspects.

3592 ALC\_LCD.1 Developer defined life-cycle processes

3593 **12.2.7.1 Objectives**

3594 A unique reference is required to ensure that there is no ambiguity in terms of which instance of the  
3595 TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be  
3596 aware of which instance of the TOE they are using.

3597 Unique identification of the configuration items leads to a clearer understanding of the composition of  
3598 the TOE, which in turn helps to determine those items which are subject to the evaluation  
3599 requirements for the TOE.

3600 The use of a CM system increases assurance that the configuration items are maintained in a  
3601 controlled manner.

3602 Providing access controls to help ensure that unauthorised modifications are not made to the TOE  
3603 ("CM access control"), and ensuring proper functionality and use of the CM system, helps to maintain  
3604 the integrity of the TOE.

3605 The purpose of the acceptance procedures is to ensure that the parts of the TOE are of adequate  
3606 quality and to confirm that any creation or modification of configuration items is authorised.  
3607 Acceptance procedures are an essential element in integration processes and in the life-cycle  
3608 management of the TOE.

3609 In a CM system where the quantity and organization of configuration items is complex, it is difficult to  
3610 control changes without the support of automated tools. In particular, these automated tools need to  
3611 be able to support the numerous changes that occur during development and ensure that those  
3612 changes are authorised. It is an objective of this component to ensure that the configuration items are  
3613 controlled through automated means. In the case where the overall CM system includes more than one  
3614 CM application then automated tools can also support integration between the CM applications and of  
3615 the TOE.

3616 Production support procedures help to ensure that the generation of the TOE from a managed set of  
3617 configuration items is correctly performed in an authorised manner, particularly in the case when  
3618 different developers are involved and integration processes have to be carried out.

3619 **12.2.7.2 Developer action elements**

3620 **12.2.7.2.1 ALC\_CMC.4.1D**

3621 The developer shall provide the TOE and a unique reference for the TOE.

3622 **12.2.7.2.2 ALC\_CMC.4.2D**

3623 The developer shall provide the CM documentation.

3624 **12.2.7.2.3 ALC\_CMC.4.3D**

3625 The developer shall use a CM system.

- 3626 **12.2.7.3 Content and presentation elements**
- 3627 **12.2.7.3.1 ALC\_CMC.4.1C**
- 3628 The TOE shall be labelled with its unique reference.
- 3629 **12.2.7.3.2 ALC\_CMC.4.2C**
- 3630 The CM documentation shall describe the method or methods used to uniquely identify the  
3631 configuration items.
- 3632 **12.2.7.3.3 ALC\_CMC.4.3C**
- 3633 The CM system shall uniquely identify all configuration items.
- 3634 **12.2.7.3.4 ALC\_CMC.4.4C**
- 3635 The CM system shall provide **automated** controls such that only authorised changes are made to the  
3636 configuration items.
- 3637 **12.2.7.3.5 ALC\_CMC.4.5C**
- 3638 **The CM system shall support the production of the TOE by automated means.**
- 3639 **12.2.7.3.6 ALC\_CMC.4.6C**
- 3640 The CM documentation shall include a CM plan.
- 3641 **12.2.7.3.7 ALC\_CMC.4.7C**
- 3642 The CM plan shall describe how the CM system is used for the development of the TOE.
- 3643 **12.2.7.3.8 ALC\_CMC.4.8C**
- 3644 **The CM plan shall describe the procedures used to accept modified or newly created**  
3645 **configuration items as part of the TOE.**
- 3646 **12.2.7.3.9 ALC\_CMC.4.9C**
- 3647 The evidence shall demonstrate that all configuration items are being maintained under the CM  
3648 system.
- 3649 **12.2.7.3.10 ALC\_CMC.4.10C**
- 3650 The evidence shall demonstrate that the CM system is being operated in accordance with the CM plan.
- 3651 **12.2.7.4 Evaluator action elements**
- 3652 **12.2.7.4.1 ALC\_CMC.4.1E**
- 3653 The evaluator shall confirm that the information provided meets all requirements for content and  
3654 presentation of evidence.
- 3655 **12.2.8 ALC\_CMC.5 Advanced support**
- 3656 Dependencies: ALC\_CMS.1 TOE CM coverage

3657 ALC\_DVS.2 Sufficiency of security measures

3658 A life-cycle model encompasses the procedures, tools and techniques used to develop  
 3659 and maintain the TOE. Aspects of the process that may be covered by such a model include design  
 3660 methods, review procedures, project management controls, change control procedures, test methods  
 3661 and acceptance procedures. An effective life-cycle model will address these aspects of the  
 3662 development and maintenance process within an overall management structure that assigns  
 3663 responsibilities and monitors progress.

3664 There are different types of acceptance situations that are dealt with at different locations in the  
 3665 criteria:

- 3666 • acceptance of parts delivered by subcontractors (“integration”) should be treated in this family
- 3667 • Development Life-cycle definition (ALC\_LCD),
- 3668 • acceptance subsequent to internal transportations in Developer environment security  
 3669 (ALC\_DVS),
- 3670 • acceptance of parts into the CM system in CM capabilities (ALC\_CMC), and
- 3671 • acceptance of the delivered TOE by the consumer in Delivery (ALC\_DEL).

3672 The first three types may overlap.

3673 Although life-cycle definition deals with the maintenance of the TOE and hence with aspects becoming  
 3674 relevant after the completion of the evaluation, its evaluation adds assurance through an analysis of  
 3675 the life-cycle information for the TOE provided at the time of the evaluation.

3676 A life-cycle model provides for the necessary control over the development and maintenance of the  
 3677 TOE, if the model enables sufficient minimisation of the danger that the TOE will not meet its security  
 3678 requirement.

3679 A measurable life-cycle model is a model using some quantitative valuation (arithmetic parameters  
 3680 and/or metrics) of the managed product in order to measure development properties of the product.  
 3681 Typical metrics are source code complexity metrics, defect density (errors per size of code) or mean  
 3682 time to failure. For the security evaluation all those metrics are of relevance, which are used to  
 3683 increase quality by decreasing the probability of faults and thereby in turn increasing assurance in the  
 3684 security of the TOE.

3685 One should take into account that there exist standardised life-cycle models on the one hand (like the  
 3686 waterfall model) and standardised metrics on the other hand (like error density), which may be  
 3687 combined. ISO/IEC 15408 does not require the life-cycle to follow exactly one standard defining both  
 3688 aspects.

3689 ALC\_LCD.1 Developer defined life-cycle processes

### 3690 **12.2.8.1 Objectives**

3691 A unique reference is required to ensure that there is no ambiguity in terms of which instance of the  
 3692 TOE is being evaluated. Labelling the TOE with its reference ensures that users of the TOE can be  
 3693 aware of which instance of the TOE they are using.

- 3694 Unique identification of the configuration items leads to a clearer understanding of the composition of  
 3695 the TOE, which in turn helps to determine those items which are subject to the evaluation  
 3696 requirements for the TOE.
- 3697 The use of a CM system increases assurance that the configuration items are maintained in a  
 3698 controlled manner.
- 3699 Providing controls to ensure that unauthorised modifications are not made to the TOE (“CM access  
 3700 control”), and ensuring proper functionality and use of the CM system, helps to maintain the integrity  
 3701 of the TOE.
- 3702 The purpose of the acceptance procedures is to ensure that the parts of the TOE meet defined criteria  
 3703 in regard to the integrity of the TOE. Criteria for acceptance procedures may include code review,  
 3704 checking for vulnerabilities, authenticity checking, and functional testing to confirm that any creation  
 3705 or modification of configuration items is authorised. Acceptance procedures are an essential element  
 3706 in integration processes and in the life-cycle management of the TOE.
- 3707 In development environments where the configuration items are complex, it is difficult to control  
 3708 changes without the support of automated tools. In particular, these automated tools need to be able  
 3709 to support the numerous changes that occur during development and ensure that those changes are  
 3710 authorised. It is an objective of this component to ensure that the configuration items are controlled  
 3711 through automated means. If the TOE is developed by multiple developers, i.e. integration has to take  
 3712 place, the use of automatic tools is adequate.
- 3713 Production support procedures help to ensure that the generation of the TOE from a managed set of  
 3714 configuration items is correctly performed in an authorised manner, particularly in the case when  
 3715 different developers are involved and integration processes have to be carried out.
- 3716 Requiring that the CM system be able to identify the version of the implementation representation  
 3717 from which the TOE is generated helps to ensure that the integrity of this material is preserved by the  
 3718 appropriate technical, physical and procedural safeguards.
- 3719 Providing an automated means of ascertaining changes between versions of the TOE and identifying  
 3720 which configuration items are affected by modifications to other configuration items assists in  
 3721 determining the impact of the changes between successive versions of the TOE. This in turn can  
 3722 provide valuable information in determining whether changes to the TOE result in all configuration  
 3723 items being consistent with one another.
- 3724 **12.2.8.2 Developer action elements**
- 3725 **12.2.8.2.1 ALC\_CMC.5.1D**
- 3726 The developer shall provide the TOE and a unique reference for the TOE.
- 3727 **12.2.8.2.2 ALC\_CMC.5.2D**
- 3728 The developer shall provide the CM documentation.
- 3729 **12.2.8.2.3 ALC\_CMC.5.3D**
- 3730 The developer shall use a CM system.

- 3731 **12.2.8.3 Content and presentation elements**
- 3732 **12.2.8.3.1 ALC\_CMC.5.1C**
- 3733 The TOE shall be labelled with its unique reference.
- 3734 **12.2.8.3.2 ALC\_CMC.5.2C**
- 3735 The CM documentation shall describe the method used to uniquely identify the configuration items.
- 3736 **12.2.8.3.3 ALC\_CMC.5.3C**
- 3737 **The CM documentation shall justify that the acceptance procedures provide for an adequate**  
3738 **and appropriate review of changes to all configuration items.**
- 3739 **12.2.8.3.4 ALC\_CMC.5.4C**
- 3740 The CM system shall uniquely identify all configuration items.
- 3741 **12.2.8.3.5 ALC\_CMC.5.5C**
- 3742 The CM system shall provide automated controls such that only authorised changes are made to the  
3743 configuration items.
- 3744 **12.2.8.3.6 ALC\_CMC.5.6C**
- 3745 The CM system shall support the production of the TOE by automated means.
- 3746 **12.2.8.3.7 ALC\_CMC.5.7C**
- 3747 **The CM system shall ensure that the person responsible for accepting a configuration item into**  
3748 **CM is not the person who developed it.**
- 3749 **12.2.8.3.8 ALC\_CMC.5.8C**
- 3750 **The CM system shall identify the configuration items that comprise the TSF.**
- 3751 **12.2.8.3.9 ALC\_CMC.5.9C**
- 3752 **The CM system shall support the audit of all changes to the TOE by automated means, including**  
3753 **the originator, date, and time in the audit trail.**
- 3754 **12.2.8.3.10 ALC\_CMC.5.10C**
- 3755 **The CM system shall provide an automated means to identify all other configuration items that**  
3756 **are affected by the change of a given configuration item.**
- 3757 **12.2.8.3.11 ALC\_CMC.5.11C**
- 3758 **The CM system shall be able to identify the version of the implementation representation from**  
3759 **which the TOE is generated.**
- 3760 **12.2.8.3.12 ALC\_CMC.5.12C**
- 3761 The CM documentation shall include a CM plan.

3762 **12.2.8.3.13 ALC\_CMC.5.13C**

3763 The CM plan shall describe how the CM system is used for the development of the TOE.

3764 **12.2.8.3.14 ALC\_CMC.5.14C**3765 The CM plan shall describe the procedures used to accept modified or newly created configuration  
3766 items as part of the TOE.3767 **12.2.8.3.15 ALC\_CMC.5.15C**3768 The evidence shall demonstrate that all configuration items are being maintained under the CM  
3769 system.3770 **12.2.8.3.16 ALC\_CMC.5.16C**

3771 The evidence shall demonstrate that the CM system is being operated in accordance with the CM plan.

3772 **12.2.8.4 Evaluator action elements**3773 **12.2.8.4.1 ALC\_CMC.5.1E**3774 The evaluator shall confirm that the information provided meets all requirements for content and  
3775 presentation of evidence.3776 **12.2.8.4.2 ALC\_CMC.5.2E**3777 **The evaluator shall determine that the application of the production support procedures**  
3778 **results in a TOE as provided by the developer for testing activities.**3779 **12.3 CM scope (ALC\_CMS)**3780 **12.3.1 Objectives**3781 The objective of this family is to identify items to be included as configuration items and hence placed  
3782 under the CM requirements of CM capabilities (ALC\_CMC). Applying configuration management to  
3783 these additional items provides additional assurance that the integrity of TOE is maintained.3784 **12.3.2 Component levelling**3785 The components in this family are levelled on the basis of which of the following are required to be  
3786 included as configuration items: the TOE and the evaluation evidence required by the SARs; the parts  
3787 of the TOE; the implementation representation; security flaws; and development tools and related  
3788 information.3789 **12.3.3 Application notes**3790 While CM scope (ALC\_CMS) mandates a list of configuration items and that each item on this list be  
3791 under CM, CM capabilities (ALC\_CMC) leaves the contents of the configuration list to the discretion of  
3792 the developer. CM scope (ALC\_CMS) narrows this discretion by identifying items that must be included  
3793 in the configuration list, and hence come under the CM requirements of CM capabilities (ALC\_CMC).3794 **12.3.4 ALC\_CMS.1 TOE CM coverage**

3795 Dependencies: No dependencies.

3796 **12.3.4.1 Objectives**

3797 A CM system can control changes only to those items that have been placed under CM (i.e., the  
3798 configuration items identified in the configuration list). Placing the TOE itself and the evaluation  
3799 evidence required by the other SARs in the ST under CM provides assurance that they have been  
3800 modified in a controlled manner with proper authorisations.

3801 **12.3.4.2 Application notes**

3802 ALC\_CMS.1.1C introduces the requirement that the TOE itself and the evaluation evidence required by  
3803 the other SARs in the ST be included in the configuration list and hence be subject to the CM  
3804 requirements of CM capabilities (ALC\_CMC).

3805 **12.3.4.3 Developer action elements**

3806 **12.3.4.3.1 ALC\_CMS.1.1D**

3807 **The developer shall provide a configuration list for the TOE.**

3808 **12.3.4.4 Content and presentation elements**

3809 **12.3.4.4.1 ALC\_CMS.1.1C**

3810 **The configuration list shall include the following: the TOE itself; and the evaluation evidence**  
3811 **required by the SARs.**

3812 **12.3.4.4.2 ALC\_CMS.1.2C**

3813 **The configuration list shall uniquely identify the configuration items.**

3814 **12.3.4.5 Evaluator action elements**

3815 **12.3.4.5.1 ALC\_CMS.1.1E**

3816 **The evaluator shall confirm that the information provided meets all requirements for content**  
3817 **and presentation of evidence.**

3818 **12.3.5 ALC\_CMS.2 Parts of the TOE CM coverage**

3819 Dependencies: No dependencies.

3820 **12.3.5.1 Objectives**

3821 A CM system can control changes only to those items that have been placed under CM (i.e., the  
3822 configuration items identified in the configuration list). Placing the TOE itself, the parts that comprise  
3823 the TOE, and the evaluation evidence required by the other SARs under CM provides assurance that  
3824 they have been modified in a controlled manner with proper authorisations.

3825 **12.3.5.2 Application notes**

3826 ALC\_CMS.2.1C introduces the requirement that the parts that comprise the TOE (all parts that are  
3827 delivered to the consumer, for example hardware parts or executable files) be included in the  
3828 configuration list and hence be subject to the CM requirements of CM capabilities (ALC\_CMC).

3829 ALC\_CMS.2.3C introduces the requirement that the configuration list indicate the developer of each  
3830 TSF relevant configuration item.

### 3831 **12.3.5.3 Developer action elements**

#### 3832 **12.3.5.3.1 ALC\_CMS.2.1D**

3833 The developer shall provide a configuration list for the TOE.

### 3834 **12.3.5.4 Content and presentation elements**

#### 3835 **12.3.5.4.1 ALC\_CMS.2.1C**

3836 The configuration list shall include the following: the TOE itself; the evaluation evidence required by  
3837 the SARs; **and the parts that comprise the TOE.**

#### 3838 **12.3.5.4.2 ALC\_CMS.2.2C**

3839 The configuration list shall uniquely identify the configuration items.

#### 3840 **12.3.5.4.3 ALC\_CMS.2.3C**

3841 **For each TSF relevant configuration item, the configuration list shall indicate the developer of**  
3842 **the item.**

### 3843 **12.3.5.5 Evaluator action elements**

#### 3844 **12.3.5.5.1 ALC\_CMS.2.1E**

3845 The evaluator shall confirm that the information provided meets all requirements for content and  
3846 presentation of evidence.

### 3847 **12.3.6 ALC\_CMS.3 Implementation representation CM coverage**

3848 Dependencies: No dependencies.

#### 3849 **12.3.6.1 Objectives**

3850 A CM system can control changes only to those items that have been placed under CM (i.e., the  
3851 configuration items identified in the configuration list). Placing the TOE itself, the parts that comprise  
3852 the TOE, the TOE implementation representation and the evaluation evidence required by the other  
3853 SARs under CM provides assurance that they have been modified in a controlled manner with proper  
3854 authorisations.

#### 3855 **12.3.6.2 Application notes**

3856 ALC\_CMS.3.1C introduces the requirement that the TOE implementation representation be included in  
3857 the list of configuration items and hence be subject to the CM requirements of CM capabilities  
3858 (ALC\_CMC).

### 3859 **12.3.6.3 Developer action elements**

#### 3860 **12.3.6.3.1 ALC\_CMS.3.1D**

3861 The developer shall provide a configuration list for the TOE.

3862 **12.3.6.4 Content and presentation elements**

3863 **12.3.6.4.1 ALC\_CMS.3.1C**

3864 The configuration list shall include the following: the TOE itself; the evaluation evidence required by  
3865 the SARs; the parts that comprise the TOE; **and the implementation representation.**

3866 **12.3.6.4.2 ALC\_CMS.3.2C**

3867 The configuration list shall uniquely identify the configuration items.

3868 **12.3.6.4.3 ALC\_CMS.3.3C**

3869 For each TSF relevant configuration item, the configuration list shall indicate the developer of the item.

3870 **12.3.6.5 Evaluator action elements**

3871 **12.3.6.5.1 ALC\_CMS.3.1E**

3872 The evaluator shall confirm that the information provided meets all requirements for content and  
3873 presentation of evidence.

3874 **12.3.7 ALC\_CMS.4 Problem tracking CM coverage**

3875 Dependencies: No dependencies.

3876 **12.3.7.1 Objectives**

3877 A CM system can control changes only to those items that have been placed under CM (i.e., the  
3878 configuration items identified in the configuration list). Placing the TOE itself, the parts that comprise  
3879 the TOE, the TOE implementation representation and the evaluation evidence required by the other  
3880 SARs under CM provides assurance that they have been modified in a controlled manner with proper  
3881 authorisations.

3882 Placing security flaw reports under CM ensures that the integrity of the reports is maintained and that  
3883 access to them is managed, further, it may support developers in tracking security flaws to their  
3884 resolution.

3885 **12.3.7.2 Application notes**

3886 ALC\_CMS.4.1C introduces the requirement that reports of identified security flaws be included in the  
3887 configuration list and hence be subject to the CM requirements of CM capabilities (ALC\_CMC). This  
3888 requires that information regarding previously identified security flaw reports and their resolution be  
3889 maintained.

3890 **12.3.7.3 Developer action elements**

3891 **12.3.7.3.1 ALC\_CMS.4.1D**

3892 The developer shall provide a configuration list for the TOE.

3893 **12.3.7.4 Content and presentation elements**3894 **12.3.7.4.1 ALC\_CMS.4.1C**

3895 The configuration list shall include the following: the TOE itself; the evaluation evidence required by  
 3896 the SARs; the parts that comprise the TOE; the implementation representation; **and security flaw**  
 3897 **reports and resolution status.**

3898 **12.3.7.4.2 ALC\_CMS.4.2C**

3899 The configuration list shall uniquely identify the configuration items.

3900 **12.3.7.4.3 ALC\_CMS.4.3C**

3901 For each TSF relevant configuration item, the configuration list shall indicate the developer of the item.

3902 **12.3.7.5 Evaluator action elements**3903 **12.3.7.5.1 ALC\_CMS.4.1E**

3904 The evaluator shall confirm that the information provided meets all requirements for content and  
 3905 presentation of evidence.

3906 **12.3.8 ALC\_CMS.5 Development tools CM coverage**

3907 Dependencies: No dependencies.

3908 **12.3.8.1 Objectives**

3909 A CM system can control changes only to those items that have been placed under CM (i.e., the  
 3910 configuration items identified in the configuration list). Placing the TOE itself, the parts that comprise  
 3911 the TOE, the TOE implementation representation and the evaluation evidence required by the other  
 3912 SARs under CM provides assurance that they have been modified in a controlled manner with proper  
 3913 authorisations.

3914 Placing security flaw reports under CM ensures that the integrity of the reports is maintained and that  
 3915 access to them is managed, further, it may support developers in tracking security flaws to their  
 3916 resolution.

3917 Development tools play an important role in ensuring the production of a quality version of the TOE.  
 3918 Therefore, it is important to control modifications to these tools.

3919 **12.3.8.2 Application notes**

3920 ALC\_CMS.5.1C introduces the requirement that development tools and other related information be  
 3921 included in the list of configuration items and hence be subject to the CM requirements of CM  
 3922 capabilities (ALC\_CMC). Examples of development tools are programming languages and compilers.  
 3923 Information pertaining to TOE generation items (such as compiler options, generation options, and  
 3924 build options) is an example of information relating to development tools.

3925 **12.3.8.3 Developer action elements**3926 **12.3.8.3.1 ALC\_CMS.5.1D**

3927 The developer shall provide a configuration list for the TOE.

3928 **12.3.8.4 Content and presentation elements**

3929 **12.3.8.4.1 ALC\_CMS.5.1C**

3930 The configuration list shall include the following: the TOE itself; the evaluation evidence required by  
3931 the SARs; the parts that comprise the TOE; the implementation representation; security flaw reports  
3932 and resolution status; **and development tools and related information.**

3933 **12.3.8.4.2 ALC\_CMS.5.2C**

3934 The configuration list shall uniquely identify the configuration items.

3935 **12.3.8.4.3 ALC\_CMS.5.3C**

3936 For each TSF relevant configuration item, the configuration list shall indicate the developer of the item.

3937 **12.3.8.5 Evaluator action elements**

3938 **12.3.8.5.1 ALC\_CMS.5.1E**

3939 The evaluator shall confirm that the information provided meets all requirements for content and  
3940 presentation of evidence.

3941 **12.4 Delivery (ALC\_DEL)**

3942 **12.4.1 Objectives**

3943 The concern of this family is the secure transfer of the finished TOE from the development  
3944 environment into the responsibility of the user.

3945 The requirements for delivery call for system control and distribution facilities and procedures that  
3946 detail the controls necessary to provide assurance that the security of the TOE is maintained during  
3947 distribution of the TOE to the user. For a valid distribution of the TOE, the procedures used for the  
3948 distribution of the TOE address the implied or identified objectives identified in the PP/ST relating to  
3949 the security of the TOE during delivery.

3950 **12.4.2 Component levelling**

3951 This family contains only one component. An increasing level of protection for the TOE is established  
3952 by requiring that the delivery procedures are commensurate with the assumed attack potential in the  
3953 family Vulnerability analysis (AVA\_VAN) specified in the ST.

3954 **12.4.3 Application notes**

3955 Transfers from subcontractors to the developer or between different development sites are not  
3956 considered here, but in the family Developer environment security (ALC\_DVS).

3957 The end of the delivery phase is marked by the acceptance of the transfer of the TOE into the  
3958 responsibility of the downstream user.

3959 NOTE: This does not necessarily coincide with the arrival of the TOE at the downstream user's  
3960 location.

3961 The delivery procedures should consider, if applicable, issues such as:

- 3962 a) ensuring that the TOE received by the consumer corresponds precisely to the evaluated version of  
3963 the TOE;
- 3964 b) avoiding or detecting any tampering with the actual version of the TOE;
- 3965 c) preventing submission of a counterfeit version of the TOE;
- 3966 d) avoiding unwanted knowledge of distribution of the TOE to the consumer: there might be cases  
3967 where potential attackers should not know when and how it is delivered;
- 3968 e) avoiding or detecting the TOE being intercepted during delivery; and
- 3969 f) avoiding the TOE being delayed or stopped during distribution.

3970 The delivery procedures should include the recipient's actions implied by these issues. The consistent  
3971 description of these implied actions is examined in the Preparative procedures (AGD\_PRE) family, if  
3972 present.

#### 3973 **12.4.4 ALC\_DEL.1 Delivery procedures**

3974 Dependencies: No dependencies.

##### 3975 **12.4.4.1 Developer action elements**

###### 3976 **12.4.4.1.1 ALC\_DEL.1.1D**

3977 **The developer shall document and provide procedures for delivery of the TOE or parts of it to**  
3978 **the consumer.**

###### 3979 **12.4.4.1.2 ALC\_DEL.1.2D**

3980 **The developer shall use the delivery procedures.**

##### 3981 **12.4.4.2 Content and presentation elements**

###### 3982 **12.4.4.2.1 ALC\_DEL.1.1C**

3983 **The delivery documentation shall describe all procedures that are necessary to maintain**  
3984 **security when distributing versions of the TOE to the consumer.**

##### 3985 **12.4.4.3 Evaluator action elements**

###### 3986 **12.4.4.3.1 ALC\_DEL.1.1E**

3987 **The evaluator shall confirm that the information provided meets all requirements for content**  
3988 **and presentation of evidence.**

#### 3989 **12.5 Developer environment security (ALC\_DVS)**

##### 3990 **12.5.1 Objectives**

3991 Development security is concerned with the determination and specification of security controls  
3992 relating to the developer provided environment.

3993 NOTE: Such controls include coverage of security relevant aspects of asset management, human  
 3994 resources security, physical and environmental security, communications and operations  
 3995 management, access control, information systems acquisition, development and maintenance,  
 3996 information security incident management, and business continuity management.

### 3997 **12.5.2 Component levelling**

3998 The components in this family are levelled on the basis of whether justification of the sufficiency of the  
 3999 security controls is required.

### 4000 **12.5.3 Application notes**

4001 This family deals with controls to remove or reduce threads and security risks existing at the  
 4002 developer's site.

4003 The evaluator should visit the site(s) in order to assess evidence for development security. This may  
 4004 include sites of subcontractors involved in the TOE development and production. Any decision not to  
 4005 visit shall be agreed with the evaluation authority.

4006 Although development security deals with the maintenance of the TOE and hence with aspects  
 4007 becoming relevant after the completion of the evaluation, the Developer environment security  
 4008 (ALC\_DVS) requirements specify only that the development security controls be in place at the time of  
 4009 evaluation. Furthermore, Developer environment security (ALC\_DVS) does not contain any  
 4010 requirements related to the sponsor's intention to apply the development security controls in the  
 4011 future, after completion of the evaluation.

4012 It is recognised that confidentiality may not always be an issue for the protection of the TOE in its  
 4013 development environment. The use of the word "necessary" allows for the selection of appropriate  
 4014 safeguards.

### 4015 **12.5.4 ALC\_DVS.1 Identification of security controls**

4016 Dependencies: No dependencies.

#### 4017 **12.5.4.1 Developer action elements**

##### 4018 **12.5.4.1.1 ALC\_DVS.1.1D**

4019 **The developer shall produce and provide development security documentation.**

#### 4020 **12.5.4.2 Content and presentation elements**

##### 4021 **12.5.4.2.1 ALC\_DVS.1.1C**

4022 **The development security documentation shall describe all the physical, logical, procedural,**  
 4023 **personnel, and other security controls that are necessary to protect the confidentiality and**  
 4024 **integrity of the TOE design and implementation in its development environment.**

#### 4025 **12.5.4.3 Evaluator action elements**

##### 4026 **12.5.4.3.1 ALC\_DVS.1.1E**

4027 **The evaluator shall confirm that the information provided meets all requirements for content**  
 4028 **and presentation of evidence.**

4029 **12.5.4.3.2 ALC\_DVS.1.2E**

4030 **The evaluator shall confirm that the security controls are being applied.**

4031 **12.5.5 ALC\_DVS.2 Sufficiency of security controls**

4032 Dependencies: No dependencies.

4033 **12.5.5.1 Developer action elements**

4034 **12.5.5.1.1 ALC\_DVS.2.1D**

4035 The developer shall produce and provide development security documentation.

4036 **12.5.5.2 Content and presentation elements**

4037 **12.5.5.2.1 ALC\_DVS.2.1C**

4038 The development security documentation shall describe all the physical, procedural, personnel, and  
4039 other security controls that are necessary to protect the confidentiality and integrity of the TOE design  
4040 and implementation in its development environment.

4041 **12.5.5.2.2 ALC\_DVS.2.2C**

4042 **The development security documentation shall justify that the security controls provide the**  
4043 **necessary level of protection to maintain the confidentiality and integrity of the TOE.**

4044 **12.5.5.3 Evaluator action elements**

4045 **12.5.5.3.1 ALC\_DVS.2.1E**

4046 The evaluator shall confirm that the information provided meets all requirements for content and  
4047 presentation of evidence.

4048 **12.5.5.3.2 ALC\_DVS.2.2E**

4049 The evaluator shall confirm that the security controls are being applied.

4050 **12.6 Flaw remediation (ALC\_FLR)**

4051 **12.6.1 Objectives**

4052 Flaw remediation requires that discovered security flaws be tracked and corrected by the developer.  
4053 Although future compliance with flaw remediation procedures cannot be determined at the time of the  
4054 TOE evaluation, it is possible to evaluate the policies and procedures that a developer has in place to  
4055 track and correct flaws, and to distribute the flaw information and corrections.

4056 **12.6.2 Component levelling**

4057 The components in this family are levelled on the basis of the increasing extent in scope of the flaw  
4058 remediation procedures and the rigour of the flaw remediation policies.

4059 **12.6.3 Application notes**

4060 This family provides assurance that the TOE will be maintained and supported in the future, requiring  
4061 the TOE developer to track and correct flaws in the TOE. Additionally, requirements are included for

4062 the distribution of flaw corrections. However, this family does not impose evaluation requirements  
4063 beyond the current evaluation.

4064 The TOE user is considered to be the focal point in the user organisation that is responsible for  
4065 receiving and implementing fixes to security flaws. This is not necessarily an individual user, but may  
4066 be an organisational representative who is responsible for the handling of security flaws. The use of  
4067 the term TOE user recognises that different organisations have different procedures for handling flaw  
4068 reporting, which may be done either by an individual user, or by a central administrative body.

4069 The flaw remediation procedures should describe the methods for dealing with all types of flaws  
4070 encountered. These flaws may be reported by the developer, by users of the TOE, or by other parties  
4071 with familiarity with the TOE. Some flaws may not be reparable immediately. There may be some  
4072 occasions where a flaw cannot be fixed and other (e.g. procedural) controls must be taken. The  
4073 documentation provided should cover the procedures for providing the operational sites with fixes,  
4074 and providing information on flaws where fixes are delayed (and what to do in the interim) or when  
4075 fixes are not possible.

4076 Changes applied to a TOE after its release render it unevaluated; although some information from the  
4077 original evaluation may still apply. The phrase “release of the TOE” used in this family therefore refers  
4078 to a version of a product that is a release of a certified TOE, to which changes have been applied.

#### 4079 **12.6.4 ALC\_FLR.1 Basic flaw remediation**

4080 Dependencies: No dependencies.

##### 4081 **12.6.4.1 Developer action elements**

###### 4082 **12.6.4.1.1 ALC\_FLR.1.1D**

4083 **The developer shall document and provide flaw remediation procedures addressed to TOE**  
4084 **developers.**

##### 4085 **12.6.4.2 Content and presentation elements**

###### 4086 **12.6.4.2.1 ALC\_FLR.1.1C**

4087 **The flaw remediation procedures documentation shall describe the procedures used to track**  
4088 **all reported security flaws in each release of the TOE.**

###### 4089 **12.6.4.2.2 ALC\_FLR.1.2C**

4090 **The flaw remediation procedures shall require that a description of the nature and effect of**  
4091 **each security flaw be provided, as well as the status of finding a correction to that flaw.**

###### 4092 **12.6.4.2.3 ALC\_FLR.1.3C**

4093 **The flaw remediation procedures shall require that corrective actions be identified for each of**  
4094 **the security flaws.**

###### 4095 **12.6.4.2.4 ALC\_FLR.1.4C**

4096 **The flaw remediation procedures documentation shall describe the methods used to provide**  
4097 **flaw information, corrections and guidance on corrective actions to TOE users.**

4098 **12.6.4.3 Evaluator action elements**4099 **12.6.4.3.1 ALC\_FLR.1.1E**

4100 **The evaluator shall confirm that the information provided meets all requirements for content**  
 4101 **and presentation of evidence.**

4102 **12.6.5 ALC\_FLR.2 Flaw reporting procedures**

4103 Dependencies: No dependencies.

4104 **12.6.5.1 Objectives**

4105 In order for the developer to be able to act appropriately upon security flaw reports from TOE users,  
 4106 and to know to whom to send corrective fixes, TOE users need to understand how to submit security  
 4107 flaw reports to the developer. Flaw remediation guidance from the developer to the TOE user ensures  
 4108 that TOE users are aware of this important information.

4109 **12.6.5.2 Developer action elements**4110 **12.6.5.2.1 ALC\_FLR.2.1D**

4111 The developer shall document and provide flaw remediation procedures addressed to TOE developers.

4112 **12.6.5.2.2 ALC\_FLR.2.2D**

4113 **The developer shall establish a procedure for accepting and acting upon all reports of security**  
 4114 **flaws and requests for corrections to those flaws.**

4115 **12.6.5.2.3 ALC\_FLR.2.3D**

4116 **The developer shall provide flaw remediation guidance addressed to TOE users.**

4117 **12.6.5.3 Content and presentation elements**4118 **12.6.5.3.1 ALC\_FLR.2.1C**

4119 The flaw remediation procedures documentation shall describe the procedures used to track all  
 4120 reported security flaws in each release of the TOE.

4121 **12.6.5.3.2 ALC\_FLR.2.2C**

4122 The flaw remediation procedures shall require that a description of the nature and effect of each  
 4123 security flaw be provided, as well as the status of finding a correction to that flaw.

4124 **12.6.5.3.3 ALC\_FLR.2.3C**

4125 The flaw remediation procedures shall require that corrective actions be identified for each of the  
 4126 security flaws.

4127 **12.6.5.3.4 ALC\_FLR.2.4C**

4128 The flaw remediation procedures documentation shall describe the methods used to provide flaw  
 4129 information, corrections and guidance on corrective actions to TOE users.

- 4130 **12.6.5.3.5 ALC\_FLR.2.5C**
- 4131 **The flaw remediation procedures shall describe a means by which the developer receives from**  
4132 **TOE users reports and enquiries of suspected security flaws in the TOE.**
- 4133 **12.6.5.3.6 ALC\_FLR.2.6C**
- 4134 **The procedures for processing reported security flaws shall ensure that any reported flaws are**  
4135 **remediated and the remediation procedures issued to TOE users.**
- 4136 **12.6.5.3.7 ALC\_FLR.2.7C**
- 4137 **The procedures for processing reported security flaws shall provide safeguards that any**  
4138 **corrections to these security flaws do not introduce any new flaws.**
- 4139 **12.6.5.3.8 ALC\_FLR.2.8C**
- 4140 **The flaw remediation guidance shall describe a means by which TOE users report to the**  
4141 **developer any suspected security flaws in the TOE.**
- 4142 **12.6.5.4 Evaluator action elements**
- 4143 **12.6.5.4.1 ALC\_FLR.2.1E**
- 4144 The evaluator shall confirm that the information provided meets all requirements for content and  
4145 presentation of evidence.
- 4146 **12.6.6 ALC\_FLR.3 Systematic flaw remediation**
- 4147 Dependencies: No dependencies.
- 4148 **12.6.6.1 Objectives**
- 4149 In order for the developer to be able to act appropriately upon security flaw reports from TOE users,  
4150 and to know to whom to send corrective fixes, TOE users need to understand how to submit security  
4151 flaw reports to the developer, and how to register themselves with the developer so that they may  
4152 receive these corrective fixes. Flaw remediation guidance from the developer to the TOE user ensures  
4153 that TOE users are aware of this important information.
- 4154 **12.6.6.2 Developer action elements**
- 4155 **12.6.6.2.1 ALC\_FLR.3.1D**
- 4156 The developer shall document and provide flaw remediation procedures addressed to TOE developers.
- 4157 **12.6.6.2.2 ALC\_FLR.3.2D**
- 4158 The developer shall establish a procedure for accepting and acting upon all reports of security flaws  
4159 and requests for corrections to those flaws.
- 4160 **12.6.6.2.3 ALC\_FLR.3.3D**
- 4161 The developer shall provide flaw remediation guidance addressed to TOE users.

- 4162 **12.6.6.3 Content and presentation elements**
- 4163 **12.6.6.3.1 ALC\_FLR.3.1C**
- 4164 The flaw remediation procedures documentation shall describe the procedures used to track all  
4165 reported security flaws in each release of the TOE.
- 4166 **12.6.6.3.2 ALC\_FLR.3.2C**
- 4167 The flaw remediation procedures shall require that a description of the nature and effect of each  
4168 security flaw be provided, as well as the status of finding a correction to that flaw.
- 4169 **12.6.6.3.3 ALC\_FLR.3.3C**
- 4170 The flaw remediation procedures shall require that corrective actions be identified for each of the  
4171 security flaws.
- 4172 **12.6.6.3.4 ALC\_FLR.3.4C**
- 4173 The flaw remediation procedures documentation shall describe the methods used to provide flaw  
4174 information, corrections and guidance on corrective actions to TOE users.
- 4175 **12.6.6.3.5 ALC\_FLR.3.5C**
- 4176 The flaw remediation procedures shall describe a means by which the developer receives from TOE  
4177 users reports and enquiries of suspected security flaws in the TOE.
- 4178 **12.6.6.3.6 ALC\_FLR.3.6C**
- 4179 **The flaw remediation procedures shall include a procedure requiring timely response and the**  
4180 **automatic distribution of security flaw reports and the associated corrections to registered**  
4181 **users who might be affected by the security flaw.**
- 4182 **12.6.6.3.7 ALC\_FLR.3.7C**
- 4183 The procedures for processing reported security flaws shall ensure that any reported flaws are  
4184 remediated and the remediation procedures issued to TOE users.
- 4185 **12.6.6.3.8 ALC\_FLR.3.8C**
- 4186 The procedures for processing reported security flaws shall provide safeguards that any corrections to  
4187 these security flaws do not introduce any new flaws.
- 4188 **12.6.6.3.9 ALC\_FLR.3.9C**
- 4189 The flaw remediation guidance shall describe a means by which TOE users report to the developer any  
4190 suspected security flaws in the TOE.
- 4191 **12.6.6.3.10 ALC\_FLR.3.10C**
- 4192 **The flaw remediation guidance shall describe a means by which TOE users may register with**  
4193 **the developer, to be eligible to receive security flaw reports and corrections.**

4194 **12.6.6.3.11 ALC\_FLR.3.11C**

4195 **The flaw remediation guidance shall identify the specific points of contact for all reports and**  
 4196 **enquiries about security issues involving the TOE.**

4197 **12.6.6.4 Evaluator action elements**4198 **12.6.6.4.1 ALC\_FLR.3.1E**

4199 The evaluator shall confirm that the information provided meets all requirements for content and  
 4200 presentation of evidence.

4201 **12.7 Development Life-cycle definition (ALC\_LCD)**4202 **12.7.1 Objectives**

4203 Poorly defined or uncontrolled processes applied during the development, production and  
 4204 maintenance of the TOE can result in a TOE that does not meet all of its security objectives. Therefore,  
 4205 it is important that well defined and controlled processes be established as early as possible in the  
 4206 TOE's life-cycle.

4207 Defining and implementing such processes does not guarantee that the TOE meets all of its SFRs. It is  
 4208 possible that the processes will be insufficient or inadequate.

4209 Adopting a life-cycle model, or models that meets the needs of the developer's organization will  
 4210 improve the likelihood that the development, production and maintenance processes applied to TOE  
 4211 support the correct design and implementation of a TOE that meets the specified SFRs.

4212 The determination of appropriate process controls in order to support process improvement is a long  
 4213 established best practice.

4214 **12.7.2 Component levelling**

4215 The components in this family are levelled on the basis of increasing requirements for measurability of  
 4216 the life-cycle model, and for compliance with that model.

4217 **12.7.3 Application notes**

4218 A life-cycle model encompasses the procedures, tools and techniques used to develop and maintain the  
 4219 TOE. Aspects of the process that may be covered by such a model include design methods, review  
 4220 procedures, project management controls, change control procedures, test methods and acceptance  
 4221 procedures. An effective life-cycle model will address these aspects of the development and  
 4222 maintenance process within an overall management structure that assigns responsibilities and  
 4223 monitors progress.

4224 There are different types of acceptance situations that are dealt with at different locations in the  
 4225 criteria:

- 4226 • acceptance of parts delivered by subcontractors ("integration") should be treated in this family
- 4227 • Development Life-cycle definition (ALC\_LCD),
- 4228 • acceptance subsequent to internal transportations in Developer environment security  
 4229 (ALC\_DVS),

4230 • acceptance of parts into the CM system in CM capabilities (ALC\_CMC), and

4231 • acceptance of the delivered TOE by the consumer in Delivery (ALC\_DEL).

4232 The first three types may overlap.

4233 Although life-cycle definition deals with the maintenance of the TOE and hence with aspects becoming  
4234 relevant after the completion of the evaluation, its evaluation adds assurance through an analysis of  
4235 the life-cycle information for the TOE provided at the time of the evaluation.

4236 A life-cycle model provides for the necessary control over the development and maintenance of the  
4237 TOE, if the model enables sufficient minimisation of the danger that the TOE will not meet its security  
4238 requirement.

4239 A measurable life-cycle model is a model using some quantitative valuation (arithmetic parameters  
4240 and/or metrics) of the managed product in order to measure development properties of the product.  
4241 Typical metrics are source code complexity metrics, defect density (errors per size of code) or mean  
4242 time to failure. For the security evaluation all those metrics are of relevance, which are used to  
4243 increase quality by decreasing the probability of faults and thereby in turn increasing assurance in the  
4244 security of the TOE.

4245 One should take into account that there exist standardised life-cycle models on the one hand (like the  
4246 waterfall model) and standardised metrics on the other hand (like error density), which may be  
4247 combined. ISO/IEC 15408 does not require the life-cycle to follow exactly one standard defining both  
4248 aspects.

#### 4249 **12.7.4 ALC\_LCD.1 Developer defined life-cycle processes**

4250 Dependencies: No dependencies.

##### 4251 **12.7.4.1 Developer action elements**

###### 4252 **12.7.4.1.1 ALC\_LCD.1.1D**

4253 **The developer shall establish a life-cycle model to be used in the development and**  
4254 **maintenance of the TOE.**

###### 4255 **12.7.4.1.2 ALC\_LCD.1.2D**

4256 **The developer shall provide life-cycle definition documentation.**

##### 4257 **12.7.4.2 Content and presentation elements**

###### 4258 **12.7.4.2.1 ALC\_LCD.1.1C**

4259 **The life-cycle definition documentation shall describe the processes used to develop and**  
4260 **maintain the TOE.**

###### 4261 **12.7.4.2.2 ALC\_LCD.1.2C**

4262 **The life-cycle model shall provide for the necessary control over the development and**  
4263 **maintenance of the TOE.**

- 4264 **12.7.4.3 Evaluator action elements**
- 4265 **12.7.4.3.1 ALC\_LCD.1.1E**
- 4266 **The evaluator shall confirm that the information provided meets all requirements for content**  
4267 **and presentation of evidence.**
- 4268 **12.7.5 ALC\_LCD.2 Measurable life-cycle model**
- 4269 Dependencies: No dependencies.
- 4270 **12.7.5.1 Developer action elements**
- 4271 **12.7.5.1.1 ALC\_LCD.2.1D**
- 4272 The developer shall establish a life-cycle model to be used in the development and maintenance of the  
4273 **TOE that is based on a measurable life-cycle model.**
- 4274 **12.7.5.1.2 ALC\_LCD.2.2D**
- 4275 The developer shall provide life-cycle definition documentation.
- 4276 **12.7.5.1.3 ALC\_LCD.2.3D**
- 4277 **The developer shall measure the TOE development using the measurable life-cycle model..**
- 4278 **12.7.5.1.4 ALC\_LCD.2.4D**
- 4279 **The developer shall provide life-cycle output documentation.**
- 4280 **12.7.5.2 Content and presentation elements**
- 4281 **12.7.5.2.1 ALC\_LCD.2.1C**
- 4282 The life-cycle definition documentation shall describe the model used to develop and maintain the TOE  
4283 **including the details of its arithmetic parameters and/or metrics used to measure the quality of**  
4284 **the TOE and/or its development.**
- 4285 **12.7.5.2.2 ALC\_LCD.2.2C**
- 4286 The life-cycle model shall provide for the necessary control over the development and maintenance of  
4287 the TOE.
- 4288 **12.7.5.2.3 ALC\_LCD.2.3C**
- 4289 **The life-cycle output documentation shall provide the results of the measurements of the TOE**  
4290 **development using the measurable life-cycle model.**
- 4291 **12.7.5.3 Evaluator action elements**
- 4292 **12.7.5.3.1 ALC\_LCD.2.1E**
- 4293 The evaluator shall confirm that the information provided meets all requirements for content and  
4294 presentation of evidence.

4295 **12.7.5.3.2 ALC\_LCD.2.2E**

4296 **The evaluator shall confirm that the measurements of the TOE development processes and**  
 4297 **security relevant properties of the TOE support improvements in the development processes**  
 4298 **and/or the TOE itself.**

4299 **12.8 TOE Development Artifacts (ALC\_TDA)**4300 **12.8.1 Objectives**

4301 This family aims to add trust to the development process or a development (for brevity). It focuses on  
 4302 the generation of certain artifacts in the development process. These artifacts are used at a later point  
 4303 in time to assess the degree to which the development process or the development is trustable. This  
 4304 trust is realized through the validation of the generated artifacts for confirming them as sufficient  
 4305 evidence for trustable development.

4306 This family introduces developer practices within the development process to generate the required  
 4307 artifacts for realizing trustable development. If a requirement in this family does not explicitly specify  
 4308 the use of automation to generate the required artifacts, the developer is free to undertake the  
 4309 corresponding practice manually, or using some integrated automation in the development process, or  
 4310 using a hybrid method of both. It is expected that the degree of trust in the development process is  
 4311 proportional to the degree of automation adoption to implement the corresponding practice in the  
 4312 development process.

4313 This family also has a closer relationship with ALC\_TAT than most of the other families. As ALC\_TAT  
 4314 focuses on the tools and techniques aspect for developing, analysing, and implementing the TOE, it  
 4315 provides the necessary context when describing the practices of this family being introduced into the  
 4316 development process.

4317 **12.8.2 Component levelling**

4318 The components in this family are levelled on the basis of increasing cross-checking for consistency  
 4319 with relevant evidence from components of other families of other security assurance classes.

4320 **12.8.3 Application notes**

4321 The requirements in ALC\_TDA.1 provide a degree of trust in the developer's ability to identify the set  
 4322 of implementation representation which actually has been used during the TOE generation time. This  
 4323 degree of trust helps to positively answer the question "is that really the source code for this software"  
 4324 or "is that really the register-transfer level (RTL) design or description for this integrated circuit  
 4325 hardware" or "is that really the set of implementation representation for this TOE", which is  
 4326 potentially relevant in an evaluation. Such degree of trust is built on

- 4327 a) the timing of when the set of implementation representation identifiers is recorded or logged,
- 4328 b) the integrity and authenticity of the record of implementation representation identifiers, and
- 4329 c) the traceability of implementation representation identifiers from the TOE.

4330 In the case where some implementation representation elements are also covered in the configuration  
 4331 list due to ALC\_CMS.3, the requirements in ALC\_TDA.2 make sure that these implementation  
 4332 representation elements actually are identifiable through the use of the implementation  
 4333 representation identifiers of ALC\_TDA.1.

4334 With the accurate recording or logging of the actual implementation representation being used by the  
 4335 development tools under the scope of ALC\_TAT, it provides an additional evidence to convince a third  
 4336 party that a regeneration of the TOE is functionally equivalent to the original TOE.

4337 The requirements in ALC\_TDA.3 provide the developer an opportunity to testify the absence of  
 4338 functional differences between the two possibly visibly different TOEs which have been independently  
 4339 generated from the identical set of implementation representation.

#### 4340 **12.8.4 ALC\_TDA.1 Uniquely identifying implementation representation**

4341 Dependencies: No dependencies.

##### 4342 **12.8.4.1 Developer action elements**

###### 4343 **12.8.4.1.1 ALC\_TDA.1.1D**

4344 **The developer shall identify individual elements of the TOE implementation representation to**  
 4345 **record the list of unique TOE implementation representation identifiers, as the development**  
 4346 **tool generates the TOE.**

###### 4347 **12.8.4.1.2 ALC\_TDA.1.2D**

4348 **The developer shall use the current date and time to timestamp the list of unique TOE**  
 4349 **implementation representation identifiers as recorded during the TOE generation time.**

###### 4350 **12.8.4.1.3 ALC\_TDA.1.3D**

4351 **The developer shall maintain the integrity of the list of unique TOE implementation**  
 4352 **representation identifiers as recorded during the TOE generation time.**

###### 4353 **12.8.4.1.4 ALC\_TDA.1.4D**

4354 **The developer shall ensure the authenticity of the list of unique TOE implementation**  
 4355 **representation identifiers as recorded during the TOE generation time, with the maintenance**  
 4356 **of the (author) origination information.**

###### 4357 **12.8.4.1.5 ALC\_TDA.1.5D**

4358 **The developer shall be able to trace from the TOE to the list of unique TOE implementation**  
 4359 **representation identifiers as recorded during the TOE generation time.**

###### 4360 **12.8.4.1.6 ALC\_TDA.1.6D**

4361 **The developer shall produce and provide documentation describing**

4362 **a) the developer's creation of the list of unique TOE implementation representation**  
 4363 **identifiers as recorded during the TOE generation time;**

4364 **b) the developer's timestamp being applied to the list of unique TOE implementation**  
 4365 **representation identifiers as recorded during the TOE generation time;**

4366 **c) the maintenance of the (author) origination information of the list of unique TOE**  
 4367 **implementation representation identifiers as recorded during the TOE generation time;**

4368 d) the maintenance of the integrity of the list of unique TOE implementation  
 4369 representation identifiers as recorded during the TOE generation time and its  
 4370 associated timestamp and (author) origination information;

4371 e) the developer's mechanism to trace from the TOE to the list of unique TOE  
 4372 implementation representation identifiers as recorded during the TOE generation time.

#### 4373 12.8.4.2 Content and presentation elements

##### 4374 12.8.4.2.1 ALC\_TDA.1.1C

4375 The list of unique TOE implementation representation identifiers as recorded during the TOE  
 4376 generation time shall demonstrate the correspondence between the TOE implementation  
 4377 representation element identifiers and the TOE implementation representation element  
 4378 names.

##### 4379 12.8.4.2.2 ALC\_TDA.1.2C

4380 The TOE implementation representation element names shall be in the same form as used or  
 4381 referenced by the development tool to generate the TOE.

##### 4382 12.8.4.2.3 ALC\_TDA.1.3C

4383 The timestamp of the list of unique TOE implementation representation identifiers as recorded  
 4384 during the TOE generation time shall be consistent with the creation time of the TOE.

##### 4385 12.8.4.2.4 ALC\_TDA.1.4C

4386 The (author) origination information of the list of unique TOE implementation representation  
 4387 identifiers as recorded during the TOE generation time shall be consistent with the (author)  
 4388 origination information of the TOE.

#### 4389 12.8.4.3 Evaluator action elements

##### 4390 12.8.4.3.1 ALC\_TDA.1.1E

4391 The evaluator shall confirm that the information provided meets all requirements for content  
 4392 and presentation of evidence.

##### 4393 12.8.4.3.2 ALC\_TDA.1.2E

4394 The evaluator shall confirm that the development tool for generating the TOE is capable to use  
 4395 or reference the implementation representation element names.

##### 4396 12.8.4.3.3 ALC\_TDA.1.3E

4397 The evaluator shall confirm that the list of unique TOE implementation representation  
 4398 identifiers as recorded during the TOE generation time is consistent with the creation time of  
 4399 the TOE.

##### 4400 12.8.4.3.4 ALC\_TDA.1.4E

4401 The evaluator shall confirm that the (author) origination information of the list of unique TOE  
 4402 implementation representation identifiers as recorded during the TOE generation time is  
 4403 consistent with the (author) origination information of the TOE.

4404 **12.8.4.3.5 ALC\_TDA.1.5E**

4405 **The evaluator shall check the integrity of the list of unique TOE implementation representation**  
 4406 **identifiers as recorded during the TOE generation time and its associated timestamp and**  
 4407 **(author) origination information.**

4408 **12.8.4.3.6 ALC\_TDA.1.6E**

4409 **The evaluator shall confirm the developer's ability to trace from the TOE to the list of unique**  
 4410 **TOE implementation representation identifiers as recorded during the TOE generation time.**

4411 **12.8.5 ALC\_TDA.2 Matching CMS scope of implementation representation**

4412 Dependencies: ALC\_CMS.3 Implementation representation CM coverage

4413 **12.8.5.1 Developer action elements**4414 **12.8.5.1.1 ALC\_TDA.2.1D**

4415 The developer shall identify individual elements of the TOE implementation representation to record  
 4416 the list of unique TOE implementation representation identifiers, as the development tool generates  
 4417 the TOE.

4418 **12.8.5.1.2 ALC\_TDA.2.2D**

4419 The developer shall use the current date and time to timestamp the list of unique TOE implementation  
 4420 representation identifiers as recorded during the TOE generation time.

4421 **12.8.5.1.3 ALC\_TDA.2.3D**

4422 The developer shall maintain the integrity of the list of unique TOE implementation representation  
 4423 identifiers as recorded during the TOE generation time.

4424 **12.8.5.1.4 ALC\_TDA.2.4D**

4425 The developer shall ensure the authenticity of the list of unique TOE implementation representation  
 4426 identifiers as recorded during the TOE generation time, with the maintenance of the (author)  
 4427 origination information.

4428 **12.8.5.1.5 ALC\_TDA.2.5D**

4429 The developer shall be able to trace from the TOE to the list of unique TOE implementation  
 4430 representation identifiers as recorded during the TOE generation time.

4431 **12.8.5.1.6 ALC\_TDA.2.6D**

4432 The developer shall produce and provide documentation describing

- 4433 a) the developer's creation of the list of unique TOE implementation representation identifiers as  
 4434 recorded during the TOE generation time;
- 4435 b) the developer's timestamp being applied to the list of unique TOE implementation  
 4436 representation identifiers as recorded during the TOE generation time;
- 4437 c) the maintenance of the (author) origination information of the list of unique TOE  
 4438 implementation representation identifiers as recorded during the TOE generation time;

- 4439 d) the maintenance of the integrity of the list of unique TOE implementation representation  
 4440 identifiers as recorded during the TOE generation time and its associated timestamp and  
 4441 (author) origination information;  
 4442 e) the developer's mechanism to trace from the TOE to the list of unique TOE implementation  
 4443 representation identifiers as recorded during the TOE generation time.

4444 **12.8.5.1.7 ALC\_TDA.2.7D**

4445 **The developer shall provide evidence that the elements of implementation representation**  
 4446 **under the configuration scope of ALC\_CMS.3 are identified by the list of unique TOE**  
 4447 **implementation representation identifiers as recorded during the TOE generation time.**

4448 **12.8.5.2 Content and presentation elements**

4449 **12.8.5.2.1 ALC\_TDA.2.1C**

4450 The list of unique TOE implementation representation identifiers as recorded during the TOE  
 4451 generation time shall demonstrate the correspondence between the TOE implementation  
 4452 representation element identifiers and the TOE implementation representation element names.

4453 **12.8.5.2.2 ALC\_TDA.2.2C**

4454 The TOE implementation representation element names shall be in the same form as used or  
 4455 referenced by the development tool to generate the TOE.

4456 **12.8.5.2.3 ALC\_TDA.2.3C**

4457 The timestamp of the list of unique TOE implementation representation identifiers as recorded during  
 4458 the TOE generation time shall be consistent with the creation time of the TOE.

4459 **12.8.5.2.4 ALC\_TDA.2.4C**

4460 The (author) origination information of the list of unique TOE implementation representation  
 4461 identifiers as recorded during the TOE generation time shall be consistent with the (author)  
 4462 origination information of the TOE.

4463 **12.8.5.2.5 ALC\_TDA.2.5C**

4464 **The list of identifiers of the elements of implementation representation under the**  
 4465 **configuration scope of ALC\_CMS.3 shall match with the list of unique TOE implementation**  
 4466 **representation identifiers as recorded during the TOE generation time.**

4467 **12.8.5.3 Evaluator action elements**

4468 **12.8.5.3.1 ALC\_TDA.2.1E**

4469 The evaluator shall confirm that the information provided meets all requirements for content and  
 4470 presentation of evidence.

4471 **12.8.5.3.2 ALC\_TDA.2.2E**

4472 The evaluator shall confirm that the development tool for generating the TOE is capable to use or  
 4473 reference the implementation representation element names.

4474 **12.8.5.3.3 ALC\_TDA.2.3E**

4475 The evaluator shall confirm that the list of unique TOE implementation representation identifiers as  
4476 recorded during the TOE generation time is consistent with the creation time of the TOE.

4477 **12.8.5.3.4 ALC\_TDA.2.4E**

4478 The evaluator shall confirm that the (author) origination information of the list of unique TOE  
4479 implementation representation identifiers as recorded during the TOE generation time is consistent  
4480 with the (author) origination information of the TOE.

4481 **12.8.5.3.5 ALC\_TDA.2.5E**

4482 The evaluator shall check the integrity of the list of unique TOE implementation representation  
4483 identifiers as recorded during the TOE generation time and its associated timestamp and (author)  
4484 origination information.

4485 **12.8.5.3.6 ALC\_TDA.2.6E**

4486 The evaluator shall confirm the developer's ability to trace from the TOE to the list of unique TOE  
4487 implementation representation identifiers as recorded during the TOE generation time.

4488 **12.8.5.3.7 ALC\_TDA.2.7E**

4489 **The evaluator shall confirm that the list of identifiers of the elements of implementation**  
4490 **representation under the configuration scope of ALC\_CMS.3 matches with the list of unique TOE**  
4491 **implementation representation identifiers as recorded during the TOE generation time.**

4492 **12.8.6 ALC\_TDA.3 Regenerate TOE with well-defined development tools**

4493 Dependencies: ALC\_CMS.3 Implementation representation CM coverage

4494 ALC\_TAT.1 Well-defined development tools and

4495 ADV\_IMP.1 Implementation representation of the TSF

4496 **12.8.6.1 Developer action elements**4497 **12.8.6.1.1 ALC\_TDA.3.1D**

4498 The developer shall identify individual elements of the TOE implementation representation to record  
4499 the list of unique TOE implementation representation identifiers, as the development tool generates  
4500 the TOE.

4501 **12.8.6.1.2 ALC\_TDA.3.2D**

4502 The developer shall use the current date and time to timestamp the list of unique TOE implementation  
4503 representation identifiers as recorded during the TOE generation time.

4504 **12.8.6.1.3 ALC\_TDA.3.3D**

4505 The developer shall maintain the integrity of the list of unique TOE implementation representation  
4506 identifiers as recorded during the TOE generation time.

4507 **12.8.6.1.4 ALC\_TDA.3.4D**

4508 The developer shall ensure the authenticity of the list of unique TOE implementation representation  
 4509 identifiers as recorded during the TOE generation time, with the maintenance of the (author)  
 4510 origination information.

4511 **12.8.6.1.5 ALC\_TDA.3.5D**

4512 The developer shall be able to trace from the TOE to the list of unique TOE implementation  
 4513 representation identifiers as recorded during the TOE generation time.

4514 **12.8.6.1.6 ALC\_TDA.3.6D**

4515 The developer shall produce and provide documentation describing

- 4516 a) the developer's creation of the list of unique TOE implementation representation identifiers as  
 4517 recorded during the TOE generation time;
- 4518 b) the developer's timestamp being applied to the list of unique TOE implementation  
 4519 representation identifiers as recorded during the TOE generation time;
- 4520 c) the maintenance of the (author) origination information of the list of unique TOE  
 4521 implementation representation identifiers as recorded during the TOE generation time;
- 4522 d) the maintenance of the integrity of the list of unique TOE implementation representation  
 4523 identifiers as recorded during the TOE generation time and its associated timestamp and  
 4524 (author) origination information;
- 4525 e) the developer's mechanism to trace from the TOE to the list of unique TOE implementation  
 4526 representation identifiers as recorded during the TOE generation time.

4527 **12.8.6.1.7 ALC\_TDA.3.7D**

4528 The developer shall provide evidence that the elements of implementation representation under the  
 4529 configuration scope of ALC\_CMS.3 are identified by the list of unique TOE implementation  
 4530 representation identifiers as recorded during the TOE generation time.

4531 **12.8.6.1.8 ALC\_TDA.3.8D**

4532 **After applying the development tools to another copy of the TOE implementation**  
 4533 **representation according to the list of unique TOE implementation representation identifiers**  
 4534 **to regenerate a TOE copy, the developer shall explain the functional differences, if any, between**  
 4535 **the TOE copy and the original TOE.**

4536 **12.8.6.1.9 ALC\_TDA.3.9D**

4537 **The developer shall produce and provide documentation explaining the functional differences,**  
 4538 **if any, between the regenerated TOE copy and the original TOE.**

4539 **12.8.6.2 Content and presentation elements**4540 **12.8.6.2.1 ALC\_TDA.3.1C**

4541 The list of unique TOE implementation representation identifiers as recorded during the TOE  
 4542 generation time shall demonstrate the correspondence between the TOE implementation  
 4543 representation element identifiers and the TOE implementation representation element names.

4544 **12.8.6.2.2 ALC\_TDA.3.2C**

4545 The TOE implementation representation element names shall be in the same form as used or  
4546 referenced by the development tool to generate the TOE.

4547 **12.8.6.2.3 ALC\_TDA.3.3C**

4548 The timestamp of the list of unique TOE implementation representation identifiers as recorded during  
4549 the TOE generation time shall be consistent with the creation time of the TOE.

4550 **12.8.6.2.4 ALC\_TDA.3.4C**

4551 The (author) origination information of the list of unique TOE implementation representation  
4552 identifiers as recorded during the TOE generation time shall be consistent with the (author)  
4553 origination information of the TOE.

4554 **12.8.6.2.5 ALC\_TDA.3.5C**

4555 The list of identifiers of the elements of implementation representation under the configuration scope  
4556 of ALC\_CMS.3 shall match with the list of unique TOE implementation representation identifiers as  
4557 recorded during the TOE generation time.

4558 **12.8.6.2.6 ALC\_TDA.3.6C**

4559 **The developer's explanation of the functional differences, if any, between the regenerated TOE**  
4560 **copy and the original TOE shall take into account all visible differences, if any, between the**  
4561 **regenerated TOE copy and the original TOE.**

4562 **12.8.6.3 Evaluator action elements**

4563 **12.8.6.3.1 ALC\_TDA.3.1E**

4564 The evaluator shall confirm that the information provided meets all requirements for content and  
4565 presentation of evidence.

4566 **12.8.6.3.2 ALC\_TDA.3.2E**

4567 The evaluator shall confirm that the development tool for generating the TOE is capable to use or  
4568 reference the implementation representation element names.

4569 **12.8.6.3.3 ALC\_TDA.3.3E**

4570 The evaluator shall confirm that the list of unique TOE implementation representation identifiers as  
4571 recorded during the TOE generation time is consistent with the creation time of the TOE.

4572 **12.8.6.3.4 ALC\_TDA.3.4E**

4573 The evaluator shall confirm that the (author) origination information of the list of unique TOE  
4574 implementation representation identifiers as recorded during the TOE generation time is consistent  
4575 with the (author) origination information of the TOE.

4576 **12.8.6.3.5 ALC\_TDA.3.5E**

4577 The evaluator shall check the integrity of the list of unique TOE implementation representation  
4578 identifiers as recorded during the TOE generation time and its associated timestamp and (author)  
4579 origination information.

4580 **12.8.6.3.6 ALC\_TDA.3.6E**

4581 The evaluator shall confirm the developer's ability to trace from the TOE to the list of unique TOE  
4582 implementation representation identifiers as recorded during the TOE generation time.

4583 **12.8.6.3.7 ALC\_TDA.3.7E**

4584 The evaluator shall confirm that the list of identifiers of the elements of implementation  
4585 representation under the configuration scope of ALC\_CMS.3 matches with the list of unique TOE  
4586 implementation representation identifiers as recorded during the TOE generation time.

4587 **12.8.6.3.8 ALC\_TDA.3.8E**

4588 **The evaluator shall check that the developer's explanation of the functional differences, if any,**  
4589 **between the regenerated TOE copy and the original TOE takes into account all visible**  
4590 **differences, if any, between the regenerated TOE copy and the original TOE.**

4591 **12.9 Tools and techniques (ALC\_TAT)**4592 **12.9.1 Objectives**

4593 Tools and techniques is an aspect of selecting tools that are used to develop, analyse and implement  
4594 the TOE. It includes requirements to prevent ill-defined, inconsistent or incorrect development tools  
4595 from being used to develop the TOE. This includes, but is not limited to, programming languages,  
4596 documentation, implementation standards, and other parts of the TOE such as supporting runtime  
4597 libraries.

4598 **12.9.2 Component levelling**

4599 The components in this family are levelled on the basis of increasing requirements on the description  
4600 and scope of the implementation standards and the documentation of implementation-dependent  
4601 options.

4602 **12.9.3 Application notes**

4603 There is a requirement for well-defined development tools. These are tools that are clearly and  
4604 completely described. For example, programming languages and computer aided design (CAD)  
4605 systems that are based on a standard published by standards bodies are considered to be well-defined.  
4606 Self-made tools would need further investigation to clarify whether they are well-defined.

4607 The requirement in ALC\_TAT.1.2C is especially applicable to programming languages so as to ensure  
4608 that all statements in the source code have an unambiguous meaning.

4609 In ALC\_TAT.2 and ALC\_TAT.3, implementation guidelines may be accepted as an implementation  
4610 standard if they have been approved by some group of experts (e.g. academic experts, standards  
4611 bodies). Implementation standards are normally public, well accepted and common practise in a  
4612 specific industry, but developer-specific implementation guidelines may also be accepted as a  
4613 standard; the emphasis is on the expertise.

4614 Tools and techniques distinguishes between the implementation standards applied by the developer  
 4615 (ALC\_TAT.2.3D) and the implementation standards for “all parts of the TOE” (ALC\_TAT.3.3D) which  
 4616 include third party software, hardware, or firmware. The configuration list introduced in CM scope  
 4617 (ALC\_CMS) requires that for each TSF relevant configuration item to indicate if it has been generated  
 4618 by the TOE developer or by third party developers

#### 4619 **12.9.4 ALC\_TAT.1 Well-defined development tools**

4620 Dependencies: ADV\_IMP.1 Implementation representation of the TSF

##### 4621 **12.9.4.1 Developer action elements**

###### 4622 **12.9.4.1.1 ALC\_TAT.1.1D**

4623 **The developer shall provide the documentation identifying each development tool being used**  
 4624 **for the TOE.**

###### 4625 **12.9.4.1.2 ALC\_TAT.1.2D**

4626 **The developer shall document and provide the selected implementation-dependent options of**  
 4627 **each development tool.**

##### 4628 **12.9.4.2 Content and presentation elements**

###### 4629 **12.9.4.2.1 ALC\_TAT.1.1C**

4630 **Each development tool used for implementation shall be well-defined.**

###### 4631 **12.9.4.2.2 ALC\_TAT.1.2C**

4632 **The documentation of each development tool shall unambiguously define the meaning of all**  
 4633 **statements as well as all conventions and directives used in the implementation.**

###### 4634 **12.9.4.2.3 ALC\_TAT.1.3C**

4635 **The documentation of each development tool shall unambiguously define the meaning of all**  
 4636 **implementation-dependent options.**

##### 4637 **12.9.4.3 Evaluator action elements**

###### 4638 **12.9.4.3.1 ALC\_TAT.1.1E**

4639 **The evaluator shall confirm that the information provided meets all requirements for content**  
 4640 **and presentation of evidence.**

#### 4641 **12.9.5 ALC\_TAT.2 Compliance with implementation standards**

4642 Dependencies: ADV\_IMP.1 Implementation representation of the TSF

##### 4643 **12.9.5.1 Developer action elements**

###### 4644 **12.9.5.1.1 ALC\_TAT.2.1D**

4645 The developer shall provide the documentation identifying each development tool being used for the  
 4646 TOE.

4647 **12.9.5.1.2 ALC\_TAT.2.2D**

4648 The developer shall document and provide the selected implementation-dependent options of each  
4649 development tool.

4650 **12.9.5.1.3 ALC\_TAT.2.3D**

4651 **The developer shall describe and provide the implementation standards that are being applied**  
4652 **by the developer.**

4653 **12.9.5.2 Content and presentation elements**4654 **12.9.5.2.1 ALC\_TAT.2.1C**

4655 Each development tool used for implementation shall be well-defined.

4656 **12.9.5.2.2 ALC\_TAT.2.2C**

4657 The documentation of each development tool shall unambiguously define the meaning of all  
4658 statements as well as all conventions and directives used in the implementation.

4659 **12.9.5.2.3 ALC\_TAT.2.3C**

4660 The documentation of each development tool shall unambiguously define the meaning of all  
4661 implementation-dependent options.

4662 **12.9.5.3 Evaluator action elements**4663 **12.9.5.3.1 ALC\_TAT.2.1E**

4664 The evaluator shall confirm that the information provided meets all requirements for content and  
4665 presentation of evidence.

4666 **12.9.5.3.2 ALC\_TAT.2.2E**

4667 **The evaluator shall confirm that the implementation standards have been applied.**

4668 **12.9.6 ALC\_TAT.3 Compliance with implementation standards - all parts**

4669 Dependencies: ADV\_IMP.1 Implementation representation of the TSF

4670 **12.9.6.1 Developer action elements**4671 **12.9.6.1.1 ALC\_TAT.3.1D**

4672 The developer shall provide the documentation identifying each development tool being used for the  
4673 TOE.

4674 **12.9.6.1.2 ALC\_TAT.3.2D**

4675 The developer shall document and provide the selected implementation-dependent options of each  
4676 development tool.

4677 **12.9.6.1.3 ALC\_TAT.3.3D**

4678 The developer shall describe and provide the implementation standards that are being applied by the  
4679 developer **and by any third-party providers for all parts of the TOE.**

4680 **12.9.6.2 Content and presentation elements**4681 **12.9.6.2.1 ALC\_TAT.3.1C**

4682 Each development tool used for implementation shall be well-defined.

4683 **12.9.6.2.2 ALC\_TAT.3.2C**

4684 The documentation of each development tool shall unambiguously define the meaning of all  
4685 statements as well as all conventions and directives used in the implementation.

4686 **12.9.6.2.3 ALC\_TAT.3.3C**

4687 The documentation of each development tool shall unambiguously define the meaning of all  
4688 implementation-dependent options.

4689 **12.9.6.3 Evaluator action elements**4690 **12.9.6.3.1 ALC\_TAT.3.1E**

4691 The evaluator shall confirm that the information provided meets all requirements for content and  
4692 presentation of evidence.

4693 **12.9.6.3.2 ALC\_TAT.3.2E**

4694 **The evaluator shall confirm that the implementation standards have been applied.**

4695 **12.10 Integration of composition parts and consistency check of delivery procedures**  
4696 **(ALC\_COMP)**

4697 **12.10.1 Objectives**

4698 The aims of this activity are to determine whether

- 4699 - the correct version of the application is installed onto/into the correct version of the  
4700 underlying platform, and
- 4701 - the preparative guidance procedures of Platform and Application Developers are compatible  
4702 with the acceptance procedure of the Composite Product Integrator.

4703 **12.10.2 Component levelling**

4704 This family contains only one component.

4705 **12.10.3 ALC\_COMP.1 Integration of the application into the underlying platform and**  
4706 **Consistency check for delivery and acceptance procedures**

4707 Dependencies: No dependencies

4708 **12.10.3.1 Developer action elements**4709 **12.10.3.1.1 ALC\_COMP.1.1D**

4710 The developer shall provide components configuration evidence; cf. item #7, item #8 and item #3 in  
4711 Table D1, section Annex A.

4712 **12.10.3.2 Content and presentation elements**4713 **12.10.3.2.1 ALC\_COMP.1.1C**

4714 The components configuration evidence shall show that the evaluated version of the application has  
4715 been installed onto / embedded into the certified version of the underlying platform.

4716 **12.10.3.2.2 ALC\_COMP.1.2C**

4717 The components configuration evidence shall show that:

4718 i. The evidence for delivery and acceptance compatibility shall show that the delivery procedures of  
4719 the Platform and Application Developers are compatible with the acceptance procedure of the  
4720 Composite Product Integrator.

4721 ii. the evidence shall show that preparative guidance procedures prescribed by the Platform and  
4722 Application Developers are either actually being used by the Composite Product Integrator or  
4723 compatible with the Composite Product Integrator guidance and do not contradict each other.

4724 **12.10.3.3 Evaluator action elements**4725 **12.10.3.3.1 ALC\_COMP.1.1E**

4726 The evaluator shall confirm that the information provided meets all requirements for content and  
4727 presentation of evidence.

4728 **12.10.3.3.2 ALC\_COMP.1.2E**

4729 The evaluator shall confirm that the evidence for delivery compatibility is complete, coherent, and  
4730 internally consistent.

4731 **13 Class ATE: Tests**4732 **13.1 Introduction**

4733 The class "Tests" encompasses five families: Coverage (ATE\_COV), Depth (ATE\_DPT), Independent  
4734 testing (ATE\_IND) (i.e. functional testing performed by evaluators), Functional tests (ATE\_FUN) and  
4735 Composite functional testing (ATE\_COMP). Testing provides assurance that the TSF behaves as  
4736 described (in the functional specification, TOE design, implementation representation, and allows  
4737 straightforward traceability of SFR in test scenario).

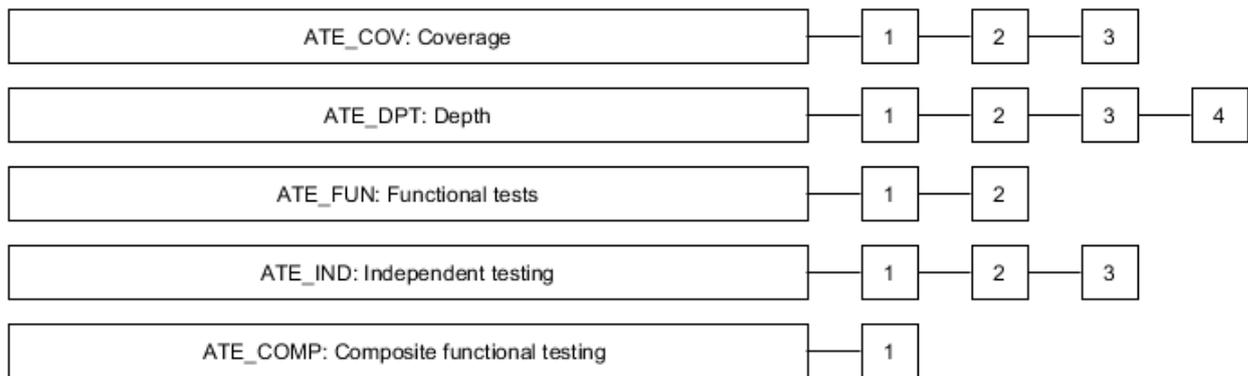
4738 The emphasis in this class is on confirmation that the TSF operates according to its design  
4739 descriptions. This class does not address penetration testing, which is based upon an analysis of the  
4740 TSF that specifically seeks to identify vulnerabilities in the design and implementation of the TSF.  
4741 Penetration testing is addressed separately as an aspect of vulnerability assessment in the AVA:  
4742 Vulnerability assessment class.

4743 The ATE: Tests class separates testing into developer testing and evaluator testing. The Coverage  
 4744 (ATE\_COV), and Depth (ATE\_DPT) families address the completeness of developer testing. Coverage  
 4745 (ATE\_COV) addresses the rigour with which the functional specification is tested; Depth (ATE\_DPT)  
 4746 addresses whether testing against other design descriptions (security architecture, TOE design, and  
 4747 implementation representation) is required.

4748 Functional tests (ATE\_FUN) addresses the performing of the tests by the developer and how this  
 4749 testing should be documented. Finally, Independent testing (ATE\_IND) then addresses evaluator  
 4750 testing: whether the evaluator should repeat part or all of the developer testing and how much  
 4751 independent testing the evaluator should do.

4752 Composite functional testing (ATE\_COMP) determines whether composite product as a whole exhibits  
 4753 the properties necessary to satisfy the functional requirements of its Security Target.

4754 Figure 12 shows the families within this class, and the hierarchy of components within the families.



4755

4756 **Figure 12 — ATE: Tests class decomposition**

4757 **13.2 Coverage (ATE\_COV)**

4758 **13.2.1 Objectives**

4759 This family establishes that the TSF has been tested against its functional specification. This is  
 4760 achieved through an examination of developer evidence of correspondence.

4761 **13.2.2 Component levelling**

4762 The components in this family are levelled on the basis of specification.

4763 **13.2.3 Application notes**

4764 **13.2.4 ATE\_COV.1 Evidence of coverage**

4765 Dependencies: ADV\_FSP.2 Security-enforcing functional specification

4766 ATE\_FUN.1 Functional testing

4767 **13.2.4.1 Objectives**

4768 The objective of this component is to establish that some of the TSFIs have been tested.

4769 **13.2.4.2 Application notes**

4770 In this component the developer shows how tests in the test documentation correspond to TSFIs in the  
4771 functional specification. This can be achieved by a statement of correspondence, perhaps using a table.

4772 **13.2.4.3 Developer action elements**4773 **13.2.4.3.1 ATE\_COV.1.1D**

4774 **The developer shall provide evidence of the test coverage.**

4775 **13.2.4.4 Content and presentation elements**4776 **13.2.4.4.1 ATE\_COV.1.1C**

4777 **The evidence of the test coverage shall show the correspondence between the tests in the test**  
4778 **documentation and the TSFIs in the functional specification.**

4779 **13.2.4.5 Evaluator action elements**4780 **13.2.4.5.1 ATE\_COV.1.1E**

4781 **The evaluator shall confirm that the information provided meets all requirements for content**  
4782 **and presentation of evidence.**

4783 **13.2.5 ATE\_COV.2 Analysis of coverage**

4784 Dependencies: ADV\_FSP.2 Security-enforcing functional specification

4785 ATE\_FUN.1 Functional testing

4786 **13.2.5.1 Objectives**

4787 The objective of this component is to confirm that all of the TSFIs have been tested.

4788 **13.2.5.2 Application notes**

4789 In this component the developer confirms that tests in the test documentation correspond to all of the  
4790 TSFIs in the functional specification. This can be achieved by a statement of correspondence, perhaps  
4791 using a table, but the developer also provides an analysis of the test coverage.

4792 **13.2.5.3 Developer action elements**4793 **13.2.5.3.1 ATE\_COV.2.1D**

4794 The developer shall provide **an analysis** of the test coverage.

4795 **13.2.5.4 Content and presentation elements**4796 **13.2.5.4.1 ATE\_COV.2.1C**

4797 The **analysis** of the test coverage shall **demonstrate** the correspondence between the tests in the test  
4798 documentation and the TSFIs in the functional specification.

4799 **13.2.5.4.2 ATE\_COV.2.2C**

4800 **The analysis of the test coverage shall demonstrate that all TSFIs in the functional specification**  
 4801 **have been tested.**

4802 **13.2.5.5 Evaluator action elements**4803 **13.2.5.5.1 ATE\_COV.2.1E**

4804 The evaluator shall confirm that the information provided meets all requirements for content and  
 4805 presentation of evidence.

4806 **13.2.6 ATE\_COV.3 Rigorous analysis of coverage**

4807 Dependencies: ADV\_FSP.2 Security-enforcing functional specification

4808 ATE\_FUN.1 Functional testing

4809 **13.2.6.1 Objectives**

4810 In this component, the objective is to confirm that the developer performed exhaustive tests of all  
 4811 interfaces in the functional specification.

4812 The objective of this component is to confirm that all parameters of all of the TSFIs have been tested.

4813 **13.2.6.2 Application notes**

4814 In this component the developer is required to show how tests in the test documentation correspond  
 4815 to all of the TSFIs in the functional specification. This can be achieved by a statement of  
 4816 correspondence, perhaps using a table, but in addition the developer is required to demonstrate that  
 4817 the tests exercise all of the parameters of all TSFIs. This additional requirement includes bounds  
 4818 testing (i.e. verifying that errors are generated when stated limits are exceeded) and negative testing  
 4819 (e.g. when access is given to User A, verifying not only that User A now has access, but also that User B  
 4820 did not suddenly gain access). This kind of testing is not, strictly speaking, *exhaustive* because not  
 4821 every possible value of the parameters is expected to be checked.

4822 **13.2.6.3 Developer action elements**4823 **13.2.6.3.1 ATE\_COV.3.1D**

4824 The developer shall provide an analysis of the test coverage.

4825 **13.2.6.4 Content and presentation elements**4826 **13.2.6.4.1 ATE\_COV.3.1C**

4827 The analysis of the test coverage shall demonstrate the correspondence between the tests in the test  
 4828 documentation and the TSFIs in the functional specification.

4829 **13.2.6.4.2 ATE\_COV.3.2C**

4830 The analysis of the test coverage shall demonstrate that all TSFIs in the functional specification have  
 4831 been **completely** tested.

4832 **13.2.6.5 Evaluator action elements**4833 **13.2.6.5.1 ATE\_COV.3.1E**

4834 The evaluator shall confirm that the information provided meets all requirements for content and  
4835 presentation of evidence.

4836 **13.3 Depth (ATE\_DPT)**4837 **13.3.1 Objectives**

4838 The components in this family deal with the level of detail to which the TSF is tested by the developer.  
4839 Testing of the TSF is based upon increasing depth of information derived from additional design  
4840 representations and descriptions (TOE design, implementation representation, and security  
4841 architecture description).

4842 The objective is to counter the risk of missing an error in the development of the TOE. Testing that  
4843 exercises specific internal interfaces can provide assurance not only that the TSF exhibits the desired  
4844 external security behaviour, but also that this behaviour stems from correctly operating internal  
4845 functionality.

4846 **13.3.2 Component levelling**

4847 The components in this family are levelled on the basis of increasing detail provided in the TSF  
4848 representations, from the TOE design to the implementation representation. This levelling reflects the  
4849 TSF representations presented in the ADV class.

4850 **13.3.3 Application notes**

4851 The TOE design describes the internal components (e.g. subsystems) and, perhaps, modules of the TSF,  
4852 together with a description of the interfaces among these components and modules. Evidence of  
4853 testing of this TOE design must show that the internal interfaces have been exercised and seen to  
4854 behave as described. This may be achieved through testing via the external interfaces of the TSF, or by  
4855 testing of the TOE subsystem or module interfaces in isolation, perhaps employing a test harness. In  
4856 cases where some aspects of an internal interface cannot be tested via the external interfaces, there  
4857 should either be justification that these aspects need not be tested, or the internal interface needs to  
4858 be tested directly. In the latter case the TOE design needs to be sufficiently detailed in order to  
4859 facilitate direct testing.

4860 In cases where the description of the TSF's architectural soundness (in Security Architecture  
4861 (ADV\_ARC)) cites specific mechanisms, the tests performed by the developer must show that the  
4862 mechanisms have been exercised and seen to behave as described.

4863 At the highest component of this family, the testing is performed not only against the TOE design, but  
4864 also against the implementation representation.

4865 **13.3.4 ATE\_DPT.1 Testing: basic design**

4866 Dependencies: ADV\_ARC.1 Security architecture description

4867 ADV\_TDS.2 Architectural design

4868 ATE\_FUN.1 Functional testing

4869 **13.3.4.1 Objectives**

4870 The subsystem descriptions of the TSF provide a high-level description of the internal workings of the  
4871 TSF. Testing at the level of the TOE subsystems provides assurance that the TSF subsystems behave  
4872 and interact as described in the TOE design and the security architecture description.

4873 **13.3.4.2 Developer action elements**

4874 **13.3.4.2.1 ATE\_DPT.1.1D**

4875 **The developer shall provide the analysis of the depth of testing.**

4876 **13.3.4.3 Content and presentation elements**

4877 **13.3.4.3.1 ATE\_DPT.1.1C**

4878 **The analysis of the depth of testing shall demonstrate the correspondence between the tests in**  
4879 **the test documentation and the TSF subsystems in the TOE design.**

4880 **13.3.4.3.2 ATE\_DPT.1.2C**

4881 **The analysis of the depth of testing shall demonstrate that all TSF subsystems in the TOE design**  
4882 **have been tested.**

4883 **13.3.4.4 Evaluator action elements**

4884 **13.3.4.4.1 ATE\_DPT.1.1E**

4885 **The evaluator shall confirm that the information provided meets all requirements for content**  
4886 **and presentation of evidence.**

4887 **13.3.5 ATE\_DPT.2 Testing: security enforcing modules**

4888 Dependencies: ADV\_ARC.1 Security architecture description

4889 ADV\_TDS.3 Basic modular design

4890 ATE\_FUN.1 Functional testing

4891 **13.3.5.1 Objectives**

4892 The subsystem and module descriptions of the TSF provide a high-level description of the internal  
4893 workings, and a description of the interfaces of the SFR-enforcing modules, of the TSF. Testing at this  
4894 level of TOE description provides assurance that the TSF subsystems and SFR-enforcing modules  
4895 behave and interact as described in the TOE design and the security architecture description.

4896 **13.3.5.2 Developer action elements**

4897 **13.3.5.2.1 ATE\_DPT.2.1D**

4898 The developer shall provide the analysis of the depth of testing.

- 4899 **13.3.5.3 Content and presentation elements**
- 4900 **13.3.5.3.1 ATE\_DPT.2.1C**
- 4901 The analysis of the depth of testing shall demonstrate the correspondence between the tests in the test  
4902 documentation and the TSF subsystems **and SFR-enforcing modules** in the TOE design.
- 4903 **13.3.5.3.2 ATE\_DPT.2.2C**
- 4904 The analysis of the depth of testing shall demonstrate that all TSF subsystems in the TOE design have  
4905 been tested.
- 4906 **13.3.5.3.3 ATE\_DPT.2.3C**
- 4907 **The analysis of the depth of testing shall demonstrate that the SFR-enforcing modules in the**  
4908 **TOE design have been tested.**
- 4909 **13.3.5.4 Evaluator action elements**
- 4910 **13.3.5.4.1 ATE\_DPT.2.1E**
- 4911 The evaluator shall confirm that the information provided meets all requirements for content and  
4912 presentation of evidence.
- 4913 **13.3.6 ATE\_DPT.3 Testing: modular design**
- 4914 Dependencies: ADV\_ARC.1 Security architecture description
- 4915 ADV\_TDS.4 Semiformal modular design
- 4916 ATE\_FUN.1 Functional testing
- 4917 **13.3.6.1 Objectives**
- 4918 The subsystem and module descriptions of the TSF provide a high-level description of the internal  
4919 workings, and a description of the interfaces of the modules, of the TSF. Testing at this level of TOE  
4920 description provides assurance that the TSF subsystems and modules behave and interact as  
4921 described in the TOE design and the security architecture description.
- 4922 **13.3.6.2 Developer action elements**
- 4923 **13.3.6.2.1 ATE\_DPT.3.1D**
- 4924 The developer shall provide the analysis of the depth of testing.
- 4925 **13.3.6.3 Content and presentation elements**
- 4926 **13.3.6.3.1 ATE\_DPT.3.1C**
- 4927 The analysis of the depth of testing shall demonstrate the correspondence between the tests in the test  
4928 documentation and the TSF subsystems and modules in the TOE design.
- 4929 **13.3.6.3.2 ATE\_DPT.3.2C**
- 4930 The analysis of the depth of testing shall demonstrate that all TSF subsystems in the TOE design have  
4931 been tested.

4932 **13.3.6.3.3 ATE\_DPT.3.3C**

4933 The analysis of the depth of testing shall demonstrate that **all TSF** modules in the TOE design have  
4934 been tested.

4935 **13.3.6.4 Evaluator action elements**4936 **13.3.6.4.1 ATE\_DPT.3.1E**

4937 The evaluator shall confirm that the information provided meets all requirements for content and  
4938 presentation of evidence.

4939 **13.3.7 ATE\_DPT.4 Testing: implementation representation**

4940 Dependencies: ADV\_ARC.1 Security architecture description

4941 ADV\_TDS.4 Semiformal modular design

4942 ADV\_IMP.1 Implementation representation of the TSF

4943 ATE\_FUN.1 Functional testing

4944 **13.3.7.1 Objectives**

4945 The subsystem and module descriptions of the TSF provide a high-level description of the internal  
4946 workings, and a description of the interfaces of the modules, of the TSF. Testing at this level of TOE  
4947 description provides assurance that the TSF subsystems and modules behave and interact as  
4948 described in the TOE design and the security architecture description, and in accordance with the  
4949 implementation representation.

4950 **13.3.7.2 Developer action elements**4951 **13.3.7.2.1 ATE\_DPT.4.1D**

4952 The developer shall provide the analysis of the depth of testing.

4953 **13.3.7.3 Content and presentation elements**4954 **13.3.7.3.1 ATE\_DPT.4.1C**

4955 The analysis of the depth of testing shall demonstrate the correspondence between the tests in the test  
4956 documentation and the TSF subsystems and modules in the TOE design.

4957 **13.3.7.3.2 ATE\_DPT.4.2C**

4958 The analysis of the depth of testing shall demonstrate that all TSF subsystems in the TOE design have  
4959 been tested.

4960 **13.3.7.3.3 ATE\_DPT.4.3C**

4961 The analysis of the depth of testing shall demonstrate that all modules in the TOE design have been  
4962 tested.

4963 **13.3.7.3.4 ATE\_DPT.4.4C**

4964 **The analysis of the depth of testing shall demonstrate that the TSF operates in accordance with**  
 4965 **its implementation representation.**

4966 **13.3.7.4 Evaluator action elements**4967 **13.3.7.4.1 ATE\_DPT.4.1E**

4968 The evaluator shall confirm that the information provided meets all requirements for content and  
 4969 presentation of evidence.

4970 **13.4 Functional tests (ATE\_FUN)**4971 **13.4.1 Objectives**

4972 Functional testing performed by the developer provides assurance that the tests in the test  
 4973 documentation are performed and documented correctly. The correspondence of these tests to the  
 4974 design descriptions of the TSF is achieved through the Coverage (ATE\_COV) and Depth (ATE\_DPT)  
 4975 families.

4976 This family contributes to providing assurance that the likelihood of undiscovered flaws is relatively  
 4977 small.

4978 The families Coverage (ATE\_COV), Depth (ATE\_DPT) and Functional tests (ATE\_FUN) are used in  
 4979 combination to define the evidence of testing to be supplied by a developer. Independent functional  
 4980 testing by the evaluator is specified by Independent testing (ATE\_IND).

4981 **13.4.2 Component levelling**

4982 This family contains two components, the higher requiring that ordering dependencies are analysed.

4983 **13.4.3 Application notes**

4984 Procedures for performing tests are expected to provide instructions for using test programs and test  
 4985 suites, including the test environment, test conditions, test data parameters and values. The test  
 4986 procedures should also show how the test results are derived from the test inputs.

4987 Ordering dependencies are relevant when the successful execution of a particular test depends upon  
 4988 the existence of a particular state. For example, this might require that test A be executed immediately  
 4989 before test B, since the state resulting from the successful execution of test A is a prerequisite for the  
 4990 successful execution of test B. Thus, failure of test B could be related to a problem with the ordering  
 4991 dependencies. In the above example, test B could fail because test C (rather than test A) was executed  
 4992 immediately before it, or the failure of test B could be related to a failure of test A.

4993 **13.4.4 ATE\_FUN.1 Functional testing**

4994 Dependencies: ATE\_COV.1 Evidence of coverage

4995 **13.4.4.1 Objectives**

4996 The objective is for the developer to demonstrate that the tests in the test documentation are  
 4997 performed and documented correctly.

- 4998 **13.4.4.2 Developer action elements**
- 4999 **13.4.4.2.1 ATE\_FUN.1.1D**
- 5000 **The developer shall test the TSF and document the results.**
- 5001 **13.4.4.2.2 ATE\_FUN.1.2D**
- 5002 **The developer shall provide test documentation.**
- 5003 **13.4.4.3 Content and presentation elements**
- 5004 **13.4.4.3.1 ATE\_FUN.1.1C**
- 5005 **The test documentation shall consist of test plans, expected test results and actual test results.**
- 5006 **13.4.4.3.2 ATE\_FUN.1.2C**
- 5007 **The test plans shall identify the tests to be performed and describe the scenarios for**  
 5008 **performing each test. These scenarios shall include any ordering dependencies on the results**  
 5009 **of other tests.**
- 5010 **13.4.4.3.3 ATE\_FUN.1.3C**
- 5011 **The expected test results shall show the anticipated outputs from a successful execution of the**  
 5012 **tests.**
- 5013 **13.4.4.3.4 ATE\_FUN.1.4C**
- 5014 **The actual test results shall be consistent with the expected test results.**
- 5015 **13.4.4.4 Evaluator action elements**
- 5016 **13.4.4.4.1 ATE\_FUN.1.1E**
- 5017 **The evaluator shall confirm that the information provided meets all requirements for content**  
 5018 **and presentation of evidence.**
- 5019 **13.4.5 ATE\_FUN.2 Ordered functional testing**
- 5020 Dependencies: ATE\_COV.1 Evidence of coverage
- 5021 **13.4.5.1 Objectives**
- 5022 The objectives are for the developer to demonstrate that the tests in the test documentation are  
 5023 performed and documented correctly, and to ensure that testing is structured such as to avoid circular  
 5024 arguments about the correctness of the interfaces being tested.
- 5025 **13.4.5.2 Application notes**
- 5026 Although the test procedures may state pre-requisite initial test conditions in terms of ordering of  
 5027 tests, they may not provide a rationale for the ordering. An analysis of test ordering is an important  
 5028 factor in determining the adequacy of testing, as there is a possibility of faults being concealed by the  
 5029 ordering of tests.

5030 **13.4.5.3 Developer action elements**

5031 **13.4.5.3.1 ATE\_FUN.2.1D**

5032 The developer shall test the TSF and document the results.

5033 **13.4.5.3.2 ATE\_FUN.2.2D**

5034 The developer shall provide test documentation.

5035 **13.4.5.4 Content and presentation elements**

5036 **13.4.5.4.1 ATE\_FUN.2.1C**

5037 The test documentation shall consist of test plans, expected test results and actual test results.

5038 **13.4.5.4.2 ATE\_FUN.2.2C**

5039 The test plans shall identify the tests to be performed and describe the scenarios for performing each  
5040 test. These scenarios shall include any ordering dependencies on the results of other tests.

5041 **13.4.5.4.3 ATE\_FUN.2.3C**

5042 The expected test results shall show the anticipated outputs from a successful execution of the tests.

5043 **13.4.5.4.4 ATE\_FUN.2.4C**

5044 The actual test results shall be consistent with the expected test results.

5045 **13.4.5.4.5 ATE\_FUN.2.5C**

5046 **The test documentation shall include an analysis of the test procedure ordering dependencies.**

5047 **13.4.5.5 Evaluator action elements**

5048 **13.4.5.5.1 ATE\_FUN.2.1E**

5049 The evaluator shall confirm that the information provided meets all requirements for content and  
5050 presentation of evidence.

5051 **13.5 Independent testing (ATE\_IND)**

5052 **13.5.1 Objectives**

5053 The objectives of this family are built upon the assurances achieved in the ATE\_FUN, ATE\_COV, and  
5054 ATE\_DPT families by verifying the developer testing and performing additional tests by the evaluator.

5055 **13.5.2 Component levelling**

5056 Levelling is based upon the amount of developer test documentation and test support and the amount  
5057 of evaluator testing.

5058 **13.5.3 Application notes**

5059 This family deals with the degree to which there is independent functional testing of the TSF.  
5060 Independent functional testing may take the form of repeating the developer's functional tests (in

5061 whole or in part) or of extending the scope or the depth of the developer's tests. These activities are  
 5062 complementary, and an appropriate mix must be planned for each TOE, which takes into account the  
 5063 availability and coverage of test results, and the functional complexity of the TSF.

5064 Sampling of developer tests is intended to provide confirmation that the developer has carried out his  
 5065 planned test programme on the TSF, and has correctly recorded the results. The size of sample  
 5066 selected will be influenced by the detail and quality of the developer's functional test results. The  
 5067 evaluator will also need to consider the scope for devising additional tests, and the relative benefit that  
 5068 may be gained from effort in these two areas. It is recognised that repetition of all developer tests may  
 5069 be feasible and desirable in some cases, but may be very arduous and less productive in others. The  
 5070 highest component in this family should therefore be used with caution. Sampling will address the  
 5071 whole range of test results available, including those supplied to meet the requirements of both  
 5072 Coverage (ATE\_COV) and Depth (ATE\_DPT).

5073 There is also a need to consider the different configurations of the TOE that are included within the  
 5074 evaluation. The evaluator will need to assess the applicability of the results provided, and to plan his  
 5075 own testing accordingly.

5076 The suitability of the TOE for testing is based on the access to the TOE, and the supporting  
 5077 documentation and information required (including any test software or tools) to run tests. The need  
 5078 for such support is addressed by the dependencies to other assurance families.

5079 Additionally, suitability of the TOE for testing may be based on other considerations. For example, the  
 5080 version of the TOE submitted by the developer may not be the final version.

5081 The term *interfaces* refers to interfaces described in the functional specification and TOE design, and  
 5082 parameters passed through invocations identified in the implementation representation. The exact set  
 5083 of interfaces to be used is selected through Coverage (ATE\_COV) and the Depth (ATE\_DPT)  
 5084 components.

5085 References to a subset of the interfaces are intended to allow the evaluator to design an appropriate  
 5086 set of tests which is consistent with the objectives of the evaluation being conducted.

#### 5087 **13.5.4 ATE\_IND.1 Independent testing - conformance**

5088 Dependencies: ADV\_FSP.1 Basic functional specification

5089 AGD\_OPE.1 Operational user guidance

5090 AGD\_PRE.1 Preparative procedures

##### 5091 **13.5.4.1 Objectives**

5092 In this component, the objective is to demonstrate that the TOE operates in accordance with its design  
 5093 representations and guidance documents.

##### 5094 **13.5.4.2 Application notes**

5095 This component does not address the use of developer test results. It is applicable where such results  
 5096 are not available, and also in cases where the developer's testing is accepted without validation. The  
 5097 evaluator is required to devise and conduct tests with the objective of confirming that the TOE  
 5098 operates in accordance with its design representations, including but not limited to the functional  
 5099 specification. The approach is to gain confidence in correct operation through representative testing,  
 5100 rather than to conduct every possible test. The extent of testing to be planned for this purpose is a

5101 methodology issue, and needs to be considered in the context of a particular TOE and the balance of  
5102 other evaluation activities.

5103 **13.5.4.3 Developer action elements**

5104 **13.5.4.3.1 ATE\_IND.1.1D**

5105 **The developer shall provide the TOE for testing.**

5106 **13.5.4.4 Content and presentation elements**

5107 **13.5.4.4.1 ATE\_IND.1.1C**

5108 **The TOE shall be suitable for testing.**

5109 **13.5.4.5 Evaluator action elements**

5110 **13.5.4.5.1 ATE\_IND.1.1E**

5111 **The evaluator shall confirm that the information provided meets all requirements for content**  
5112 **and presentation of evidence.**

5113 **13.5.4.5.2 ATE\_IND.1.2E**

5114 **The evaluator shall test a subset of the TSF to confirm that the TSF operates as specified.**

5115 **13.5.5 ATE\_IND.2 Independent testing - sample**

5116 Dependencies: ADV\_FSP.2 Security-enforcing functional specification

5117 AGD\_OPE.1 Operational user guidance

5118 AGD\_PRE.1 Preparative procedures

5119 ATE\_COV.1 Evidence of coverage

5120 ATE\_FUN.1 Functional testing

5121 **13.5.5.1 Objectives**

5122 In this component, the objective is to demonstrate that the TOE operates in accordance with its design  
5123 representations and guidance documents. Evaluator testing confirms that the developer performed  
5124 some tests of some interfaces in the functional specification.

5125 **13.5.5.2 Application notes**

5126 The intent is that the developer should provide the evaluator with materials necessary for the efficient  
5127 reproduction of developer tests. This may include such things as machine-readable test  
5128 documentation, test programs, etc.

5129 This component contains a requirement that the evaluator has available test results from the  
5130 developer to supplement the programme of testing. The evaluator will repeat a sample of the  
5131 developer's tests to gain confidence in the results obtained. Having established such confidence the  
5132 evaluator will build upon the developer's testing by conducting additional tests that exercise the TOE  
5133 in a different manner. By using a platform of validated developer test results the evaluator is able to

5134 gain confidence that the TOE operates correctly in a wider range of conditions than would be possible  
 5135 purely using the developer's own efforts, given a fixed level of resource. Having gained confidence that  
 5136 the developer has tested the TOE, the evaluator will also have more freedom, where appropriate, to  
 5137 concentrate testing in areas where examination of documentation or specialist knowledge has raised  
 5138 particular concerns.

5139 **13.5.5.3 Developer action elements**

5140 **13.5.5.3.1 ATE\_IND.2.1D**

5141 The developer shall provide the TOE for testing.

5142 **13.5.5.4 Content and presentation elements**

5143 **13.5.5.4.1 ATE\_IND.2.1C**

5144 The TOE shall be suitable for testing.

5145 **13.5.5.4.2 ATE\_IND.2.2C**

5146 **The developer shall provide an equivalent set of resources to those that were used in the**  
 5147 **developer's functional testing of the TSF.**

5148 **13.5.5.5 Evaluator action elements**

5149 **13.5.5.5.1 ATE\_IND.2.1E**

5150 The evaluator shall confirm that the information provided meets all requirements for content and  
 5151 presentation of evidence.

5152 **13.5.5.5.2 ATE\_IND.2.2E**

5153 **The evaluator shall execute a sample of tests in the test documentation to verify the developer**  
 5154 **test results.**

5155 **13.5.5.5.3 ATE\_IND.2.3E**

5156 The evaluator shall test a subset of the TSF to confirm that the TSF operates as specified.

5157 **13.5.6 ATE\_IND.3 Independent testing - complete**

5158 Dependencies: ADV\_FSP.4 Complete functional specification

5159 AGD\_OPE.1 Operational user guidance

5160 AGD\_PRE.1 Preparative procedures

5161 ATE\_COV.1 Evidence of coverage

5162 ATE\_FUN.1 Functional testing

5163 **13.5.6.1 Objectives**

5164 In this component, the objective is to demonstrate that the TOE operates in accordance with its design  
5165 representations and guidance documents. Evaluator testing includes repeating all of the developer  
5166 tests.

5167 **13.5.6.2 Application notes**

5168 The intent is that the developer should provide the evaluator with materials necessary for the efficient  
5169 reproduction of developer tests. This may include such things as machine-readable test  
5170 documentation, test programs, etc.

5171 In this component the evaluator must repeat all of the developer's tests as part of the programme of  
5172 testing. As in the previous component the evaluator will also conduct tests that aim to exercise the TSF  
5173 in a different manner from that achieved by the developer. In cases where developer testing has been  
5174 exhaustive, there may remain little scope for this.

5175 **13.5.6.3 Developer action elements**

5176 **13.5.6.3.1 ATE\_IND.3.1D**

5177 The developer shall provide the TOE for testing.

5178 **13.5.6.4 Content and presentation elements**

5179 **13.5.6.4.1 ATE\_IND.3.1C**

5180 The TOE shall be suitable for testing.

5181 **13.5.6.4.2 ATE\_IND.3.2C**

5182 The developer shall provide an equivalent set of resources to those that were used in the developer's  
5183 functional testing of the TSF.

5184 **13.5.6.5 Evaluator action elements**

5185 **13.5.6.5.1 ATE\_IND.3.1E**

5186 The evaluator shall confirm that the information provided meets all requirements for content and  
5187 presentation of evidence.

5188 **13.5.6.5.2 ATE\_IND.3.2E**

5189 The evaluator shall execute **all** tests in the test documentation to verify the developer test results.

5190 **13.5.6.5.3 ATE\_IND.3.3E**

5191 The evaluator shall test the TSF to confirm that the **entire** TSF operates as specified.

5192 **13.6 Composite functional testing (ATE\_COMP)**

5193 **13.6.1 Objectives**

5194 The objective of this family is to determine whether composite product as a whole exhibits the  
5195 properties necessary to satisfy the functional requirements of its Security Target.

5196 **13.6.2 Application notes**

5197 A composite product can be tested by testing the components of it separately and by testing the  
 5198 integrated product. Separate testing means that the platform and the application are being tested  
 5199 independent of each other. A lot of tests of the platform may have been performed within the scope of  
 5200 its accomplished evaluation. The application may be tested on a simulator or an emulator, which  
 5201 represent a virtual machine.

5202 Integration testing means that the composite product is being tested as it is: the application is running  
 5203 on the platform.

5204 Behaviour of implementation of some SFRs can depend on properties of the underlying platform as  
 5205 well as of the application (e.g. correctness of the measures of the composite product to withstand a  
 5206 side channel attack or correctness of the implementation of tamper resistance against physical  
 5207 attacks). In such a case the SFR implementation shall be tested on the final composite product, but not  
 5208 on a simulator or an emulator.

5209 This activity focuses exclusively on testing of the composite product as a whole and represents merely  
 5210 partial efforts within the general test approach being covered by the assurance ATE. These integration  
 5211 tests shall be specified and performed, whereby the approach of the standard assurance families of  
 5212 the class ATE shall be applied.

5213 **13.6.3 ATE\_COMP.1 Composite product functional testing**

5214 Dependencies: No dependencies

5215 **13.6.3.1 Developer action elements**5216 **13.6.3.1.1 ATE\_COMP.1.1D**

5217 **The developer shall provide a set of tests as required by the assurance package chosen.**

5218 **13.6.3.1.2 ATE\_COMP.1.2D**

5219 **The developer shall provide the composite TOE for testing.**

5220 **13.6.3.2 Content and presentation elements**5221 **13.6.3.2.1 ATE\_COMP.1.1C**

5222 **Content and presentation of the specification and documentation of the *integration* tests shall**  
 5223 **correspond to the standard<sup>7</sup> requirements of the assurance families ATE\_FUN and ATE\_COV.**

5224 **13.6.3.2.2 ATE\_COMP.1.2C**

5225 **The composite TOE provided shall be suitable for testing.**

---

<sup>7</sup> i.e. as defined by ISO/IEC 18045

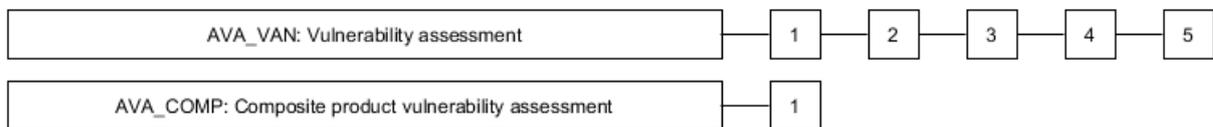
5226 **13.6.3.3 Evaluator action elements**5227 **13.6.3.3.1 ATE\_COMP.1.1E**

5228 **The evaluator shall confirm that the information provided meets all requirements for content**  
 5229 **and presentation of evidence.**

5230 **14 Class AVA: Vulnerability assessment**5231 **14.1 Introduction**

5232 The AVA: Vulnerability assessment class addresses the possibility of exploitable vulnerabilities  
 5233 introduced in the development or the operation of the TOE.

5234 Figure 13 shows the families within this class, and the hierarchy of components within the families.



5235

5236 **Figure 13 — AVA: Vulnerability assessment class decomposition**5237 **14.2 Application notes**

5238 Generally, the vulnerability assessment activity covers various vulnerabilities in the development and  
 5239 operation of the TOE. Development vulnerabilities take advantage of some property of the TOE ,or the  
 5240 product where the TOE resides, which was introduced during its development, e.g. defeating the TSF  
 5241 self-protection through tampering, direct attack or monitoring of the TSF, defeating the TSF domain  
 5242 separation through monitoring or direct attack the TSF, or defeating non-bypassability through  
 5243 circumventing (bypassing) the TSF. Explicit dependencies of the TOE on IT systems in the  
 5244 environment must also be considered. Operational vulnerabilities take advantage of weaknesses in  
 5245 non-technical countermeasures to violate the TOE SFRs, e.g. misuse or incorrect configuration. Misuse  
 5246 investigates whether the TOE can be configured or used in a manner that is insecure, but that an  
 5247 administrator or user of the TOE would reasonably believe to be secure.

5248 Assessment of development vulnerabilities is covered by the assurance family AVA\_VAN. Basically, all  
 5249 development vulnerabilities can be considered in the context of AVA\_VAN due to the fact, that this  
 5250 family allows application of a wide range of assessment methodologies being unspecific to the kind of  
 5251 an attack scenario. These unspecific assessment methodologies comprise, among other, also the  
 5252 specific methodologies for those TSF where covert channels are to be considered (a channel capacity  
 5253 estimation can be done using informal engineering measurements, as well as actual test  
 5254 measurements) or can be overcome by the use of sufficient resources in the form of a direct attack  
 5255 (underlying technical concept of those TSF is based on probabilistic or permutational mechanisms; a  
 5256 qualification of their security behaviour and the effort required to overcome them can be made using a  
 5257 quantitative or statistical analysis).

5258 If there are security objectives specified in the ST to either to prevent one user of the TOE from  
 5259 observing activity associated with another user of the TOE, or to ensure that information flows cannot  
 5260 be used to achieve enforced illicit data signals, covert channel analysis should be considered during  
 5261 the conduct of the vulnerability analysis. This is often reflected by the inclusion of Unobservability

5262 (FPR\_UNO) and multilevel access control policies specified through Access control policy (FDP\_ACC)  
5263 and/or Information flow control policy (FDP\_IFC) requirements in the ST.

### 5264 **14.3 Vulnerability analysis (AVA\_VAN)**

#### 5265 **14.3.1 Objectives**

5266 Vulnerability analysis is an assessment to determine whether potential vulnerabilities identified,  
5267 during the evaluation of the development and anticipated operation of the TOE or by other methods  
5268 (e.g. by flaw hypotheses or quantitative or statistical analysis of the security behaviour of the  
5269 underlying security mechanisms), could allow attackers to violate the SFRs.

5270 Vulnerability analysis deals with the threats that an attacker will be able to discover flaws that will  
5271 allow unauthorised access to data and functionality, allow the ability to interfere with or alter the TSF,  
5272 or interfere with the authorised capabilities of other users.

5273 *In case of a **multi-assurance evaluation** the vulnerability analysis will assess the defined **sub-TSF** as  
5274 well as the TOE as a whole.*

#### 5275 **14.3.2 Component levelling**

5276 Levelling is based on an increasing rigour of vulnerability analysis by the evaluator and increased  
5277 levels of attack potential required by an attacker to identify and exploit the potential vulnerabilities.

#### 5278 **14.3.3 AVA\_VAN.1 Vulnerability survey**

5279 Dependencies: ADV\_FSP.1 Basic functional specification

5280 AGD\_OPE.1 Operational user guidance

5281 AGD\_PRE.1 Preparative procedures

##### 5282 **14.3.3.1 Objectives**

5283 A vulnerability survey of information available in the public domain is performed by the evaluator to  
5284 ascertain potential vulnerabilities that may be easily found by an attacker.

5285 The evaluator performs penetration testing, to confirm that the potential vulnerabilities cannot be  
5286 exploited in the operational environment for the TOE. Penetration testing is performed by the  
5287 evaluator assuming an attack potential of Basic.

##### 5288 **14.3.3.2 Developer action elements**

###### 5289 **14.3.3.2.1 AVA\_VAN.1.1D**

5290 **The developer shall provide the TOE for testing.**

##### 5291 **14.3.3.3 Content and presentation elements**

###### 5292 **14.3.3.3.1 AVA\_VAN.1.1C**

5293 **The TOE shall be suitable for testing.**

- 5294 **14.3.3.4 Evaluator action elements**
- 5295 **14.3.3.4.1 AVA\_VAN.1.1E**
- 5296 **The evaluator shall confirm that the information provided meets all requirements for content**  
5297 **and presentation of evidence.**
- 5298 **14.3.3.4.2 AVA\_VAN.1.2E**
- 5299 **The evaluator shall perform a search of public domain sources to identify potential**  
5300 **vulnerabilities in the TOE.**
- 5301 **14.3.3.4.3 AVA\_VAN.1.3E**
- 5302 **The evaluator shall conduct penetration testing, based on the identified potential**  
5303 **vulnerabilities, to determine that the TOE is resistant to attacks performed by an attacker**  
5304 **possessing Basic attack potential.**
- 5305 **14.3.4 AVA\_VAN.2 Vulnerability analysis**
- 5306 Dependencies: ADV\_ARC.1 Security architecture description
- 5307 ADV\_FSP.2 Security-enforcing functional specification
- 5308 ADV\_TDS.1 Basic design
- 5309 AGD\_OPE.1 Operational user guidance
- 5310 AGD\_PRE.1 Preparative procedures
- 5311 **14.3.4.1 Objectives**
- 5312 A vulnerability analysis is performed by the evaluator to ascertain the presence of potential  
5313 vulnerabilities.
- 5314 The evaluator performs penetration testing, to confirm that the potential vulnerabilities cannot be  
5315 exploited in the operational environment for the TOE. Penetration testing is performed by the  
5316 evaluator assuming an attack potential of Basic.
- 5317 **14.3.4.2 Developer action elements**
- 5318 **14.3.4.2.1 AVA\_VAN.2.1D**
- 5319 The developer shall provide the TOE for testing.
- 5320 **14.3.4.2.2 AVA\_VAN.2.2D**
- 5321 **The developer shall provide a list of third party components included in the TOE and the TOE**  
5322 **delivery.**
- 5323 **14.3.4.3 Content and presentation elements**
- 5324 **14.3.4.3.1 AVA\_VAN.2.1C**
- 5325 The TOE shall be suitable for testing.

5326 **14.3.4.3.2 AVA\_VAN.2.2C**

5327 **The list of third party components shall include components provided by third parties, and that**  
 5328 **are part of the TOE or otherwise part of the TOE delivery.**

5329 **14.3.4.4 Evaluator action elements**5330 **14.3.4.4.1 AVA\_VAN.2.1E**

5331 The evaluator shall confirm that the information provided meets all requirements for content and  
 5332 presentation of evidence.

5333 **14.3.4.4.2 AVA\_VAN.2.2E**

5334 The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in  
 5335 the TOE **the components in the list of third party components, and specific IT products in the**  
 5336 **environment that the TOE depends on.**

5337 **14.3.4.4.3 AVA\_VAN.2.3E**

5338 **The evaluator shall perform an independent vulnerability analysis of the TOE using the**  
 5339 **guidance documentation, functional specification, TOE design and security architecture**  
 5340 **description to identify potential vulnerabilities in the TOE.**

5341 **14.3.4.4.4 AVA\_VAN.2.4E**

5342 The evaluator shall conduct penetration testing, based on the identified potential vulnerabilities, to  
 5343 determine that the TOE is resistant to attacks performed by an attacker possessing Basic attack  
 5344 potential.

5345 **14.3.5 AVA\_VAN.3 Focused vulnerability analysis**

5346 Dependencies: ADV\_ARC.1 Security architecture description

5347 ADV\_FSP.4 Complete functional specification

5348 ADV\_TDS.3 Basic modular design

5349 ADV\_IMP.1 Implementation representation of the TSF

5350 AGD\_OPE.1 Operational user guidance

5351 AGD\_PRE.1 Preparative procedures

5352 ATE\_DPT.1 Testing: basic design

5353 **14.3.5.1 Objectives**

5354 A vulnerability analysis is performed by the evaluator to ascertain the presence of potential  
 5355 vulnerabilities.

5356 The evaluator performs penetration testing, to confirm that the potential vulnerabilities cannot be  
 5357 exploited in the operational environment for the TOE. Penetration testing is performed by the  
 5358 evaluator assuming an attack potential of Enhanced-Basic.

5359 **14.3.5.2 Developer action elements**5360 **14.3.5.2.1 AVA\_VAN.3.1D**

5361 The developer shall provide the TOE for testing.

5362 **14.3.5.2.2 AVA\_VAN.3.2D**

5363 The developer shall provide a list of third party components included in the TOE and the TOE delivery.

5364 **14.3.5.3 Content and presentation elements**5365 **14.3.5.3.1 AVA\_VAN.3.1C**

5366 The TOE shall be suitable for testing.

5367 **14.3.5.3.2 AVA\_VAN.3.2C**

5368 The list of third party components shall include components provided by third parties, and that are  
5369 part of the TOE or otherwise part of the TOE delivery.

5370 **14.3.5.4 Evaluator action elements**5371 **14.3.5.4.1 AVA\_VAN.3.1E**

5372 The evaluator shall confirm that the information provided meets all requirements for content and  
5373 presentation of evidence.

5374 **14.3.5.4.2 AVA\_VAN.3.2E**

5375 The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in  
5376 the TOE the components in the list of third party components, and specific IT products in the  
5377 environment that the TOE depends on.

5378 **14.3.5.4.3 AVA\_VAN.3.3E**

5379 The evaluator shall perform an independent, **focused** vulnerability analysis of the TOE using the  
5380 guidance documentation, functional specification, TOE design, security architecture description **and**  
5381 **implementation representation** to identify potential vulnerabilities in the TOE.

5382 **14.3.5.4.4 AVA\_VAN.3.4E**

5383 The evaluator shall conduct penetration testing, based on the identified potential vulnerabilities, to  
5384 determine that the TOE is resistant to attacks performed by an attacker possessing **Enhanced-Basic**  
5385 attack potential.

5386 **14.3.6 AVA\_VAN.4 Methodical vulnerability analysis**

5387 Dependencies: ADV\_ARC.1 Security architecture description

5388 ADV\_FSP.4 Complete functional specification

5389 ADV\_TDS.3 Basic modular design

5390 ADV\_IMP.1 Implementation representation of the TSF

- 5391 AGD\_OPE.1 Operational user guidance
- 5392 AGD\_PRE.1 Preparative procedures
- 5393 ATE\_DPT.1 Testing: basic design
- 5394 **14.3.6.1 Objectives**
- 5395 A methodical vulnerability analysis is performed by the evaluator to ascertain the presence of  
5396 potential vulnerabilities.
- 5397 The evaluator performs penetration testing, to confirm that the potential vulnerabilities cannot be  
5398 exploited in the operational environment for the TOE. Penetration testing is performed by the  
5399 evaluator assuming an attack potential of Moderate.
- 5400 **14.3.6.2 Developer action elements**
- 5401 **14.3.6.2.1 AVA\_VAN.4.1D**
- 5402 The developer shall provide the TOE for testing.
- 5403 **14.3.6.2.2 AVA\_VAN.4.2D**
- 5404 The developer shall provide a list of third party components included in the TOE and the TOE delivery.
- 5405 **14.3.6.3 Content and presentation elements**
- 5406 **14.3.6.3.1 AVA\_VAN.4.1C**
- 5407 The TOE shall be suitable for testing.
- 5408 **14.3.6.3.2 AVA\_VAN.4.2C**
- 5409 The list of third party components shall include components provided by third parties, and that are  
5410 part of the TOE or otherwise part of the TOE delivery.
- 5411 **14.3.6.4 Evaluator action elements**
- 5412 **14.3.6.4.1 AVA\_VAN.4.1E**
- 5413 The evaluator shall confirm that the information provided meets all requirements for content and  
5414 presentation of evidence.
- 5415 **14.3.6.4.2 AVA\_VAN.4.2E**
- 5416 The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in  
5417 the TOE the components in the list of third party components, and specific IT products in the  
5418 environment that the TOE depends on.
- 5419 **14.3.6.4.3 AVA\_VAN.4.3E**
- 5420 The evaluator shall perform an independent, **methodical** vulnerability analysis of the TOE using the  
5421 guidance documentation, functional specification, TOE design, security architecture description and  
5422 implementation representation to identify potential vulnerabilities in the TOE.

5423 **14.3.6.4.4 AVA\_VAN.4.4E**

5424 The evaluator shall conduct penetration testing based on the identified potential vulnerabilities to  
 5425 determine that the TOE is resistant to attacks performed by an attacker possessing **Moderate** attack  
 5426 potential.

5427 **14.3.7 AVA\_VAN.5 Advanced methodical vulnerability analysis**

5428 Dependencies: ADV\_ARC.1 Security architecture description

5429 ADV\_FSP.4 Complete functional specification

5430 ADV\_TDS.3 Basic modular design

5431 ADV\_IMP.1 Implementation representation of the TSF

5432 AGD\_OPE.1 Operational user guidance

5433 AGD\_PRE.1 Preparative procedures

5434 ATE\_DPT.1 Testing: basic design

5435 **14.3.7.1 Objectives**

5436 A methodical vulnerability analysis is performed by the evaluator to ascertain the presence of  
 5437 potential vulnerabilities.

5438 The evaluator performs penetration testing, to confirm that the potential vulnerabilities cannot be  
 5439 exploited in the operational environment for the TOE. Penetration testing is performed by the  
 5440 evaluator assuming an attack potential of High.

5441 **14.3.7.2 Developer action elements**5442 **14.3.7.2.1 AVA\_VAN.5.1D**

5443 The developer shall provide the TOE for testing.

5444 **14.3.7.2.2 AVA\_VAN.5.2D**

5445 The developer shall provide a list of third party components included in the TOE and the TOE delivery.

5446 **14.3.7.3 Content and presentation elements**5447 **14.3.7.3.1 AVA\_VAN.5.1C**

5448 The TOE shall be suitable for testing.

5449 **14.3.7.3.2 AVA\_VAN.2.2C**

5450 The list of third party components shall include components provided by third parties, and that are  
 5451 part of the TOE or otherwise part of the TOE delivery.

5452 **14.3.7.4 Evaluator action elements**5453 **14.3.7.4.1 AVA\_VAN.5.1E**

5454 The evaluator shall confirm that the information provided meets all requirements for content and  
5455 presentation of evidence.

5456 **14.3.7.4.2 AVA\_VAN.5.2E**

5457 The evaluator shall perform a search of public domain sources to identify potential vulnerabilities in  
5458 the TOE the components in the list of third party components, and specific IT products in the  
5459 environment that the TOE depends on.

5460 **14.3.7.4.3 AVA\_VAN.5.3E**

5461 The evaluator shall perform an independent, methodical vulnerability analysis of the TOE using the  
5462 guidance documentation, functional specification, TOE design, security architecture description and  
5463 implementation representation to identify potential vulnerabilities in the TOE.

5464 **14.3.7.4.4 AVA\_VAN.5.4E**

5465 The evaluator shall conduct penetration testing based on the identified potential vulnerabilities to  
5466 determine that the TOE is resistant to attacks performed by an attacker possessing **High** attack  
5467 potential.

5468 **14.4 Composite vulnerability assessment (AVA\_COMP)**5469 **14.4.1 Objectives**

5470 The aim of this activity is to determine the exploitability of flaws or weaknesses in the composite TOE  
5471 as a whole in the intended environment.

5472 **14.4.2 AVA\_COMP.1 Composite product vulnerability assessment**

5473 Dependencies: No dependencies

5474 **14.4.2.1 Application notes**

5475 This activity focuses exclusively on vulnerability assessment of the composite product *as a whole* and  
5476 represents merely *partial efforts* within the general approach being covered by the standard<sup>8</sup>  
5477 assurance family of the class AVA: AVA\_VAN.

5478 The results of the vulnerability assessment for the underlying platform represented in the ETR\_COMP  
5479 can be reused under the following conditions: they are up to date and all composite activities for  
5480 correctness – ASE\_COMP.1, ALC\_COMP.1, ADV\_COMP.1 and ATE\_COMP.1 – are finalised with the  
5481 verdict PASS.

5482 **The yellow marked references need to be aligned with CEM. This will be done later on if the necessary**  
5483 **content has been included in CEM.**

---

<sup>8</sup> i.e. as defined by ISO/IEC 18045

5484 Due to composing of the platform and the application a new quality arises, which can cause additional  
 5485 vulnerabilities of the platform which might be not mentioned in the ETR\_COMP. In these  
 5486 circumstances [R44] in chapter Annex A applies.

#### 5487 **14.4.2.2 Developer action elements**

##### 5488 **14.4.2.2.1 AVA\_COMP.1.1D**

5489 The developer shall provide the composite TOE for penetrating testing.

#### 5490 **14.4.2.3 Content and presentation elements**

##### 5491 **14.4.2.3.1 AVA\_COMP.1.1C**

5492 The composite TOE provided shall be suitable for testing as a whole.

#### 5493 **14.4.2.4 Evaluator action elements**

##### 5494 **14.4.2.4.1 AVA\_COMP.1.1E**

5495 The evaluator shall conduct penetration testing of the composite product *as a whole* building on  
 5496 evaluator's own vulnerability analysis, to ensure that the vulnerabilities being relevant for the  
 5497 Composite-ST are not exploitable.

### 5498 **15 Class ACO: Composition**

#### 5499 **15.1 Introduction**

5500 The class ACO: Composition encompasses five families. These families specify assurance requirements  
 5501 that are designed to provide confidence that a composed TOE will operate securely when relying upon  
 5502 security functionality provided by previously evaluated software, firmware or hardware components.

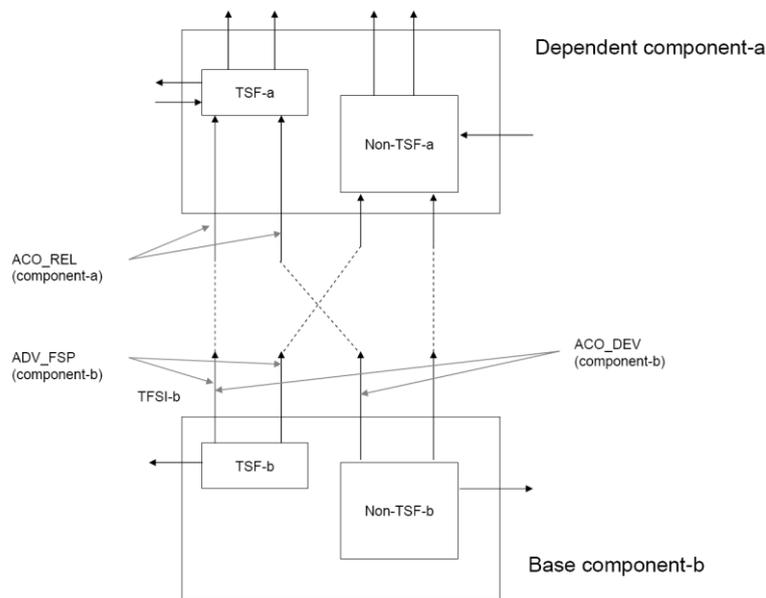
5503 Composition involves taking two or more IT entities successfully evaluated against ISO/IEC 15408  
 5504 security assurance requirements packages (base components and dependent components, see  
 5505 Annex B) and combining them for use, with no further development of either IT entity. The  
 5506 development of additional IT entities is not included (entities that have not previously been the  
 5507 subject of a component evaluation). The composed TOE forms a new product that can be installed and  
 5508 integrated into any specific environment instance that meets the objectives for the environment.

5509 This approach does not provide an alternative approach for the evaluation of components.  
 5510 Composition under ACO provides a composed TOE integrator a method, which can be used as an  
 5511 alternative to other assurance levels specified in ISO/IEC 15408, to gain confidence in a TOE that is the  
 5512 combination of two or more successfully evaluated components without having to re-evaluate the  
 5513 composite TSF. (The composed TOE integrator is referred to as "developer" throughout the ACO class,  
 5514 with any references to the developer of the base or dependent components clarified as such.)

5515 Composed Assurance Packages, as defined in part 5 provide an assurance scale for composed TOEs.  
 5516 This assurance scale is required in addition to other assurance packages, for example the EALs,  
 5517 because to combine components evaluated against another assurance package and gain equivalent  
 5518 assurance in the resulting composed TOE, all SARs have to be applied to the composed TOE.

5519 Although reuse can be made of the component TOE evaluation results, there are often additional  
 5520 aspects of the components that have to be considered in the composed TOE, as described in Annex B.3.  
 5521 Due to the different parties involved in a composed TOE evaluation activity it is generally not possible  
 5522 to gain all necessary evidence about these additional aspects of the components to apply the

5523 appropriate EAL. Hence, CAPs have been defined to address the issue of combining evaluated  
 5524 components and gaining a meaningful result. This is discussed further in Annex B.



5525

5526 **Figure 14 — Relationship between ACO families and interactions between components**

5527 In a composed TOE it is generally the case that one component relies on the services provided by  
 5528 another component. The component requiring services is termed the dependent component and the  
 5529 component providing the services is termed the base component. This interaction and distinct is  
 5530 discussed further in Annex B. It is assumed to be the case that the developer of the dependent  
 5531 component is supporting the composed TOE evaluation in some manner (as developer, sponsor, or  
 5532 just cooperating and providing the necessary evaluation evidence from the dependent component  
 5533 evaluation) The ACO components included in the CAP assurance packages should not be used as  
 5534 augmentations for component TOE evaluations, as this would provide no meaningful assurance for the  
 5535 component.

5536 The families within the ACO class interact in a similar manner to the ADV, ATE and AVA classes in a  
 5537 component TOE evaluation and hence leverage from the specification of requirements from those  
 5538 classes where applicable. There are however a few items specific to composed TOE evaluations. To  
 5539 determine how the components interact and identify any deviations from the evaluations of the  
 5540 components, the dependencies that the dependent component has upon the underlying base  
 5541 component are identified (ACO\_REL). This reliance on the base component is specified in terms of the  
 5542 interfaces through which the dependent component makes calls for services in support of the  
 5543 dependent component SFRs. The interfaces, and at higher levels the supporting behaviour, provided  
 5544 by the base component in response to those service requests are analysed in ACO\_DEV. The ACO\_DEV  
 5545 family is based on the ADV\_TDS family, as at the simplest level the TSF of each component can be  
 5546 viewed as a subsystem of the composed TOE, with additional portions of each component seen as  
 5547 additional subsystems. Therefore, the interfaces between the components are seen as interactions  
 5548 between subsystems in a component TOE evaluation.

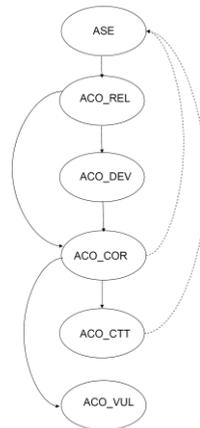
5549 It is possible that the interfaces and supporting behaviour descriptions provided for ACO\_DEV are  
 5550 incomplete. This is determined during the conduct of ACO\_COR. The ACO\_COR family takes the outputs  
 5551 of ACO\_REL and ACO\_DEV and determines whether the components are being used in their evaluated

5552 configuration and identifies where any specifications are incomplete, which are then identified as  
 5553 inputs into testing (ACO\_CTT) and vulnerability analysis (ACO\_VUL) activities of the composed TOE.

5554 Testing of the composed TOE is performed to determine that the composed TOE exhibits the expected  
 5555 behaviour as determined by the composed TOE SFRs, and at higher levels demonstrates the  
 5556 compatibility of the interfaces between the components of the composed TOE.

5557 The vulnerability analysis of the composed TOE leverages from the outputs of the vulnerability  
 5558 analysis of the component evaluations. The composed TOE vulnerability analysis considers any  
 5559 residual vulnerabilities from the component evaluations to determine that the residual vulnerabilities  
 5560 are not applicable to the composed TOE. A search of publicly available information relating to the  
 5561 components is also performed to identify any issues reported in the components since the completion  
 5562 of the respective evaluations.

5563 The interaction between the ACO families is depicted in Figure 15 below. This shows by solid arrowed  
 5564 lines where the evidence and understanding gained in one family feeds into the next activity and the  
 5565 dashed arrows identify where an activity explicitly traces back to the composed TOE SFRs, as  
 5566 described above.

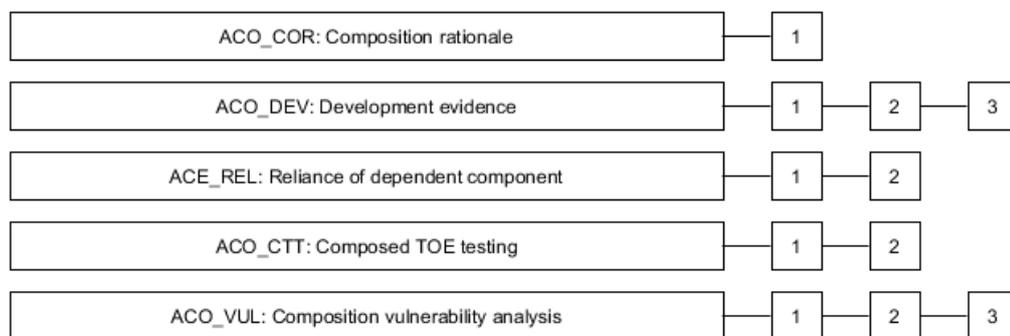


5567

5568 **Figure 15 — Relationship between ACO families**

5569 Further discussion of the definition and interactions within composed TOEs is provided in Annex B.

5570 Figure 16 shows the families within this class, and the hierarchy of components within the families.



5571

5572 **Figure 16 — ACO: Composition class decomposition**

5573 **15.2 Composition rationale (ACO\_COR)**

5574 **15.2.1 Objectives**

5575 This family addresses the requirement to demonstrate that the base component can provide an  
5576 appropriate level of assurance for use in composition.

5577 **15.2.2 Component levelling**

5578 There is only a single component in this family.

5579 **15.2.3 ACO\_COR.1 Composition rationale**

5580 Dependencies: ACO\_DEV.1 Functional Description

5581 ALC\_CMC.1 Labelling of the TOE

5582 ACO\_REL.1 Basic reliance information

5583 **15.2.3.1 Developer action elements**

5584 **15.2.3.1.1 ACO\_COR.1.1D**

5585 **The developer shall provide composition rationale for the base component.**

5586 **15.2.3.2 Content and presentation elements**

5587 **15.2.3.2.1 ACO\_COR.1.1C**

5588 **The composition rationale shall demonstrate that a level of assurance at least as high as that of**  
5589 **the dependent component has been obtained for the support functionality of the base**  
5590 **component, when the base component is configured as required to support the TSF of the**  
5591 **dependent component.**

5592 **15.2.3.3 Evaluator action elements**

5593 **15.2.3.3.1 ACO\_COR.1.1E**

5594 **The evaluator shall confirm that the information meets all requirements for content and**  
5595 **presentation of evidence.**

5596 **15.3 Development evidence (ACO\_DEV)**

5597 **15.3.1 Objectives**

5598 This family sets out requirements for a specification of the base component in increasing levels of  
5599 detail. Such information is required to gain confidence that the appropriate security functionality is  
5600 provided to support the requirements of the dependent component (as identified in the reliance  
5601 information).

5602 **15.3.2 Component levelling**

5603 The components are levelled on the basis of increasing amounts of detail about the interfaces  
5604 provided, and how they are implemented.

5605 **15.3.3 Application notes**

5606 The TSF of the base component is often defined without knowledge of the dependencies of the  
5607 possible applications with which it may be composed. The TSF of this base component is defined to  
5608 include all parts of the base component that have to be relied upon for enforcement of the base  
5609 component SFRs. This will include all parts of the base component required to implement the base  
5610 component SFRs.

5611 The functional specification of the base component will describe the TSFI in terms of the interfaces the  
5612 base component provides to allow an external entity to invoke operations of the TSF. This includes  
5613 interfaces to the human user to permit interaction with the operation of the TSF invoking SFRs and  
5614 also interfaces allowing an external IT entity to make calls into the TSF.

5615 The functional specification only provides a description of what the TSF provides at its interface and  
5616 the means by which that TSF functionality are invoked. Therefore, the functional specification does not  
5617 necessarily provide a complete interface specification of all possible interfaces available between an  
5618 external entity and the base component. It does not include what the TSF expects/requires from the  
5619 operational environment. The description of what a dependent component TSF relies upon of a base  
5620 component is considered in Reliance of dependent component (ACO\_REL) and the development  
5621 information evidence provides a response to the interfaces specified.

5622 The development information evidence includes a specification of the base component. This may be  
5623 the evidence used during evaluation of the base component to satisfy the ADV requirements, or may  
5624 be another form of evidence produced by either the base component developer or the composed TOE  
5625 developer. This specification of the base component is used during Development evidence (ACO\_DEV)  
5626 to gain confidence that the appropriate security functionality is provided to support the requirements  
5627 of the dependent component. The level of detail required of this evidence increases to reflect the level  
5628 of required assurance in the composed TOE. This is expected to broadly reflect the increasing  
5629 confidence gained from the application of the assurance packages to the components. The evaluator  
5630 determines that this description of the base component is consistent with the reliance information  
5631 provided for the dependent component.

5632 **15.3.4 ACO\_DEV.1 Functional Description**

5633 Dependencies: ACO\_REL.1 Basic reliance information

5634 **15.3.4.1 Objectives**

5635 A description of the interfaces in the base component, on which the dependent component relies, is  
5636 required. This is examined to determine whether or not it is consistent with the description of  
5637 interfaces on which the dependent component relies, as provided in the reliance information.

5638 **15.3.4.2 Developer action elements**

5639 **15.3.4.2.1 ACO\_DEV.1.1D**

5640 **The developer shall provide development information for the base component.**

5641 **15.3.4.3 Content and presentation elements**

5642 **15.3.4.3.1 ACO\_DEV.1.1C**

5643 **The development information shall describe the purpose of each interface of the base**  
5644 **component used in the composed TOE.**

5645 **15.3.4.3.2 ACO\_DEV.1.2C**

5646 **The development information shall show correspondence between the interfaces, used in the**  
 5647 **composed TOE, of the base component and the dependent component to support the TSF of the**  
 5648 **dependent component.**

5649 **15.3.4.4 Evaluator action elements**5650 **15.3.4.4.1 ACO\_DEV.1.1E**

5651 **The evaluator shall confirm that the information meets all requirements for content and**  
 5652 **presentation of evidence.**

5653 **15.3.4.4.2 ACO\_DEV.1.2E**

5654 **The evaluator shall determine that the interface description provided is consistent with the**  
 5655 **reliance information provided for the dependent component.**

5656 **15.3.5 ACO\_DEV.2 Basic evidence of design**

5657 Dependencies: ACO\_REL.1 Basic reliance information

5658 **15.3.5.1 Objectives**

5659 A description of the interfaces in the base component, on which the dependent component relies, is  
 5660 required. This is examined to determine whether or not it is consistent with the description of  
 5661 interfaces on which the dependent component relies, as provided in the reliance information.

5662 In addition, the security behaviour of the base component that supports the dependent component  
 5663 TSF is described.

5664 **15.3.5.2 Developer action elements**5665 **15.3.5.2.1 ACO\_DEV.2.1D**

5666 The developer shall provide development information for the base component.

5667 **15.3.5.3 Content and presentation elements**5668 **15.3.5.3.1 ACO\_DEV.2.1C**

5669 The development information shall describe the purpose **and method of use of** each interface of the  
 5670 base component used in the composed TOE.

5671 **15.3.5.3.2 ACO\_DEV.2.2C**

5672 **The development information shall provide a high-level description of the behaviour of the**  
 5673 **base component, which supports the enforcement of the dependent component SFRs.**

5674 **15.3.5.3.3 ACO\_DEV.2.3C**

5675 The development information shall show correspondence between the interfaces, used in the  
 5676 composed TOE, of the base component and the dependent component to support the TSF of the  
 5677 dependent component.

5678 **15.3.5.4 Evaluator action elements**5679 **15.3.5.4.1 ACO\_DEV.2.1E**

5680 The evaluator shall confirm that the information meets all requirements for content and presentation  
5681 of evidence.

5682 **15.3.5.4.2 ACO\_DEV.2.2E**

5683 The evaluator shall determine that the interface description provided is consistent with the reliance  
5684 information provided for the dependent component.

5685 **15.3.6 ACO\_DEV.3 Detailed evidence of design**

5686 Dependencies: ACO\_REL.2 Reliance information

5687 **15.3.6.1 Objectives**

5688 A description of the interfaces in the base component, on which the dependent component relies, is  
5689 required. This is examined to determine whether or not it is consistent with the description of  
5690 interfaces on which the dependent component relies, as provided in the reliance information.

5691 The interface description of the architecture of the base component is provided to enable the evaluator  
5692 to determine whether or not that interface formed part of the TSF of the base component.

5693 **15.3.6.2 Developer action elements**5694 **15.3.6.2.1 ACO\_DEV.3.1D**

5695 The developer shall provide development information for the base component.

5696 **15.3.6.3 Content and presentation elements**5697 **15.3.6.3.1 ACO\_DEV.3.1C**

5698 The development information shall describe the purpose and method of use of each interface of the  
5699 base component used in the composed TOE.

5700 **15.3.6.3.2 ACO\_DEV.3.2C**

5701 **The development information shall identify the subsystems of the base component that**  
5702 **provide interfaces of the base component used in the composed TOE.**

5703 **15.3.6.3.3 ACO\_DEV.3.3C**

5704 The development information shall provide a high-level description of the behaviour of the base  
5705 component **subsystems**, which **support** the enforcement of the dependent component SFRs.

5706 **15.3.6.3.4 ACO\_DEV.3.4C**

5707 **The development information shall provide a mapping from the interfaces to the subsystems of**  
5708 **the base component.**

5709 **15.3.6.3.5 ACO\_DEV.3.5C**

5710 The development information shall show correspondence between the interfaces, used in the  
5711 composed TOE, of the base component and the dependent component to support the TSF of the  
5712 dependent component.

5713 **15.3.6.4 Evaluator action elements**5714 **15.3.6.4.1 ACO\_DEV.3.1E**

5715 The evaluator shall confirm that the information meets all requirements for content and presentation  
5716 of evidence.

5717 **15.3.6.4.2 ACO\_DEV.3.2E**

5718 The evaluator shall determine that the interface description provided is consistent with the reliance  
5719 information provided for the dependent component.

5720 **15.4 Reliance of dependent component (ACO\_REL)**5721 **15.4.1 Objectives**

5722 The purpose of this family is to provide evidence that describes the reliance that a dependent  
5723 component has upon the base component. This information is useful to persons responsible for  
5724 integrating the component with other evaluated IT components to form the composed TOE, and for  
5725 providing insight into the security properties of the resulting composition.

5726 This provides a description of the interface between the dependent and base components of the  
5727 composed TOE that may not have been analysed during evaluation of the individual components, as  
5728 the interfaces were not TSFIs of the individual component TOEs.

5729 **15.4.2 Component levelling**

5730 The components in this family are levelled according to the amount of detail provided in the  
5731 description of the reliance by the dependent component upon the base component.

5732 **15.4.3 Application notes**

5733 The Reliance of dependent component (ACO\_REL) family considers the interactions between the  
5734 components where the dependent component relies upon a service from the base component to  
5735 support the operation of security functionality of the dependent component. The interfaces into these  
5736 services of the base component may not have been considered during evaluation of the base  
5737 component because the service in the base component was not considered security-relevant in the  
5738 component evaluation, either because of the inherent purpose of the service (e.g., adjust type font) or  
5739 because associated ISO/IEC 15408 SFRs are not being claimed in the base component's ST (e.g. the  
5740 login interface when no FIA: Identification and authentication SFRs are claimed). These interfaces into  
5741 the base component are often viewed as functional interfaces in the evaluation of the base component,  
5742 and are in addition to the security interfaces (TSFI) considered in the functional specification.

5743 In summary, the TSFIs described in the functional specification only include the calls made into a TSF  
5744 by external entities and responses to those calls. Calls made by a TSF, which were not explicitly  
5745 considered during evaluation of the components, are described by the reliance information provided  
5746 to satisfy Reliance of dependent component (ACO\_REL).

5747 **15.4.4 ACO\_REL.1 Basic reliance information**

5748 Dependencies: No dependencies.

5749 **15.4.4.1 Developer action elements**

5750 **15.4.4.1.1 ACO\_REL.1.1D**

5751 **The developer shall provide reliance information of the dependent component.**

5752 **15.4.4.2 Content and presentation elements**

5753 **15.4.4.2.1 ACO\_REL.1.1C**

5754 **The reliance information shall describe the functionality of the base component hardware,**  
5755 **firmware and/or software that is relied upon by the dependent component TSF.**

5756 **15.4.4.2.2 ACO\_REL.1.2C**

5757 **The reliance information shall describe all interactions through which the dependent**  
5758 **component TSF requests services from the base component.**

5759 **15.4.4.2.3 ACO\_REL.1.3C**

5760 **The reliance information shall describe how the dependent TSF protects itself from**  
5761 **interference and tampering by the base component.**

5762 **15.4.4.3 Evaluator action elements**

5763 **15.4.4.3.1 ACO\_REL.1.1E**

5764 **The evaluator shall confirm that the information provided meets all requirements for content**  
5765 **and presentation of evidence.**

5766 **15.4.5 ACO\_REL.2 Reliance information**

5767 Dependencies: No dependencies.

5768 **15.4.5.1 Developer action elements**

5769 **15.4.5.1.1 ACO\_REL.2.1D**

5770 The developer shall provide reliance information of the dependent component.

5771 **15.4.5.2 Content and presentation elements**

5772 **15.4.5.2.1 ACO\_REL.2.1C**

5773 **The reliance information shall describe the functionality of the base component hardware, firmware**  
5774 **and/or software that is relied upon by the dependent component TSF.**

5775 **15.4.5.2.2 ACO\_REL.2.2C**

5776 **The reliance information shall describe all interactions through which the dependent component TSF**  
5777 **requests services from the base component.**

5778 **15.4.5.2.3 ACO\_REL.2.3C**

5779 **The reliance information shall describe each interaction in terms of the interface used and the**  
 5780 **return values from those interfaces.**

5781 **15.4.5.2.4 ACO\_REL.2.4C**

5782 The reliance information shall describe how the dependent TSF protects itself from interference and  
 5783 tampering by the base component.

5784 **15.4.5.3 Evaluator action elements**5785 **15.4.5.3.1 ACO\_REL.2.1E**

5786 The evaluator shall confirm that the information provided meets all requirements for content and  
 5787 presentation of evidence.

5788 **15.5 Composed TOE testing (ACO\_CTT)**5789 **15.5.1 Objectives**

5790 This family requires that testing of composed TOE and testing of the base component, as used in the  
 5791 composed TOE, is performed.

5792 **15.5.2 Component levelling**

5793 The components in this family are levelled on the basis of increasing rigour of interface testing and  
 5794 increasing rigour of the analysis of the sufficiency of the tests to demonstrate that the composed TSF  
 5795 operates in accordance with the reliance information and the composed TOE SFRs.

5796 **15.5.3 Application notes**

5797 There are two distinct aspects of testing associated with this family:

5798 a) testing of the interfaces between the base component and the dependent component, which the  
 5799 dependent component rely upon for enforcement of security functionality, to demonstrate their  
 5800 compatibility;

5801 b) testing of the composed TOE to demonstrate that the TOE behaves in accordance with the SFRs for  
 5802 the composed TOE.

5803 If the test configurations used during evaluation of the dependent component included use of the base  
 5804 component as a “platform” and the test analysis sufficiently demonstrates that the TSF behaves in  
 5805 accordance with the SFRs, the developer need perform no further testing of the composed TOE  
 5806 functionality. However, if the base component was not used in the testing of the dependent  
 5807 component, or the configuration of either component varied, then the developer is to perform testing  
 5808 of the composed TOE. This may take the form of repeating the dependent component developer  
 5809 testing of the dependent component, provided this adequately demonstrates the composed TOE TSF  
 5810 behaves in accordance with the SFRs.

5811 The developer is to provide evidence of testing the base component interfaces used in the  
 5812 composition. The operation of base component TSFIs would have been tested as part of the ATE: Tests  
 5813 activities during evaluation of the base component. Therefore, provided the appropriate interfaces  
 5814 were included within the test sample of the base component evaluation and it was determined in  
 5815 Composition rationale (ACO\_COR) that the base component is operating in accordance with the base

5816 component evaluated configuration, with all security functionality required by the dependent  
 5817 component included in the TSF, the evaluator action ACO\_CTT.1.1E may be met through reuse of the  
 5818 base component ATE: Tests verdicts.

5819 If this is not the case, the base component interfaces used relevant to the composition that are affected  
 5820 by any variations to the evaluated configuration and any additional security functionality will be tested  
 5821 to ensure they demonstrate the expected behaviour. The expected behaviour to be tested is that  
 5822 described in the reliance information (Reliance of dependent component (ACO\_REL) evidence).

#### 5823 **15.5.4 ACO\_CTT.1 Interface testing**

5824 Dependencies: ACO\_REL.1 Basic reliance information

5825 ACO\_DEV.1 Functional Description

##### 5826 **15.5.4.1 Objectives**

5827 The objective of this component is to ensure that each interface of the base component, on which the  
 5828 dependent component relies, is tested.

##### 5829 **15.5.4.2 Developer action elements**

###### 5830 **15.5.4.2.1 ACO\_CTT.1.1D**

5831 **The developer shall provide composed TOE test documentation.**

###### 5832 **15.5.4.2.2 ACO\_CTT.1.2D**

5833 **The developer shall provide base component interface test documentation.**

###### 5834 **15.5.4.2.3 ACO\_CTT.1.3D**

5835 **The developer shall provide the composed TOE for testing.**

###### 5836 **15.5.4.2.4 ACO\_CTT.1.4D**

5837 **The developer shall provide an equivalent set of resources to those that were used in the base**  
 5838 **component developer's functional testing of the base component.**

##### 5839 **15.5.4.3 Content and presentation elements**

###### 5840 **15.5.4.3.1 ACO\_CTT.1.1C**

5841 **The composed TOE and base component interface test documentation shall consist of test**  
 5842 **plans, expected test results and actual test results.**

###### 5843 **15.5.4.3.2 ACO\_CTT.1.2C**

5844 **The test documentation from the developer execution of the composed TOE tests shall**  
 5845 **demonstrate that the TSF behaves as specified.**

5846 **15.5.4.3.3 ACO\_CTT.1.3C**

5847 **The test documentation from the developer execution of the base component interface tests**  
 5848 **shall demonstrate that the base component interface relied upon by the dependent component**  
 5849 **behaves as specified.**

5850 **15.5.4.3.4 ACO\_CTT.1.4C**

5851 **The base component shall be suitable for testing.**

5852 **15.5.4.4 Evaluator action elements**5853 **15.5.4.4.1 ACO\_CTT.1.1E**

5854 **The evaluator shall confirm that the information provided meets all requirements for content**  
 5855 **and presentation of evidence.**

5856 **15.5.4.4.2 ACO\_CTT.1.2E**

5857 **The evaluator shall execute a sample of test in the test documentation to verify the developer**  
 5858 **test results.**

5859 **15.5.4.4.3 ACO\_CTT.1.3E**

5860 **The evaluator shall test a subset of the TSF interfaces of the composed TOE to confirm that the**  
 5861 **composed TSF operates as specified.**

5862 **15.5.5 ACO\_CTT.2 Rigorous interface testing**

5863 Dependencies: ACO\_REL.2 Reliance information

5864 ACO\_DEV.2 Basic evidence of design

5865 **15.5.5.1 Objectives**

5866 The objective of this component is to ensure that each interface of the base component, on which the  
 5867 dependent component relies, is tested.

5868 **15.5.5.2 Developer action elements**5869 **15.5.5.2.1 ACO\_CTT.2.1D**

5870 The developer shall provide composed TOE test documentation.

5871 **15.5.5.2.2 ACO\_CTT.2.2D**

5872 The developer shall provide base component interface test documentation.

5873 **15.5.5.2.3 ACO\_CTT.2.3D**

5874 The developer shall provide the composed TOE for testing.

5875 **15.5.5.2.4 ACO\_CTT.2.4D**

5876 The developer shall provide an equivalent set of resources to those that were used in the base  
 5877 component developer's functional testing of the base component.

5878 **15.5.5.3 Content and presentation elements**

5879 **15.5.5.3.1 ACO\_CTT.2.1C**

5880 The composed TOE and base component interface test documentation shall consist of test plans,  
5881 expected test results and actual test results.

5882 **15.5.5.3.2 ACO\_CTT.2.2C**

5883 The test documentation from the developer execution of the composed TOE tests shall demonstrate  
5884 that the TSF behaves as specified **and is complete**.

5885 **15.5.5.3.3 ACO\_CTT.2.3C**

5886 The test documentation from the developer execution of the base component interface tests shall  
5887 demonstrate that the base component interface relied upon by the dependent component behaves as  
5888 specified **and is complete**.

5889 **15.5.5.3.4 ACO\_CTT.2.4C**

5890 The base component shall be suitable for testing.

5891 **15.5.5.4 Evaluator action elements**

5892 **15.5.5.4.1 ACO\_CTT.2.1E**

5893 The evaluator shall confirm that the information provided meets all requirements for content and  
5894 presentation of evidence.

5895 **15.5.5.4.2 ACO\_CTT.2.2E**

5896 The evaluator shall execute a sample of test in the test documentation to verify the developer test  
5897 results.

5898 **15.5.5.4.3 ACO\_CTT.2.3E**

5899 The evaluator shall test a subset of the TSF interfaces of the composed TOE to confirm that the  
5900 composed TSF operates as specified.

5901 **15.6 Composition vulnerability analysis (ACO\_VUL)**

5902 **15.6.1 Objectives**

5903 This family calls for an analysis of vulnerability information available in the public domain and of  
5904 vulnerabilities that may be introduced as a result of the composition.

5905 **15.6.2 Component levelling**

5906 The components in this family are levelled on the basis of increasing scrutiny of vulnerability  
5907 information from the public domain and independent vulnerability analysis.

5908 **15.6.3 Application notes**

5909 The developer will provide details of any residual vulnerabilities reported during evaluation of the  
5910 components. These may be gained from the component developers or evaluation reports for the

5911 components. These will be used as inputs into the evaluator's vulnerability analysis of the composed  
5912 TOE in the operational environment.

5913 The operational environment of the composed TOE is examined to ensure that the assumptions and  
5914 objectives for the component operational environment (specified in each component ST) are satisfied  
5915 in the composed TOE. An initial analysis of the consistency of assumptions and objectives between the  
5916 components and the composed TOE STs will have been performed during the conduct of the ASE  
5917 activities for the composed TOE. However, this analysis is revisited with the knowledge acquired  
5918 during the ACO\_REL, ACO\_DEV and the ACO\_COR activities to ensure that, for example, assumptions of  
5919 the dependent component that were addressed by the environment in the dependent component ST  
5920 are not reintroduced as a result of composition (i.e. that the base component adequately addresses the  
5921 assumptions of the dependent component ST in the composed TOE).

5922 A search by the evaluator for issues in each component will identify potential vulnerabilities reported  
5923 in the public domain since completion of the evaluation of the components. Any potential  
5924 vulnerabilities will then be subject to testing.

5925 If the base component used in the composed TOE has been the subject of assurance continuity  
5926 activities since certification, the evaluator will consider during the composed TOE vulnerability  
5927 analysis activities the changes made in base component.

#### 5928 **15.6.4 ACO\_VUL.1 Composition vulnerability review**

5929 Dependencies: ACO\_DEV.1 Functional Description

##### 5930 **15.6.4.1 Developer action elements**

###### 5931 **15.6.4.1.1 ACO\_VUL.1.1D**

5932 **The developer shall provide the composed TOE for testing.**

##### 5933 **15.6.4.2 Content and presentation elements**

###### 5934 **15.6.4.2.1 ACO\_VUL.1.1C**

5935 **The composed TOE shall be suitable for testing.**

##### 5936 **15.6.4.3 Evaluator action elements**

###### 5937 **15.6.4.3.1 ACO\_VUL.1.1E**

5938 **The evaluator shall confirm that the information provided meets all requirements for content**  
5939 **and presentation of evidence.**

###### 5940 **15.6.4.3.2 ACO\_VUL.1.2E**

5941 **The evaluator shall perform an analysis to determine that any residual vulnerabilities**  
5942 **identified for the base and dependent components are not exploitable in the composed TOE in**  
5943 **its operational environment.**

###### 5944 **15.6.4.3.3 ACO\_VUL.1.3E**

5945 **The evaluator shall perform a search of public domain sources to identify possible**  
5946 **vulnerabilities arising from use of the base and dependent components in the composed TOE**  
5947 **operational environment.**

5948 **15.6.4.3.4 ACO\_VUL.1.4E**

5949 **The evaluator shall conduct penetration testing, based on the identified vulnerabilities, to**  
 5950 **demonstrate that the composed TOE is resistant to attacks by an attacker with basic attack**  
 5951 **potential.**

5952 **15.6.5 ACO\_VUL.2 Composition vulnerability analysis**

5953 Dependencies: ACO\_DEV.2 Basic evidence of design

5954 **15.6.5.1 Developer action elements**5955 **15.6.5.1.1 ACO\_VUL.2.1D**

5956 The developer shall provide the composed TOE for testing.

5957 **15.6.5.2 Content and presentation elements**5958 **15.6.5.2.1 ACO\_VUL.2.1C**

5959 The composed TOE shall be suitable for testing.

5960 **15.6.5.3 Evaluator action elements**5961 **15.6.5.3.1 ACO\_VUL.2.1E**

5962 The evaluator shall confirm that the information provided meets all requirements for content and  
 5963 presentation of evidence.

5964 **15.6.5.3.2 ACO\_VUL.2.2E**

5965 The evaluator shall perform an analysis to determine that any residual vulnerabilities identified for  
 5966 the base and dependent components are not exploitable in the composed TOE in its operational  
 5967 environment.

5968 **15.6.5.3.3 ACO\_VUL.2.3E**

5969 The evaluator shall perform a search of public domain sources to identify possible vulnerabilities  
 5970 arising from use of the base and dependent components in the composed TOE operational  
 5971 environment.

5972 **15.6.5.3.4 ACO\_VUL.2.4E**

5973 **The evaluator shall perform an independent vulnerability analysis of the composed TOE, using**  
 5974 **the guidance documentation, reliance information and composition rationale to identify**  
 5975 **potential vulnerabilities in the composed TOE.**

5976 **15.6.5.3.5 ACO\_VUL.2.5E**

5977 The evaluator shall conduct penetration testing, based on the identified vulnerabilities, to demonstrate  
 5978 that the composed TOE is resistant to attacks by an attacker with basic attack potential.

5979 **15.6.6 ACO\_VUL.3 Enhanced-Basic Composition vulnerability analysis**

5980 Dependencies: ACO\_DEV.3 Detailed evidence of design

5981 **15.6.6.1 Developer action elements**

5982 **15.6.6.1.1 ACO\_VUL.3.1D**

5983 The developer shall provide the composed TOE for testing.

5984 **15.6.6.2 Content and presentation elements**

5985 **15.6.6.2.1 ACO\_VUL.3.1C**

5986 The composed TOE shall be suitable for testing.

5987 **15.6.6.3 Evaluator action elements**

5988 **15.6.6.3.1 ACO\_VUL.3.1E**

5989 The evaluator shall confirm that the information provided meets all requirements for content and  
5990 presentation of evidence.

5991 **15.6.6.3.2 ACO\_VUL.3.2E**

5992 The evaluator shall perform an analysis to determine that any residual vulnerabilities identified for  
5993 the base and dependent components are not exploitable in the composed TOE in its operational  
5994 environment.

5995 **15.6.6.3.3 ACO\_VUL.3.3E**

5996 The evaluator shall perform a search of public domain sources to identify possible vulnerabilities  
5997 arising from use of the base and dependent components in the composed TOE operational  
5998 environment.

5999 **15.6.6.3.4 ACO\_VUL.3.4E**

6000 The evaluator shall perform an independent vulnerability analysis of the composed TOE, using the  
6001 guidance documentation, reliance information and composition rationale to identify potential  
6002 vulnerabilities in the composed TOE.

6003 **15.6.6.3.5 ACO\_VUL.3.5E**

6004 The evaluator shall conduct penetration testing, based on the identified vulnerabilities, to demonstrate  
6005 that the composed TOE is resistant to attacks by an attacker with **Enhanced-Basic** attack potential.

## Annex A (informative)

### Development (ADV)

- 6006  
6007  
6008  
6009
- 6010 This annex contains ancillary material to further explain and provide additional examples for the  
6011 topics brought up in families of the ADV: Development class.
- 6012 **A.1ADV\_ARC: Supplementary material on security architectures**
- 6013 A security architecture is a set of properties that the TSF exhibits; these properties include self-  
6014 protection, domain separation, and non-bypassability. Having these properties provides a basis of  
6015 confidence that the TSF is providing its security services. This annex provides additional material on  
6016 these properties, as well as discussion on contents of a security architecture description.
- 6017 The remainder of this subclause first explains these properties, then discusses the kinds of information  
6018 that are needed to describe how the TSF exhibits those properties.
- 6019 **A.1.1 Security architecture properties**
- 6020 *Self-protection* refers to the ability of the TSF to protect itself from manipulation from external entities  
6021 that may result in changes to the TSF. Without these properties, the TSF might be disabled from  
6022 performing its security services.
- 6023 It is oftentimes the case that a TOE uses services or resources supplied by other IT entities in order to  
6024 perform its functions (e.g. an application that relies upon its underlying operating system). In these  
6025 cases, the TSF does not protect itself entirely on its own, because it depends on the other IT entities to  
6026 protect the services it uses.
- 6027 *Domain separation* is a property whereby the TSF creates separate *security domains* for each untrusted  
6028 active entity to operate on its resources, and then keeps those domains separated from one another so  
6029 that no entity can run in the domain of any other. For example, an operating system TOE supplies a  
6030 domain (address space, per-process environment variables) for each process associated with  
6031 untrusted entities.
- 6032 For some TOEs such domains do not exist because all of the actions of the untrusted entities are  
6033 brokered by the TSF. A packet-filter firewall is an example of such a TOE, where there are no untrusted  
6034 entity domains; there are only data structures maintained by the TSF. The existence of domains, then,  
6035 is dependant upon 1) the type of TOE and 2) the SFRs levied on the TOE. In the cases where the TOE  
6036 does provide domains for untrusted entities, this family requires that those domains are isolated from  
6037 one another such that untrusted entities in one domain are prevented from tampering (affecting  
6038 without brokering by the TSF) from another untrusted entity's domain.
- 6039 *Non-bypassability* is a property that the security functionality of the TSF (as specified by the SFRs) is  
6040 always invoked and cannot be circumvented when appropriate for that specific mechanism. For  
6041 example, if access control to files is specified as a capability of the TSF via an SFR, there must be no  
6042 interfaces through which files can be accessed without invoking the TSF's access control mechanism  
6043 (an interface through which a raw disk access takes place might be an example of such an interface).

6044 As is the case with self-protection, the very nature of some TOEs might depend upon their  
 6045 environments to play a role in non-bypassability of the TSF. For example, a security application TOE  
 6046 requires that it be invoked by the underlying operating system. Similarly, a firewall depends upon the  
 6047 fact that there are no direct connections between the internal and external networks and that all traffic  
 6048 between them must go through the firewall.

### 6049 **A.1.2 Security architecture descriptions**

6050 The security architecture description explains how the properties described above are exhibited by  
 6051 the TSF. It describes how domains are defined and how the TSF keeps them separate. It describes what  
 6052 prevents untrusted processes from getting to the TSF and modifying it. It describes what ensures that  
 6053 all resources under the TSF's control are adequately protected and that all actions related to the SFRs  
 6054 are mediated by the TSF. It explains any role the environment plays in any of these (e.g. presuming it  
 6055 gets correctly invoked by its underlying environment, how are its security functions invoked?).

6056 The security architecture description presents the TSF's properties of self-protection, domain  
 6057 separation, and non-bypassability in terms of the decomposition descriptions. The level of this  
 6058 description is commensurate with the TSF description required by the ADV\_FSP, ADV\_TDS and  
 6059 ADV\_IMP requirements that are being claimed. For example, if ADV\_FSP is the only TSF description  
 6060 available, it would be difficult to provide any meaningful security architecture description because  
 6061 none of the details of any internal workings of the TSF would be available.

6062 However, if the TOE design were also available, even at the most basic level (ADV\_TDS.1), there would  
 6063 be some information available concerning the subsystems that make up the TSF, and there would be a  
 6064 description of how they work to implement self-protection, domain separation, and non-bypassability.  
 6065 For example, perhaps all user interaction with the TOE is constrained through a process that acts on  
 6066 that user's behalf, adopting all of the user's security attributes; the security architecture description  
 6067 would describe how such a process comes into being, how the process's behaviour is constrained by  
 6068 the TSF (so it cannot corrupt the TSF), how all actions of that process are mediated by the TSF  
 6069 (thereby explaining why the TSF cannot be bypassed), etc.

6070 If the available TOE design is more detailed (e.g. at the modular level), or the implementation  
 6071 representation is also available, then the security architecture description would be correspondingly  
 6072 more detailed, explaining how the user's process communicate with the TSF processes, how different  
 6073 requests are processed by the TSF, what parameters are passed, what programmatic protections  
 6074 (buffer overflow prevention, parameter bounds checking, time of check/time of use checking, etc.) are  
 6075 in place. Similarly, a TOE whose ST claimed the ADV\_IMP component would go into implementation-  
 6076 specific detail.

6077 The explanations provided in the security architecture description are expected to be of sufficient  
 6078 detail that one would be able to test their accuracy. That is, simple assertions (e.g. "The TSF keeps  
 6079 domains separate") provide no useful information to convince the reader that the TSF does indeed  
 6080 create and separate domains.

#### 6081 **A.1.2.1 Domain Separation**

6082 In cases where the TOE exhibits domain separation entirely on its own, there would be a  
 6083 straightforward description of how this is attained. The security architecture description would  
 6084 explain the different kinds of domains that are defined by the TSF, how they are defined (i.e. what  
 6085 resources are allocated to each domain), how no resources are left unprotected, and how the domains  
 6086 are kept separated so that active entities in one domain cannot tamper with resources in another  
 6087 domain.

6088 For cases where the TOE depends upon other IT entities to play a role in domain separation, that  
 6089 sharing of roles must be made clear. For example, a TOE that is solely application software relies upon  
 6090 the underlying operating system to correctly instantiate the domains that the TOE defines; if the TOE  
 6091 defines separate processing space, memory space, etc, for each domain, it depends upon the  
 6092 underlying operating system to operate correctly and benignly (e.g. allow the process to execute only  
 6093 in the execution space that is requested by the TOE software).

6094 For example, mechanisms that implement domain separation (e.g., memory management, protected  
 6095 processing modes provided by the hardware, etc.) would be identified and described. Or, the TSF  
 6096 might implement software protection constructs or coding conventions that contribute to  
 6097 implementing separation of software domains, perhaps by delineating user address space from system  
 6098 address space.

6099 The vulnerability analysis and testing (see AVA\_VAN) activities will likely include attempts to defeat  
 6100 the described TSF domain separation through the use of monitoring or direct attack the TSF.

#### 6101 **A.1.2.2 TSF Self-protection**

6102 In cases where the TOE exhibits self-protection entirely on its own, there would be a straightforward  
 6103 description of how this self-protection is attained. Mechanisms that provide domain separation to  
 6104 define a TSF domain that is protected from other (user) domains would be identified and described.

6105 For cases where the TOE depends upon other IT entities to play a role in protecting itself, that sharing  
 6106 of roles must be made clear. For example, a TOE that is solely application software relies upon the  
 6107 underlying operating system to operate correctly and benignly; the application cannot protect itself  
 6108 against a malicious operating system that subverts it (for example, by overwriting its executable code  
 6109 or TSF data).

6110 The security architecture description also covers how user input is handled by the TSF in such a way  
 6111 that the TSF does not subject itself to being corrupted by that user input. For example, the TSF might  
 6112 implement the notion of privilege and protect itself by using privileged-mode routines to handle user  
 6113 data. The TSF might make use of processor-based separation mechanisms (e.g. privilege levels or  
 6114 rings) to separate TSF code and data from user code and data. The TSF might implement software  
 6115 protection constructs or coding conventions that contribute to implementing separation of software,  
 6116 perhaps by delineating user address space from system address space.

6117 For TOEs that start up in a low-function mode (for example, a single-user mode accessible only to  
 6118 installers or administrators) and then transition to the evaluated secure configuration (a mode  
 6119 whereby untrusted users are able to login and use the services and resources of the TOE), the security  
 6120 architecture description also includes an explanation of how the TSF is protected against this  
 6121 initialisation code that does not run in the evaluated configuration. For such TOEs, the security  
 6122 architecture description would explain what prevents those services that should be available only  
 6123 during initialisation (e.g. direct access to resources) from being accessible in the evaluated  
 6124 configuration. It would also explain what prevents initialisation code from running while the TOE is in  
 6125 the evaluated configuration.

6126 There must also be an explanation of how the trusted initialisation code will maintain the integrity of  
 6127 the TSF (and of its initialisation process) such that the initialisation process is able to detect any  
 6128 modification that would result in the TSF being spoofed into believe it was in an initial secure state.

6129 The vulnerability analysis and testing (see AVA\_VAN) activities will likely include attempts to defeat  
 6130 the described TSF self protection through the use of tampering, direct attack, or monitoring of the TSF.

### 6131 **A.1.2.3 TSF Non-Bypassability**

6132 The property of non-bypassability is concerned with interfaces that permit the bypass of the  
 6133 enforcement mechanisms. In most cases this is a consequence of the implementation, where if a  
 6134 programmer is writing an interface that accesses or manipulates an object, it is that programmer's  
 6135 responsibility to use interfaces that are part of the SFR enforcement mechanism for the object and not  
 6136 to try to circumvent those interfaces. For the description pertaining to non-bypassability, then, there  
 6137 are two broad areas that have to be covered.

6138 The first consists of those interfaces to the SFR-enforcement. The property for these interfaces is that  
 6139 they contain no operations or modes that allow them to be used to bypass the TSF. It is likely that the  
 6140 evidence for ADV\_FSP and ADV\_TDS can be used in large part to make this determination. Because  
 6141 non-bypassability is the concern, if only certain operations available through these TSFIs are  
 6142 documented (because they are SFR-enforcing) and others are not, the developer should consider  
 6143 whether additional information (to that presented in ADV\_FSP and ADV\_TDS) is necessary to make a  
 6144 determination that the SFR-supporting and SFR-non-interfering operations of the TSFI do not afford  
 6145 an untrusted entity the ability to bypass the policy being enforced. If such information is necessary, it  
 6146 is included in the security architecture description.

6147 The second area of non-bypassability is concerned with those interfaces whose interactions are not  
 6148 associated with SFR-enforcement. Depending on the ADV\_FSP and ADV\_TDS components claimed,  
 6149 some information about these interfaces may or may not exist in the functional specification and TOE  
 6150 design documentation. The information presented for such interfaces (or groups of interfaces) should  
 6151 be sufficient so that a reader can make a determination (at the level of detail commensurate with the  
 6152 rest of the evidence supplied in the ADV: Development class) that the enforcement mechanisms cannot  
 6153 be bypassed.

6154 The property that the security functionality cannot be bypassed applies to all security functionality  
 6155 equally. That is, the design description should cover objects that are protected under the SFRs (e.g.  
 6156 FDP\_\* components) and functionality (e.g., audit) that is provided by the TSF. The description should  
 6157 also identify the interfaces that are associated with security functionality; this might make use of the  
 6158 information in the functional specification. This description should also describe any design  
 6159 constructs, such as object managers, and their method of use. For instance, if routines are to use a  
 6160 standard macro to produce an audit record, this convention is a part of the design that contributes to  
 6161 the non-bypassability of the audit mechanism. It is important to note that *non-bypassability* in this  
 6162 context is not an attempt to answer the question "could a part of the TSF implementation, if malicious,  
 6163 bypass the security functionality", but rather to document how the implementation does not bypass  
 6164 the security functionality.

6165 The vulnerability analysis and testing (see AVA\_VAN) activities will likely include attempts to defeat  
 6166 the described non-bypassability by circumventing the TSF.

### 6167 **A.2ADV\_FSP: Supplementary material on functional specification**

6168 The purpose in specifying the TSFIs is to provide the necessary information to conduct testing;  
 6169 without knowing the possible means interact with the TSF, one cannot adequately test the behaviour  
 6170 of the TSF.

6171 There are two parts to specifying the TSFIs: identifying them and describing them. Because of the  
 6172 diversity of possible TOEs, and of different TSFs therein, there is no standard set of interfaces that  
 6173 constitute "TSFIs". This annex provides guidance on the factors that determine which interfaces are  
 6174 TSFIs.

6175 **A.2.1 Non-TSF part of the TOE**

6176 The TSF comprises all parts of the TOE the user has to rely on in order to trust the security  
6177 functionality.

6178 To say it in other words: Those parts of the TOE that do not belong to the TSF can be modified by an  
6179 attacker without any impact on the TOE security functionality. If this isn't the case, these parts of the  
6180 TOE have to be included in the TSF.

6181 If the TSF and the TSF implementation are defined then it is clear whether there exist further parts of  
6182 the TOE which can be classified as non-TSF parts of the TOE. Such parts do not have to be part of the  
6183 TSF but they are still part of the TOE.

6184 The relationship between TSF and non-TSF parts of TOE is given by their definitions and the ARC  
6185 properties as follows:

- 6186 • non-TSF parts do not bypass the TSF and
- 6187 • parts of the TSF protects themselves against tampering.

6188 A subsystem of the TOE which is not part of the TSF has to fulfil the following condition (described as a  
6189 rule of thumb<sup>9</sup>): The subsystem must not have any security impact of the TOE even if it were  
6190 substituted by an attacker.

6191 Therefore between the Non-TSF parts and the TSF parts it seems that some kind of "separation  
6192 mechanism" is advisable<sup>10</sup> because such "separation mechanism" may build the basis for the  
6193 assessment that there is no impact on the TSF parts from the Non-TSF parts possible.

6194 Such "separation mechanism" could be implemented by the security architecture or by an explicitly  
6195 realised part of the implementation (e.g. a firewall between TSF and Non-TSF parts of the TOE).

6196 The analysis of the "separation mechanism" is then subject of the vulnerability assessment because it  
6197 must withstand attacks by an attacker of the respective strength according to the VAN level of the  
6198 evaluation.

6199 The developer shall provide evidence for non-bypassability and self-protection in its security  
6200 architecture description and the evaluator shall analyse this evidence in subactivity for ADV\_ARC.1  
6201 and assess the effectiveness in the vulnerability assessment.

6202 The goal of TOE design documentation is to provide sufficient information to determine the TSF  
6203 boundary, and to describe how the TSF implements the SFR. Further attention is needed by the fact  
6204 that the family ADV\_TDS requires only identification of the non-TSF subsystems of the TOE. No  
6205 interface description is provided for these subsystems in ADV\_FSP or ADV\_TDS. SFR non-interference  
6206 of these subsystems is assumed but not demonstrated by the developer and not examined in details by  
6207 the evaluator. However from the TOE design point of view this is not that important as long as the  
6208 above mentioned separation mechanism is in place and the vulnerability assessment confirms that it is

---

<sup>9</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>10</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.

6209 strong enough. Therefore this "separation mechanism" implements the TSF or enforces ARC  
 6210 properties as security feature. But non-bypassability may be enforced by "pure architecture  
 6211 properties" as well.

6212 Parts of the TOE classified as non-TSF must not provide means to bypass the TSF (no matter whether a  
 6213 valid user or even an attacker makes uses of those parts) and must not contribute to the TSF. It is  
 6214 important that the developer provides clear evidence and demonstrate how this requirement is  
 6215 fulfilled.

6216 Therefore the developer shall demonstrate and the evaluator shall examine that the TOE identification  
 6217 of subsystems as non-TSF (cf. ADV\_TDS.x.1) is correct and consequently no detailed description of  
 6218 these subsystems is necessary. The evaluator examination shall include the ARC properties non-  
 6219 bypassability and self-protection being described in the ADV\_ARC documentation provided by the  
 6220 developer (see the paragraphs above).

## 6221 **A.2.2 Determining the TSFI**

6222 In order to identify the interfaces to the TSF, the parts of the TOE that make up the TSF must first be  
 6223 identified. This identification is actually a part of the TOE design (ADV\_TDS) analysis, but is also  
 6224 performed implicitly (through identification and description of the TSFI) by the developer in cases  
 6225 where TOE design (ADV\_TDS) is not included in the assurance package. In this analysis, a portion of  
 6226 the TOE must be considered to be in the TSF if it contributes to the satisfaction of an SFR in the ST (in  
 6227 whole or in part). This includes, for example, everything in the TOE that contributes to TSF run-time  
 6228 initialisation, such as software that runs prior to the TSF being able to protect itself because  
 6229 enforcement of the SFRs has not yet begun (e.g., while booting up). Also included in the TSF are all  
 6230 parts of the TOE that contribute to the architectural principles of TSF self-protection, domain  
 6231 separation, and non-bypassability (see Security Architecture (ADV\_ARC)).

6232 Once the TSF has been defined, the TSFI are identified. The TSFI consists of all means by which  
 6233 external entities (or subjects in the TOE but outside of the TSF) supply data to the TSF, receive data  
 6234 from the TSF and invoke services from the TSF. These service invocations and responses are the  
 6235 means of crossing the TSF boundary. While many of these are readily apparent, others might not be as  
 6236 obvious. The question that should be asked when determining the TSFIs is: "How can a potential  
 6237 attacker interact with the TSF in an attempt to subvert the SFRs?"

6238 Therefore from the evaluation point of view it is also important whether the interface can be misused  
 6239 by an attacker to get access to the security functionality in order to compromise the assets protected  
 6240 by TSF.

6241 Any interface of the TSF which can be potentially used by an attacker belongs to the TSFI (regardless  
 6242 of the further classification as SFR-enforcing, SFR-supporting or SFR-non-interfering).

6243 It is not important whether the TSF will be accessed from outside or whether the TSF accesses the  
 6244 external resources (e.g. TSF calls platform or user). The only criteria is whether there is a potential  
 6245 interference with the TSF from outside.

6246 The following discussions illustrate the application of the TSFI definition in different contexts.

### 6247 **A.2.2.1 Electrical interfaces**

6248 In TOEs such as smart cards, where the adversary has not only logical access to the TOE, but also  
 6249 complete physical access to the TOE, the TSF boundary is the physical boundary. Therefore, the  
 6250 exposed electrical interfaces are considered TSFI because their manipulation could affect the

6251 behaviour of the TSF. As such, all these interfaces (electrical contacts) need to be described: various  
6252 voltages that might be applied, etc.

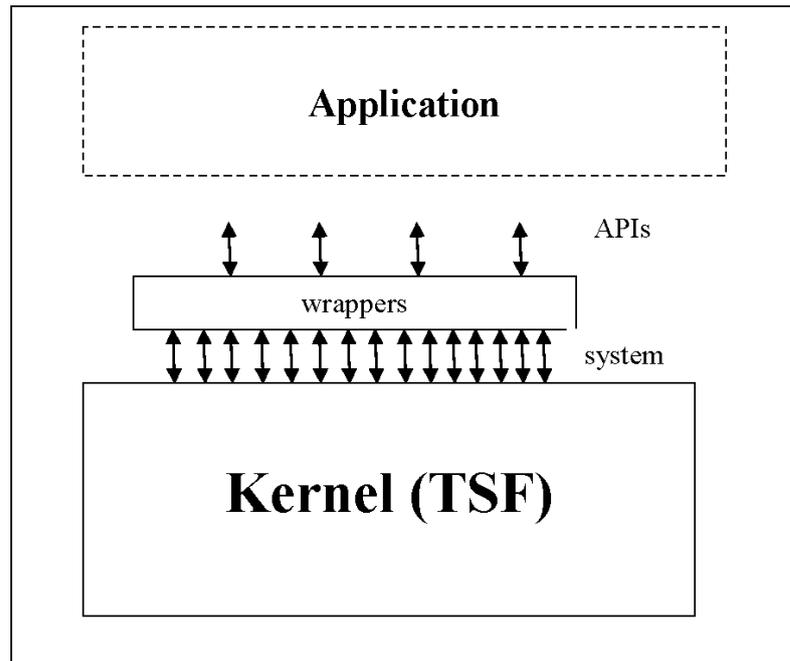
#### 6253 **A.2.2.2 Network protocol stack**

6254 The TSFIs of a TOE that performs protocol processing would be those protocol layers to which a  
6255 potential attacker has direct access. This need not be the entire protocol stack, but it might be.

6256 For example, if the TOE were some sort of a network appliance that allowed potential attackers to  
6257 affect every level of the protocol stack (i.e. to send arbitrary signals, arbitrary voltages, arbitrary  
6258 packets, arbitrary datagrams, etc.), then the TSF boundary exists at each layer of the stack. Therefore,  
6259 the functional specification would have to address every protocol at every layer of the stack.

6260 If, however, the TOE were a firewall that protects an internal network from the Internet, a potential  
6261 attacker would have no means of directly manipulating the voltages that enter the TOE; any extreme  
6262 voltages would simply not be passed through the Internet. That is, the attacker would have access only  
6263 to those protocols at the Internet layer or above. The TSF boundary exists at each layer of the stack.  
6264 Therefore, the functional specification would have to address only those protocols at or above the  
6265 Internet layer: it would describe each of the different communication layers at which the firewall is  
6266 exposed in terms of what constitutes well-formed input for what might appear on the line, and the  
6267 result of both well-formed and malformed inputs. For example, the description of the Internet protocol  
6268 layer would describe what constitutes a well-formed IP packet and what happens when both  
6269 correctly-formed and malformed packets are received. Likewise, the description of the TCP layer  
6270 would describe a successful TCP connection and what happens both when successful connections are  
6271 established and when connections cannot be established or are inadvertently dropped. Presuming the  
6272 firewall's purpose is to filter application-level commands (like FTP or telnet), the description of the  
6273 application layer would describe the application-level commands that are recognised and filtered by  
6274 the firewall, as well as the results of encountering unknown commands.

6275 The descriptions of these layers would likely reference published communication standards (telnet,  
6276 FTP, TCP, etc.) that are used, noting which user-defined options are chosen.

6277 **A.2.2.3 Wrappers**

6278

6279

**Figure A.1 — Wrappers**

6280 “Wrappers” translate complex series of interactions into simplified common services, such as when  
 6281 Operating Systems create APIs for use by applications (as shown in Figure A.1). Whether the TSFIs  
 6282 would be the system calls or the APIs depends upon what is available to the application: if the  
 6283 application can use the system calls directly, then the system calls are the TSFIs. If, however, there  
 6284 were something that prohibits their direct use and requires all communication through the APIs, then  
 6285 the APIs would be the TSFIs.

6286 A Graphical User interface is similar: it translates between machine-understandable commands and  
 6287 user-friendly graphics. Similarly, the TSFIs would be the commands if users have access to them, or the  
 6288 graphics (pull-down menus, check-boxes, text fields) if the users are constrained to using them.

6289 It is worth noting that, in both of these examples, if the user is prohibited from using the more  
 6290 primitive interfaces (i.e. the system calls or the commands), the description of this restriction and of  
 6291 its enforcement would be included in the Security Architecture Description (see A.1). Also, the  
 6292 wrapper would be part of the TSF.

6293 **A.2.2.4 Inaccessible interfaces**

6294 For a given TOE, not all of the interfaces may be *accessible*. That is, the security objectives for the  
 6295 operational environment (in the Security Target) may prevent access to these interfaces or limit  
 6296 access in such a way that they are practically inaccessible. Such interfaces would not be considered  
 6297 TSFIs. Some examples:

- 6298 a) If the security objectives for the operational environment for the stand-alone firewall state that  
 6299 “the firewall will be operational in a server room environment to which only trusted and trained  
 6300 personnel will have access, and which will be equipped with an interruptible power supply  
 6301 (against power failure)”, physical and power interfaces will not be accessible, since trusted and  
 6302 trained personnel will not attempt to dismantle the firewall and/or disable its power supply.

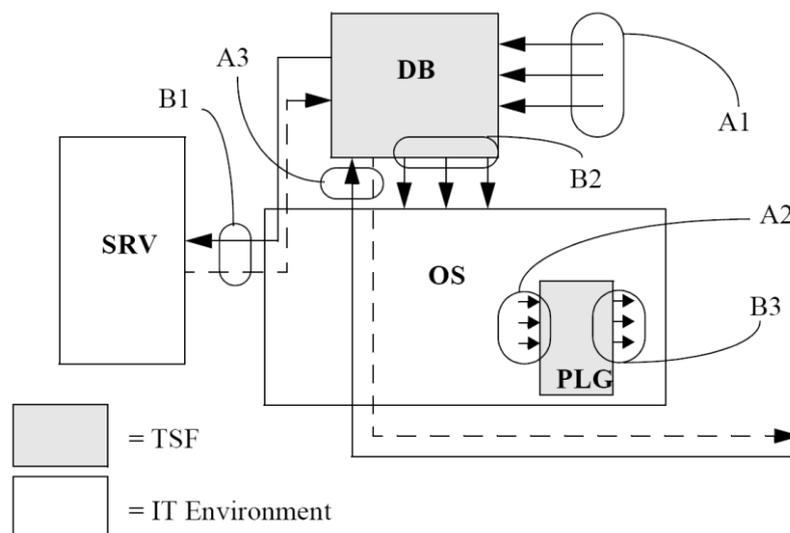
6303 b) If the security objectives for the operational environment for the software firewall (application)  
 6304 state that “the OS and the hardware will provide a security domain for the application free from  
 6305 tampering by other programs”, the interfaces through which the firewall can be accessed by other  
 6306 applications on the OS (e.g. deleting or modifying the firewall executable, direct reading or writing  
 6307 to the memory space of the firewall) will not be accessible, since the OS/hardware part of the  
 6308 operational environment makes this interface inaccessible.

6309 c) If the security objectives for the operational environment for the software firewall additionally  
 6310 state that the OS and hardware will faithfully execute the commands of the TOE, and will not  
 6311 tamper with the TOE in any manner, interfaces through which the firewall obtains primitive  
 6312 functionality from the OS and hardware (executing machine code instructions, OS APIs, such as  
 6313 creating, reading, writing or deleting files, graphical APIs etc.) will not be accessible, since the  
 6314 OS/hardware are the only entities that can access that interface, and they are completely trusted.

6315 For all of these examples, these inaccessible interfaces would not be TSFIs.

### 6316 A.2.3 Example: A complex DBMS

6317 Figure A.2 illustrates a complex TOE: a database management system that relies on hardware and  
 6318 software that is outside the TOE boundary (referred to as the *IT environment* in the rest of this  
 6319 discussion). To simplify this example, the TOE is identical to the TSF. The shaded boxes represent the  
 6320 TSF, while the unshaded boxes represent IT entities in the environment. The TSF comprises the  
 6321 database engine and management GUIs (represented by the box labelled *DB*) and a kernel module that  
 6322 runs as part of the OS that performs some security function (represented by the box labelled *PLG*). The  
 6323 TSF kernel module has entry points defined by the OS specification that the OS will call to invoke some  
 6324 function (this could be a device driver, or an authentication module, etc.). The key is that this  
 6325 pluggable kernel module is providing security services specified by functional requirements in the ST.



6326

6327 **Figure A.2 — Interfaces in a DBMS system**

6328 The IT environment consists of the operating system itself (represented by the box labelled *OS*), as  
 6329 well as an external server (labelled *SRV*). This external server, like the OS, provides a service that the  
 6330 TSF depends on, and thus needs to be in the IT environment. Interfaces in the figure are labelled *A<sub>x</sub>* for  
 6331 TSFI, and *B<sub>x</sub>* for other interfaces that would be documented in ACO: Composition. Each of these groups  
 6332 of interfaces is now discussed.

6333 Interface group A1 represents the most obvious set of TSFI. These are interfaces used by users to  
6334 directly access the database and its security functionality and resources.

6335 Interface group A2 represent the TSFI that the OS invokes to obtain the functionality provided by the  
6336 pluggable module. These are contrasted with interface group B3, which represent calls that the  
6337 pluggable module makes to obtain services from the IT environment.

6338 Interface group A3 represent TSFI that pass through the IT environment. In this case, the DBMS  
6339 communicates over the network using a proprietary application-level protocol. While the IT  
6340 environment is responsible for providing various supporting protocols (e.g., Ethernet, IP, TCP), the  
6341 application layer protocol that is used to obtain services from the DBMS is a TSFI and must be  
6342 documented as such. The dotted line indicates return values/services from the TSF over the network  
6343 connection.

6344 The interfaces labelled *Bx* represent interfaces to functionality in the IT Environment. These interfaces  
6345 are not TSFI and need only be discussed and analysed when the TOE is being used in a composite  
6346 evaluation as part of the activities associated with the ACO class.

#### 6347 **A.2.4 Example Functional Specification**

6348 The Example firewall is used between an internal network and an external network. It verifies the  
6349 source address of data received (to ensure that external data is not attempting to masquerade as  
6350 originating from the internal data); if it detects any such attempts, it saves the offending attempt to the  
6351 audit log. The administrator connects to the firewall by establishing a telnet connection to the firewall  
6352 from the internal network. Administrator actions consist of authenticating, changing passwords,  
6353 reviewing the audit log, and setting or changing the addresses of the internal and external networks.

6354 The Example firewall presents the following interfaces to the internal network:

6355 a) IP datagrams

6356 b) Administrator Commands

6357 and the following interfaces to the external network:

6358 a) IP datagrams

#### 6359 **Interfaces Descriptions: IP Datagrams**

6360 The datagrams are in the format specified by RFC 791.

6361 • Purpose - to transmit blocks of data (“datagrams”) from source hosts to destination hosts  
6362 identified by fixed length addresses; also provides for fragmentation and reassembly of long  
6363 datagrams, if necessary, for transmission through small-packet networks.

6364 • Method of Use - they arrive from the lower-level (e.g. data link) protocol.

6365 • Parameters - the following fields of the IP datagram header: source address, destination address,  
6366 don't-fragment flag.

6367 • Parameter description - [As defined by RFC 791, subclause 3.1 (“Internet Header Format”)]

6368 • Actions - Transmits datagrams that are not masquerading; fragments large datagrams if necessary;  
6369 reassembles fragments into datagrams.

- 6370 • Error messages - (none). No reliability guaranteed (reliability to be provided by upper-level  
6371 protocols) Undeliverable datagrams (e.g. must be fragmented for transmission, but don't-fragment  
6372 flag is set) dropped.

### 6373 Interfaces Descriptions: Administrator Commands

6374 The administrator commands provide a means for the administrator to interact with the firewall.  
6375 These **commands and responses ride atop a telnet (RFC 854) connection established from any**  
6376 **host on the internal** network. Available commands are:

#### 6377 • **Passwd**

- 6378 • Purpose - sets administrator password
- 6379 • Method of Use - **Passwd** <password>
- 6380 • Parameters - password
- 6381 • Parameter description - value of new password
- 6382 • Actions - changes password to new value supplied. There are no restrictions.
- 6383 • Error messages - none.

#### 6384 • **Readaudit**

- 6385 • Purpose - presents the audit log to the administrator
- 6386 • Method of Use - **Readaudit**
- 6387 • Parameters - none
- 6388 • Parameter description - none
- 6389 • Actions - provides the text of the audit log
- 6390 • Error messages - none.

#### 6391 • **Setintaddr**

- 6392 • Purpose - sets the address of the internal address.
- 6393 • Method of Use - **Setintaddr** <address>
- 6394 • Parameters - address
- 6395 • Parameter description - first three fields of an IP address (as defined in RFC 791). For example:  
6396 123.123.123.
- 6397 • Actions - changes the internal value of the variable defining the internal network, the value of  
6398 which is used to judge attempted masquerades.

- 6399 • Error messages - “address in use”: indicates the identified internal network is the same as the  
6400 external network.
- 6401 • **Setextaddr**
- 6402 • Purpose - sets the address of the external address
- 6403 • Method of Use - **Setextaddr** <address>
- 6404 • Parameters - address
- 6405 • Parameter description - first three fields of an IP address (as defined in RFC 791). For example:  
6406 123.123.123.
- 6407 • Actions - changes the internal value of the variable defining the external network.
- 6408 • Error messages - “address in use”: indicates the identified external network is the same as the  
6409 internal network.

### 6410 **A.3ADV\_INT: Supplementary material on TSF internals**

6411 The wide variety of TOEs makes it impossible to codify anything more specific than “well-structured”  
6412 or “minimum complexity”. Judgements on structure and complexity are expected to be derived from  
6413 the specific technologies used in the TOE. For example, software is likely to be considered well-  
6414 structured if it exhibits the characteristics cited in the software engineering disciplines.

6415 This annex provides supplementary material on assessing the structure and complexity of procedure-  
6416 based software portions of the TSF. This material is based on information readily available in software  
6417 engineering literature. For other kinds of internals (e.g. hardware, non-procedural software such as  
6418 object-oriented code, etc.), corresponding literature on good practises should be consulted.

#### 6419 **A.3.1 Structure of procedural software**

6420 The structure of procedural software is traditionally assessed according to its *modularity*. Software  
6421 written with a modular design aids in achieving understandability by clarifying what dependencies a  
6422 module has on other modules (*coupling*) and by including in a module only tasks that are strongly  
6423 related to each other (*cohesion*). The use of modular design reduces the interdependence between  
6424 elements of the TSF and thus reduces the risk that a change or error in one module will have effects  
6425 throughout the TOE. Its use enhances clarity of design and provides for increased assurance that  
6426 unexpected effects do not occur. Additional desirable properties of modular decomposition are a  
6427 reduction in the amount of redundant or unneeded code.

6428 Minimising the amount of functionality in the TSF allows the evaluator as well as the developer to  
6429 focus only on that functionality which is necessary for SFR enforcement, contributing further to  
6430 understandability and further lowering the likelihood of design or implementation errors.

6431 The incorporation of modular decomposition, layering and minimisation into the design and  
6432 implementation process must be accompanied by sound software engineering considerations. A  
6433 practical, useful software system will usually entail some undesirable coupling among modules, some  
6434 modules that include loosely-related functions, and some subtlety or complexity in a module's design.  
6435 These deviations from the ideals of modular decomposition are often deemed necessary to achieve  
6436 some goal or constraint, be it related to performance, compatibility, future planned functionality, or  
6437 some other factors, and may be acceptable, based on the developer's justification for them. In applying

6438 the requirements of this class, due consideration must be given to sound software engineering  
6439 principles; however, the overall objective of achieving understandability must be achieved.

#### 6440 **A.3.1.1 Cohesion**

6441 Cohesion is the manner and degree to which the tasks performed by a single software module are  
6442 related to one another; types of cohesion include coincidental, communicational, functional, logical,  
6443 sequential, and temporal. These types of cohesion are characterised below, listed in the order of  
6444 decreasing desirability.

6445 a) *functional* cohesion - a module with functional cohesion performs activities related to a single  
6446 purpose. A functionally cohesive module transforms a single type of input into a single type of  
6447 output, such as a stack manager or a queue manager.

6448 b) *sequential* cohesion - a module with sequential cohesion contains functions each of whose output  
6449 is input for the following function in the module. An example of a sequentially cohesive module is  
6450 one that contains the functions to write audit records and to maintain a running count of the  
6451 accumulated number of audit violations of a specified type.

6452 c) *communicational* cohesion - a module with communicational cohesion contains functions that  
6453 produce output for, or use output from, other functions within the module. An example of a  
6454 communicationally cohesive module is an access check module that includes mandatory,  
6455 discretionary, and capability checks.

6456 d) *temporal* cohesion - a module with temporal cohesion contains functions that need to be executed  
6457 at about the same time. Examples of temporally cohesive modules include initialisation, recovery,  
6458 and shutdown modules.

6459 e) *logical (or procedural)* cohesion - a module with logical cohesion performs similar activities on  
6460 different data structures. A module exhibits logical cohesion if its functions perform related, but  
6461 different, operations on different inputs.

6462 f) *coincidental* cohesion - a module with coincidental cohesion performs unrelated, or loosely  
6463 related, activities.

#### 6464 **A.3.1.2 Coupling**

6465 Coupling is the manner and degree of interdependence between software modules; types of coupling  
6466 include call, common and content coupling. These types of coupling are characterised below, listed in  
6467 the order of decreasing desirability:

6468 a) *call*: two modules are call coupled if they communicate strictly through the use of their  
6469 documented function calls; examples of call coupling are data, stamp, and control, which are  
6470 defined below.

6471 1) *data*: two modules are data coupled if they communicate strictly through the use of call  
6472 parameters that represent single data items.

6473 2) *stamp*: two modules are stamp coupled if they communicate through the use of call  
6474 parameters that comprise multiple fields or that have meaningful internal structures.

6475 3) *control*: two modules are control coupled if one passes information that is intended to  
6476 influence the internal logic of the other.

6477 b) *common*: two modules are common coupled if they share a common data area or a common  
 6478 system resource. Global variables indicate that modules using those global variables are common  
 6479 coupled. Common coupling through global variables is generally allowed, but only to a limited  
 6480 degree. For example, variables that are placed into a global area, but are used by only a single  
 6481 module, are inappropriately placed, and should be removed. Other factors that need to be  
 6482 considered in assessing the suitability of global variables are:

6483 1) The number of modules that modify a global variable: In general, only a single module should  
 6484 be allocated the responsibility for controlling the contents of a global variable, but there may  
 6485 be situations in which a second module may share that responsibility; in such a case, sufficient  
 6486 justification must be provided. It is unacceptable for this responsibility to be shared by more  
 6487 than two modules. (In making this assessment, care should be given to determining the  
 6488 module actually responsible for the contents of the variable; for example, if a single routine is  
 6489 used to modify the variable, but that routine simply performs the modification requested by  
 6490 its caller, it is the calling module that is responsible, and there may be more than one such  
 6491 module). Further, as part of the complexity determination, if two modules are responsible for  
 6492 the contents of a global variable, there should be clear indications of how the modifications  
 6493 are coordinated between them.

6494 2) The number of modules that reference a global variable: Although there is generally no limit  
 6495 on the number of modules that reference a global variable, cases in which many modules  
 6496 make such a reference should be examined for validity and necessity.

6497 c) *content*: two modules are content coupled if one can make direct reference to the internals of the  
 6498 other (e.g. modifying code of, or referencing labels internal to, the other module). The result is that  
 6499 some or all of the content of one module are effectively included in the other. Content coupling can  
 6500 be thought of as using unadvertised module interfaces; this is in contrast to call coupling, which  
 6501 uses only advertised module interfaces.

### 6502 **A.3.2 Complexity of procedural software**

6503 Complexity is the measure of the decision points and logical paths of execution that code takes.  
 6504 Software engineering literature cites complexity as a negative characteristic of software because it  
 6505 impedes understanding of the logic and flow of the code. Another impediment to the understanding of  
 6506 code is the presence of code that is unnecessary, in that it is unused or redundant.

6507 The use of layering to separate levels of abstraction and minimise circular dependencies further  
 6508 enables a better understanding of the TSF, providing more assurance that the TOE security functional  
 6509 requirements are accurately and completely instantiated in the implementation.

6510 Reducing complexity also includes reducing or eliminating mutual dependencies, which pertains both  
 6511 to modules in a single layer and to those in separate layers. Modules that are mutually dependent may  
 6512 rely on one another to formulate a single result, which could result in a deadlock condition, or worse  
 6513 yet, a race condition (e.g., time of check vs. time of use concern), where the ultimate conclusion could  
 6514 be indeterminate and subject to the computing environment at the given instant in time.

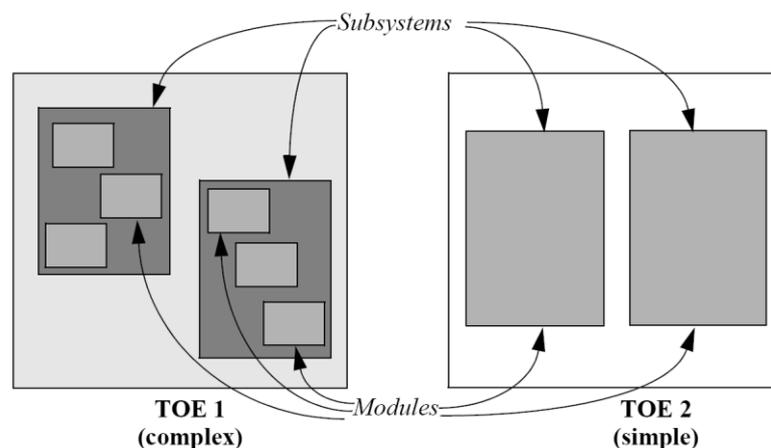
6515 Design complexity minimisation is a key characteristic of a reference validation mechanism, the  
 6516 purpose of which is to arrive at a TSF that is easily understood so that it can be completely analysed.  
 6517 (There are other important characteristics of a reference validation mechanism, such as TSF self-  
 6518 protection and non-bypassability; these other characteristics are covered by requirements in the  
 6519 ADV\_ARC family.)

6520 **A.4ADV\_TDS: Subsystems and Modules**

6521 This subclause provides additional guidance on the TDS family, and its use of the terms “subsystem”  
 6522 and “module”. This is followed by a discussion of how, as more-detailed becomes available, the  
 6523 requirement for the less-detailed is reduced.

6524 **A.4.1 Subsystems**

6525 Figure A.3 shows that, depending on the complexity of the TSF, the design may be described in terms  
 6526 of subsystems *and* modules (where subsystems are at a higher level of abstraction than modules); or it  
 6527 may just be described in terms of one level of abstraction (e.g., *subsystems* at lower assurance levels,  
 6528 *modules* at higher levels). In cases where a lower level of abstraction (modules) is presented,  
 6529 requirements levied on higher-level abstractions (subsystems) are essentially met by default. This  
 6530 concept is further elaborated in the discussion on subsystems and modules below.



6531

6532

**Figure A.3 — Subsystems and Modules**

6533 The developer is expected to describe the design of the TOE in terms of *subsystems*. The term  
 6534 “subsystem” was chosen to be specifically vague so that it could refer to units appropriate to the TOE  
 6535 (e.g., subsystems, modules). subsystems can even be uneven in scope, as long as the requirements for  
 6536 description of subsystems are met.

6537 The first use of subsystems is to distinguish the TSF boundary; that is, the portions of the TOE that  
 6538 comprise the TSF. In general, a subsystem is part of the TSF if it has the capability (whether by design  
 6539 or implementation) to affect the correct operation of any of the SFRs. For example, for software that  
 6540 depends on different hardware execution modes to provide domain separation (see A.1) where SFR-  
 6541 enforcing code is executed in one domain, then all subsystems that execute in that domain would be  
 6542 considered part of the TSF. Likewise, if a server outside that domain implemented an SFR (e.g.  
 6543 enforced an access control policy over objects it managed), then it too would be considered part of the  
 6544 TSF.

6545 The second use of subsystems is to provide a structure for describing the TSF at a level of description  
 6546 that, while describing how the TSF works, does not necessarily contain low-level implementation  
 6547 detail found in module descriptions (discussed later). subsystems are described at either a high level  
 6548 (lacking an abundance of implementation detail) or a detailed level (providing more insight into the  
 6549 implementation). The level of description provided for a subsystem is determined by the degree to  
 6550 which that subsystem is responsible for implementing an SFR.

6551 An *SFR-enforcing* subsystem is a subsystem that provides mechanisms for enforcing an element of any  
 6552 SFR, or directly supports a subsystem that is responsible for enforcing an SFR. If a subsystem provides  
 6553 (implements) an SFR-enforcing TSFI, then the subsystem is SFR-enforcing.

6554 Subsystems can also be identified as *SFR-supporting* and *SFR-non-interfering*. An SFR-supporting  
 6555 subsystem is one that is depended on by an SFR-enforcing subsystem in order to implement an SFR,  
 6556 but does not play as direct a role as an SFR-enforcing subsystem. An SFR-non-interfering subsystem is  
 6557 one that is not depended upon, in either a supporting or enforcing role, to implement an SFR.

#### 6558 **A.4.2 Modules**

6559 A module is generally a relatively small architectural unit that can be characterised in terms of the  
 6560 properties discussed in TSF internals (ADV\_INT). When both ADV\_TDS.3 Basic modular design (or  
 6561 above) requirements and TSF internals (ADV\_INT) requirements are present in a PP or ST, a “module”  
 6562 in terms of the TOE design (ADV\_TDS) requirements refers to the same entity as a “module” for the  
 6563 TSF internals (ADV\_INT) requirements. Unlike subsystems, modules describe the implementation in a  
 6564 level of detail that can serve as a guide to reviewing the implementation representation.

6565 It is important to note that, depending on the TOE, modules and subsystems may refer to the same  
 6566 abstraction. For ADV\_TDS.1 Basic design and ADV\_TDS.2 Architectural design (which do not require  
 6567 description at the module level) the subsystem description provides the lowest level detail available  
 6568 about the TSF. For ADV\_TDS.3 Basic modular design (which require module descriptions) these  
 6569 descriptions provide the lowest level of detail, while the subsystem descriptions (if they exist as  
 6570 separate entities) merely serve to put to the module descriptions in context. That is, it is not necessary  
 6571 to provide detailed subsystem descriptions if module descriptions exist. In TOEs that are sufficiently  
 6572 simple, a separate “subsystem description” is not necessary; the requirements can be met through  
 6573 documentation provided by modules. For complex TOEs, the purpose of the subsystem description  
 6574 (with respect to the TSF) is to provide the reader context so they can focus their analysis  
 6575 appropriately. This difference is illustrated in Figure A.3.

6576 An SFR-enforcing module is a module that completely or partially implements a security functional  
 6577 requirement (SFR) in the ST. Such modules may implement an SFR-enforcing TSFI, but some  
 6578 functionality expressed in an SFR (for example, audit and object re-use functionality) may not be  
 6579 directly tied to a single TSFI. As was the case with subsystems, SFR-supporting modules are those  
 6580 modules that are depended upon by an SFR-enforcing module, but are not responsible for directly  
 6581 implementing an SFR. SFR-non-interfering modules are those modules that do not deal, directly or  
 6582 indirectly, with the enforcement of SFRs.

6583 It is important to note that the determination of what “directly implements” means is somewhat  
 6584 subjective. In the narrowest sense of the term, it could be interpreted to mean the one or two lines of  
 6585 code that actually perform a comparison, zeroing operation, etc. that implements a requirement. A  
 6586 broader interpretation might be that it includes the module that is invoked in response to a SFR-  
 6587 enforcing TSFI, and all modules that may be invoked in turn by that module (and so on until the  
 6588 completion of the call). Neither of these interpretations is particularly satisfying, since the narrowness  
 6589 of the first interpretation may lead to important modules being incorrectly categorised as SFR  
 6590 supporting, while the second leads to modules that are actually not SFR-enforcing being classified as  
 6591 such.

6592 A description of a module should be such that one could create an implementation of the module from  
 6593 the description, and the resulting implementation would be 1) identical to the actual TSF  
 6594 implementation in terms of the interfaces presented, 2) identical in the use of interfaces that are  
 6595 mentioned in the design, and 3) functionally equivalent to the description of the purpose of the TSF  
 6596 module. For instance, RFC 793 provides a high-level description of the TCP protocol. It is necessarily

6597 implementation independent. While it provides a wealth of detail, it is *not* a suitable design  
 6598 description because it is not specific to an implementation. An actual implementation can add to the  
 6599 protocol specified in the RFC, and implementation choices (for example, the use of global data vs. local  
 6600 data in various parts of the implementation) may have an impact on the analysis that is performed.  
 6601 The design description of the TCP module would list the interfaces presented by the implementation  
 6602 (rather than just those defined in RFC 793), as well as an algorithm description of the processing  
 6603 associated with the modules implementing TCP (assuming they were part of the TSF).

6604 In the design, modules are described in detail in terms of the function they provide (the purpose); the  
 6605 interfaces they present (when required by the criteria); the return values from such interfaces; the  
 6606 interfaces (presented by other modules) they use (provided those interfaces are required to be also  
 6607 described); and a description of how they provide their functionality using a technique appropriate to  
 6608 the method used to implement the module.

6609 The purpose of a module should be described indicating what function the module is providing. It  
 6610 should be sufficient so that the reader could get a general idea of what the module's function is in the  
 6611 architecture.

6612 The interfaces presented by a module are those interfaces used by other modules to invoke the  
 6613 functionality provided. Interfaces include both *explicit* interfaces (e.g., a calling sequence invoked by  
 6614 other modules) as well as *implicit* interfaces (e.g., global data manipulated by the module). Interfaces  
 6615 are described in terms of how they are invoked, and any values that are returned. This description  
 6616 would include a list of parameters, and descriptions of these parameters. If a parameter were expected  
 6617 to take on a set of values (e.g., a "flag" parameter), the complete set of values the parameter could take  
 6618 on that would have an effect on module processing would be specified. Likewise, parameters  
 6619 representing data structures are described such that each field of the data structure is identified and  
 6620 described. Global data should be described to the extent required to understand their purpose. The  
 6621 level of description required for a global data structure needs to be identical to the one for module  
 6622 interfaces, where the input parameter and return values correspond to the individual fields and their  
 6623 possible values in the data structure. Global data structures may be described separate from the  
 6624 modules that manipulate or read them as long as the design of the modules contain sufficient  
 6625 information about the global data structures updated or the information extracted from global data  
 6626 structures.

6627 Note that different programming languages may have additional "interfaces" that would be non-  
 6628 obvious; an example would be operator/function overloading in C++. This "implicit interface" in the  
 6629 class description would also be described as part of the module design. Note that although a module  
 6630 could present only one interface, it is more common that a module presents a small set of related  
 6631 interfaces.

6632 When it is required to describe the interfaces used by a module, it must be clear from either the design  
 6633 description of the module or the purpose of the module called, what service is expected from the  
 6634 module called. For example if Module A is being described, and it uses Module B's bubble sort routine,  
 6635 the description of the interaction between modules must allow to identify why Module B's bubble sort  
 6636 routine is called and what this call contributes to the implementation of the SFRs. The interface and  
 6637 purpose of Module B's bubble sort routine must be described as part of the interfaces of Module B  
 6638 (provided the level of ADV\_TDS and the classification of Module B require a description its interfaces)  
 6639 and so Module A just needs to identify what data it needs to have sorted using this routine. An  
 6640 adequate description would be: "Module A invokes Module B's interface *double\_bubble()* to sort the  
 6641 usernames in alphabetical order".

6642 Note that if this sorting of the user names is not important for the enforcement of any SFR (e. g. it is  
 6643 just done to speed up things and an algorithmically identical implementation of Module A could also

6644 avoid to have the usernames sorted), the use of Module B's bubble sort routine is not SFR-enforcing  
 6645 and it is sufficient to explain in the description of Module A that the usernames are sorted in  
 6646 alphabetical order to enhance performance. Module B may be classified as "SFR-supporting" only and  
 6647 the level of ADV\_TDS chosen indicates if the interfaces of SFR-supporting modules need to be  
 6648 described or if it is sufficient to just describe the purpose of Module B.

6649 As discussed previously, the algorithmic description of the module should describe in an algorithmic  
 6650 fashion the implementation of the module. This can be done in pseudo-code, through flow charts, or  
 6651 (at ADV\_TDS.3 Basic modular design) informal text. It discusses how the module inputs and called  
 6652 functions are used to accomplish the module's function. It notes changes to global data, system state,  
 6653 and return values produced by the module. It is at the level of detail that an implementation could be  
 6654 derived that would be very similar to the actual implementation of the TOE.

6655 It should be noted that source code does not meet the module documentation requirements. Although  
 6656 the module design describes the implementation, it is *not* the implementation. The comments  
 6657 surrounding the source code might be sufficient documentation if they provide an explanation of the  
 6658 intent of the source code. In-line comments that merely state what each line of code is doing are  
 6659 useless because they provide no explanation of what the module is meant to accomplish.

6660 In the elements below, the labels (SFR-enforcing, SFR-supporting, and SFR-non-interfering) discussed  
 6661 for subsystems and modules are used to describe the amount and type of information that needs to be  
 6662 made available by the developer. The elements have been structured so that there is no expectation  
 6663 that the developer provide *only* the information specified. That is, if the developer's documentation of  
 6664 the TSF provides the information in the requirements below, there is no expectation that the  
 6665 developer update their documentation and label subsystems and modules as SFR-enforcing, SFR-  
 6666 supporting or SFR-non-interfering. The primary purpose of this labelling is to allow developers with  
 6667 less mature development methodologies (and associated artifacts, such as detailed interface and  
 6668 design documentation) to provide the necessary evidence without undue cost.

### 6669 **A.4.3 Levelling Approach**

6670 Because there is subjectivity in determining what is SFR-enforcing vs. SFR-supporting (and in some  
 6671 cases, even determining what is SFR-non-interfering the following paradigm has been adopted in this  
 6672 family. In early components of the family, the developer makes a determination about the  
 6673 classification of the subsystems into SFR-enforcing, etc., supplying the appropriate information, and  
 6674 there is little additional evidence for the evaluator to examine to support this claim. As the level of  
 6675 desired assurance increases, while the developer still makes a classification determination, the  
 6676 evaluator obtains more and more evidence that is used to confirm the developer's classification.

6677 In order to focus the evaluator's analysis on the SFR-related portions of the TOE, especially at lower  
 6678 levels of assurance, the components of the family are levelled such that initially detailed information is  
 6679 required only for SFR-enforcing architectural entities. As the level of assurance increases, more  
 6680 information is required for SFR-supporting and (eventually) SFR-non-interfering entities. It should be  
 6681 noted that even when complete information is required, it is not required that all of this information  
 6682 be analysed in the same level of detail. The focus should be in all cases on whether the *necessary*  
 6683 information has been provided and analysed.

6684 Table **AA.1** summarises the information required at each of the family components for the  
 6685 architectural entities to be described.

	TSF subsystem			TSF Module		
	SFR Enforce	SFR Support	SFR NI	SFR Enforce	SFR Support	SFR NI
ADV_TDS.1 Basic design (informal presentation)	structure, summary of SFR-Enf. behaviour, interactions	designation support <sup>11</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>12</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>13</sup>	purpose, all interfaces	purpose, all interfaces
ADV_TDS.6 Complete semiformal modular design with formal high-level design presentation	description, interactions	description, interactions	description, interactions	purpose, all interfaces	purpose, all interfaces	purpose, all interfaces

<sup>11</sup> *designation support* means that only documentation sufficient to support the classification of the subsystem / module is needed.

<sup>12</sup> *SFR interfaces* means that the module description contains, for each SFR-related interface, the returned values and the called interfaces to other modules.

<sup>13</sup> *All interfaces* means that the module description contains, for each interface, the returned values and the called interfaces to other modules.

	TSF subsystem			TSF Module		
	SFR Enforce	SFR Support	SFR NI	SFR Enforce	SFR Support	SFR NI
(semiformal presentation; additional formal presentation)						

6686

**Table AA.1 — Description Detail Levelling**6687 **A.4.4 Security relevance**

6688 The comments to WD2 regarding this chapter are pending as the contributor was not able to answer the  
6689 comments until the deadline.

6690 ISO/IEC 15408 (all parts) concentrates the description, the evidence and the analysis on the security  
6691 functionality of the TOE. This requires characterization of security relevance of functional and physical  
6692 parts of the TOE. Interfaces, subsystems and modules may be categorised (either implicitly or  
6693 explicitly) as “SFR-enforcing”, “SFR-supporting”, or “SFR-noninterfering”.

6694 The developer evidence and the evaluation analysis relates to the TOE and focus on the TSF and its  
6695 SFR-enforcing and SFR-supporting implementation. The security architecture description shall  
6696 demonstrate that the identified non-TSF subsystems of the TOE are not bypassing the TSF and the TSF  
6697 protects themselves against corruption by non-TSF code or entities. The developer shall describe the  
6698 SFR-noninterfering interfaces, subsystems and modules in the TOE design and demonstrate that they  
6699 do not interfere with the TSF because of their purposes, interactions or separation of resources.

6700 An interface, subsystem or module is

- 6701 • SFR-enforcing, if it directly implements an SFR.
- 6702 • SFR-supporting if it has to operate functionally correctly in order to support the proper  
6703 function of the SFRs.
- 6704 • SFR-non-interfering if it is not related to the implementation of the SFRs.

6705 The focus on security enforcing and security supporting functionality requires evidence of non-  
6706 interference of the other functionality. Even correct implemented security enforcing functions and  
6707 security mechanisms may be **bypassed, circumvented, deactivated, corrupted, or directly**  
6708 **attacked**. Non-interference implies that the TSF cannot be misused and unauthorized access to the  
6709 resources of the TSF implementation is prevented or impossible. Therefore the security architecture  
6710 aspects of non-bypassability and self-protection are critical if security relevance of interfaces,  
6711 subsystems and modules is categorized and this categorization is used in the vulnerability analysis.

6712 TSF self-protection is the security architecture property whereby the TSF cannot be corrupted by non-  
6713 TSF code or entities. This includes non-TSF subsystems of TOE and non-TOE parts of the IT product. It  
6714 is similar to the evidence for SFR-non-interfering subsystems/modules.

6715 The security domains are environments provided by the TSF for the use by untrusted entities in such a  
6716 way that these environments are isolated and protected from each other.

6717 Therefore the analysis of non-interference during evaluation requires examination of the security  
 6718 architecture of the TOE (ADV\_ARC) and may need more information on non-TSF subsystems than only  
 6719 the TOE structure in terms of subsystems as provided for ADV\_TDS.x.1. The developers shall provide a  
 6720 rationale that TSF is correctly defined and the analysis of SFR-non-interfering module in terms of its  
 6721 purpose and interaction with other modules

6722 • **purpose:** how a module provides their functionality, no further design decisions are needed.

6723 • **interaction:** reason that subsystems or modules communicate, and characterizes the  
 6724 information that is passed (less details than for interfaces).

6725 During evaluation non-interference shall be analyzed as part of the examination of functional  
 6726 specification and TOE design, and the vulnerability analysis. The categorization of interfaces,  
 6727 subsystems and modules as SFR-enforcing, SFR-supporting and SFR-noninterfering implies specific  
 6728 examination of the functional specification, design and testing. An interpretation of TSFI as all  
 6729 accessible external interfaces of the TSF would help this analysis. The functional tests of all TSF  
 6730 subsystems (beginning with ATE\_DPT.1) and all TSF modules (ATE\_DPT.3 and higher) should provide  
 6731 evidence for the correctness of their security categorization.

## 6732 A.5 Supplementary material on formal methods

6733 Formal methods provide a mathematical representation of the TSF and its behaviour and are required  
 6734 by the ADV\_FSP.6 Complete semi-formal functional specification with additional formal specification,  
 6735 ADV\_SPM.1 Formal TOE security policy model, and ADV\_TDS.6 Complete semiformal modular design  
 6736 with formal high-level design presentation components. There are two aspects of formal methods: the  
 6737 *specification language* that is used for formal expression, and the *theorem prover* that mathematically  
 6738 proves the completeness and correctness of the formal specification.

6739 A formal specification is expressed within a formal system based upon well-established mathematical  
 6740 concepts. These mathematical concepts are used to define well-defined semantics, syntax and rules of  
 6741 inference. A formal system is an abstract system of identities and relations that can be described by  
 6742 specifying a formal alphabet, a formal language over that alphabet which is based on a formal syntax,  
 6743 and a set of formal rules of inference for constructing derivations of sentences in the formal language.

6744 The evaluator should examine the identified formal systems to make sure that:

6745 • The semantics, syntax and inference rules of the formal system are defined or a definition is  
 6746 referenced.

6747 • Each formal system is accompanied by explanatory text that provides defined **semantics** so that:

6748 1) the explanatory text provides defined meanings of terms, abbreviations and acronyms that  
 6749 are used in a context other than that accepted by normal usage;

6750 2) the use of a formal system and semiformal notation use is accompanied by supporting  
 6751 explanatory text in informal style appropriate for unambiguous meaning;

6752 3) the formal system is able to express rules and characteristics of applicable SFPs, security  
 6753 functionality and interfaces (providing details of effects, exceptions and error messages) of  
 6754 TSF, their subsystems or modules to be specified for the assurance family for which the  
 6755 notations are used;

6756 4) the notation provides rules to determine the meaning of syntactical valid constructs.

- 6757 • Each formal system uses a formal syntax that provides rules to unambiguously recognise  
6758 constructs.
- 6759 • Each formal system provides proof rules which
- 6760 5) support logical reasoning of well-established mathematical concepts,
- 6761 6) help to prevent derivation of contradictions.
- 6762 If the developer uses a formal system which is already accepted by the evaluation authority the  
6763 evaluator can rely on the level of formality and strength of the system and focus on the instantiation of  
6764 the formal system to the TOE specifications and correspondence proofs.
- 6765 The formal style supports mathematical proofs of the security properties based on the security  
6766 features, the consistency of refinements and the correspondence of the representations. Formal tool  
6767 support seems adequate whenever manual derivations would otherwise become long winded and  
6768 incomprehensible. Formal tools are also apt to reduce the error probability inherent in manual  
6769 derivations.
- 6770 Examples of formal systems:
- 6771 • The **Z specification language** is highly expressive, and supports many different methods or styles  
6772 of formal specification. The use of Z has been predominantly for model-oriented specification,  
6773 using *schemas* to formally specify operations. See [https://en.wikipedia.org/wiki/Z\\_notation](https://en.wikipedia.org/wiki/Z_notation) for  
6774 more information.
- 6775 • **ACL2** is an open-source formal system comprising a LISP-based specification language and a  
6776 theorem prover. See <http://www.cs.utexas.edu/users/moore/acl2/> for further information.
- 6777 • **Isabelle** is a popular generic theorem proving environment that allows mathematical formulae to  
6778 be expressed in a formal language and provides tools for proving those formulae within a logical  
6779 calculus (see e.g. <http://www.cl.cam.ac.uk/Research/HVG/Isabelle/> for additional information).
- 6780 • The **B method** is a formal system based on the propositional calculus, the first order predicate  
6781 calculus with inference rules and set theory (see e.g. <https://en.wikipedia.org/wiki/B-Method> for  
6782 further information).
- 6783 • **NuSMV** (based on its predecessor SMV) is a symbolic model checker designed to be an open  
6784 architecture for model checking which can be reliably used for the verification of industrial  
6785 designs, as a core for custom verification tools, and as a testbed for formal verification techniques.  
6786 See <http://nusmv.fbk.eu/> for more information.
- 6787 • **Coq** is a formal proof management system that provides a formal language to write mathematical  
6788 definitions, executable algorithms and theorems together with an environment for semi-  
6789 interactive development of machine-checked proofs. See <https://coq.inria.fr/> for more  
6790 information.
- 6791 • **SystemVerilog** is a combined hardware description language and hardware verification language  
6792 based on Verilog.

## Annex B (informative)

### Composition (ACO)

6793  
6794  
6795  
6796

6797 The goal of this annex is to explain the concepts behind composition evaluations and the ACO criteria.  
6798 This annex does not define the ASE criteria; this definition can be found in clause 9.

#### 6799 **B.1 Necessity for composed TOE evaluations**

6800 The IT market is, on the whole, made up of vendors offering a particular type of product/technology.  
6801 Although there is some overlap, where a PC hardware vendor may also offer application software  
6802 and/or operating systems or a chip manufacturer may also develop a dedicated operating system for  
6803 their own chipset, it is often the case that an IT solution is implemented by a variety of vendors.

6804 There is sometimes a need for assurance in the combination (composition) of components in addition  
6805 to the assurance of the individual components. Although there is cooperation between these vendors,  
6806 in the dissemination of certain material required for the technical integration of the components, the  
6807 agreements rarely stretch to the extent of providing detailed design information and development  
6808 process/procedure evidence. This lack of information from the developer of a component on which  
6809 another component relies means that the dependent component developer does not have access to the  
6810 type of information necessary to perform an evaluation of both the dependent and base components at  
6811 EAL2 or above. Therefore, while an evaluation of the dependent component can still be performed at  
6812 any assurance level, to compose components with assurance at EAL2 or above it is necessary to reuse  
6813 the evaluation evidence and results of evaluations performed for the component developer.

6814 It is intended that the ACO criteria are applicable in the situation where one IT entity is dependent on  
6815 another for the provision of security services. The entity providing the services is termed the “base  
6816 component”, and that receiving the services is termed the “dependent component”. This relationship  
6817 may exist in a number of contexts. For example, an application (dependent component) may use  
6818 services provided by an operating system (base component). Alternatively, the relationship may be  
6819 peer-to-peer, in the sense of two linked applications, either running in a common operating system  
6820 environment, or on separate hardware platforms. If there is a dominant peer providing the services to  
6821 the minor peer, the dominant peer is considered to be the base component and the minor peer the  
6822 dependent component. If the peers provide services to each other in a mutual manner, each peer will  
6823 be considered to be the base component for the services offered and dependent component for the  
6824 services required. This will require iterations of the ACO components applying all requirements to  
6825 each type of component peer.

6826 The criteria are also intended to be more broadly applicable, stepwise (where a composed TOE  
6827 comprised of a dependent component and a base component itself becomes the base component of  
6828 another composed TOE), in more complex relationships, but this may require further interpretation.

6829 It is still required for composed TOE evaluations that the individual components are evaluated  
6830 independently, as the composition evaluation builds on the results of the individual component  
6831 evaluations. The evaluation of the dependent component may still be in progress when the composed  
6832 TOE evaluation commences. However, the dependent component evaluation must complete before the  
6833 composed TOE evaluation completes.

6834 The composed evaluation activities may take place at the same time as the dependent component  
6835 evaluation. This is due to two factors:

6836 a) Economic/business drivers - the dependent component developer will either be sponsoring the  
6837 composition evaluation activities or supporting these activities as the evaluation deliverables from  
6838 the dependent component evaluation are required for composed evaluation activities.

6839 b) Technical drivers - the components consider whether the requisite assurance is provided by the  
6840 base component (e.g. considering the changes to the base component since completion of the  
6841 component evaluation) with the understanding that the dependent component has recently  
6842 undergone (is undergoing) component evaluation and all evaluation deliverables associated with  
6843 the evaluation are available. Therefore, there are no activities during composition requesting the  
6844 dependent component evaluation activities to be re-verified. Also, it is verified that the base  
6845 component forms (one of) the test configurations for the testing of the dependent component  
6846 during the dependent component evaluation, leaving ACO\_CTT to consider the base component in  
6847 this configuration.

6848 The evaluation evidence from the evaluation of the dependent component is required input into the  
6849 composed TOE evaluation activities. The only evaluation material from the evaluation of the base  
6850 component that is required as input into the composed TOE evaluation activities:

6851 a) Residual vulnerabilities in the base component, as reported during the base component  
6852 evaluation. This is required for the ACO\_VUL activities.

6853 No other evaluation evidence from the base component activities should be required for the composed  
6854 TOE evaluation, as the evaluation results from the component evaluation of the base component  
6855 should be reused. Additional information about the base component may be required if the composed  
6856 TOE TSF includes more of the base component than was considered to be TSF during component  
6857 evaluation of the base component.

6858 The component evaluation of the base and dependent components are assumed to be complete by the  
6859 time final verdicts are assigned for the ACO components.

6860 The ACO\_VUL components only consider resistance against an attacker with an attack potential up to  
6861 Enhanced-Basic. This is due to the level of design information that can be provided of how the base  
6862 component provides the services on which the dependent component relies through application of the  
6863 ACO\_DEV activities. Therefore, the confidence arising from composed TOE evaluations using CAPs is  
6864 limited to a level similar to that obtained from EAL4 component TOE evaluations. Although assurance  
6865 in the components that comprise the composed TOE may be higher than EAL4.

## 6866 **B.2 Performing Security Target evaluation for a composed TOE**

6867 An ST will be submitted by the developer for the evaluation of the composed (base component +  
6868 dependent component) TOE. This ST will identify the assurance package to be applied to the  
6869 composed TOE, providing assurance in the composed entity by drawing upon the assurance gained in  
6870 the component evaluations.

6871 The purpose of considering the composition of components within an ST is to validate the  
6872 compatibility of the components from the point of view of both the environment and the requirements,  
6873 and also to assess that the composed TOE ST is consistent with the component STs and the security  
6874 policies expressed within them. This includes determining that the component STs and the security  
6875 policies expressed within them are compatible.

6876 The composed TOE ST may refer out to the content of the component STs, or the ST author may chose  
 6877 to reiterate the material of the component STs within the composed TOE ST providing a rationale of  
 6878 how the component STs are represented in the composed TOE ST.

6879 During the conduct of the ASE\_CCL evaluation activities for a composed TOE ST the evaluator  
 6880 determines that the component STs are accurately represented in the composed TOE ST. This is  
 6881 achieved through determining that the composed TOE ST demonstrably conforms to the component  
 6882 TOE STs. Also, the evaluator will need to determine that the dependencies of the dependent  
 6883 component on the operational environment are adequately fulfilled in the composed TOE.

6884 The composed TOE description will describe the composed solution. The logical and physical scope  
 6885 and boundary of the composed solution will be described, and the logical boundary(ies) between the  
 6886 components will also be identified. The description will identify the security functionality to be  
 6887 provided by each component.

6888 The statement of SFRs for the composed TOE will identify which component is to satisfy an SFR. If an  
 6889 SFR is met by both components, then the statement will identify which component meets the different  
 6890 aspects of the SFR. Similarly the composed TOE Summary Specification will identify which component  
 6891 provides the security functionality described.

6892 The package of ASE: Security Target evaluation requirements applied to the composed TOE ST should  
 6893 be consistent with the package of ASE: Security Target evaluation requirements used in the  
 6894 component evaluations.

6895 Reuse of evaluation results from the evaluation of component STs can be made in the instances that  
 6896 the composed TOE ST directly refers to the component STs. e.g. if the composed TOE ST refers to a  
 6897 component ST for part of its statement of SFRs, the evaluator can understand that the requirement for  
 6898 the completion of all assignment and selection operations (as stated in ASE\_REQ.\*.3C has been  
 6899 satisfied in the component evaluations.

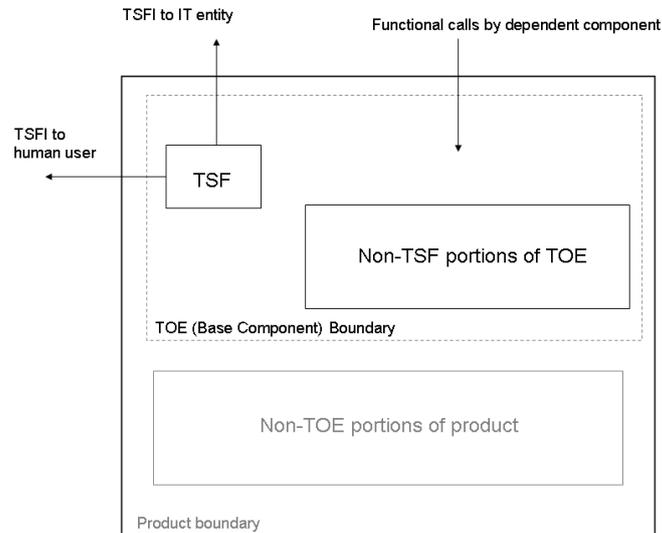
### 6900 **B.3 Interactions between composed IT entities**

6901 The TSF of the base component is often defined without knowledge of the dependencies of the  
 6902 possible applications with which it may be composed. The TSF of this base component is defined to  
 6903 include all parts of the base component that have to be relied upon for enforcement of the base  
 6904 component SFRs. This will include all parts of the base component required to implement the base  
 6905 component SFRs.

6906 The TSFI of this base component represents the interfaces provided by the TSF to the external entities  
 6907 defined in the statement of SFRs to invoke a service of the TSF. This includes interfaces to the human  
 6908 user and also interfaces to external IT entities. However, the TSFI only includes those interfaces to the  
 6909 TSF, and therefore is not necessarily an exhaustive interface specification of all possible interfaces  
 6910 available between an external entity and the base component. The base component may present  
 6911 interfaces to services that were not considered security-relevant, either because of the inherent  
 6912 purpose of the service (e.g., adjust type font) or because associated ISO/IEC 15408 SFRs are not being  
 6913 claimed in the base component's ST (e.g. the login interface when no FIA: Identification and  
 6914 authentication SFRs are claimed).

6915 The functional interfaces provided by the base component are in addition to the security interfaces  
 6916 (TSFIs), and are not required to be considered during the base component evaluation. These often  
 6917 include interfaces that are used by a dependent component to invoke a service provided by the base  
 6918 component.

6919 The base component may include some indirect interfaces through which TSFIs may be called, e.g.  
 6920 APIs that can be used to invoke a service of the TSF, which were not considered during the evaluation  
 6921 of the base component.



6922

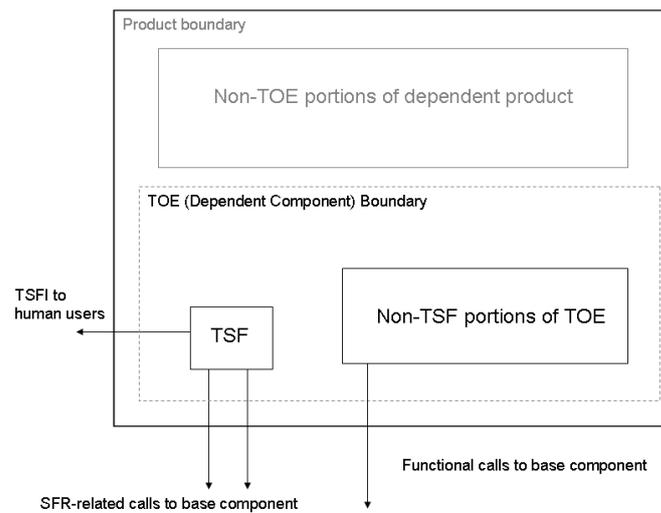
6923

**Figure BB.1 — Base component abstraction**

6924 The dependent component, which relies on the base component, is similarly defined: interfaces to  
 6925 external entities defined in the SFRs of the component ST are categorised as TSFI and are examined in  
 6926 ADV\_FSP.

6927 Any call out from the dependent TSF to the environment in support of an SFR will indicate that the  
 6928 dependent TSF requires some service from the environment in order to satisfy the enforcement of the  
 6929 stated dependent component SFRs. Such a service is outside the dependent component boundary and  
 6930 the base component is unlikely to be defined in the dependent ST as an external entity. Hence, the calls  
 6931 for services made out by the dependent TSF to its underlying platform (the base component) will not  
 6932 be analysed as part of the Functional specification (ADV\_FSP) activities. These dependencies on the  
 6933 base component are expressed in the dependent component ST as security objectives for the  
 6934 environment.

6935 This abstraction of the dependent component and the interfaces is shown in Figure BB.2 below.



6936

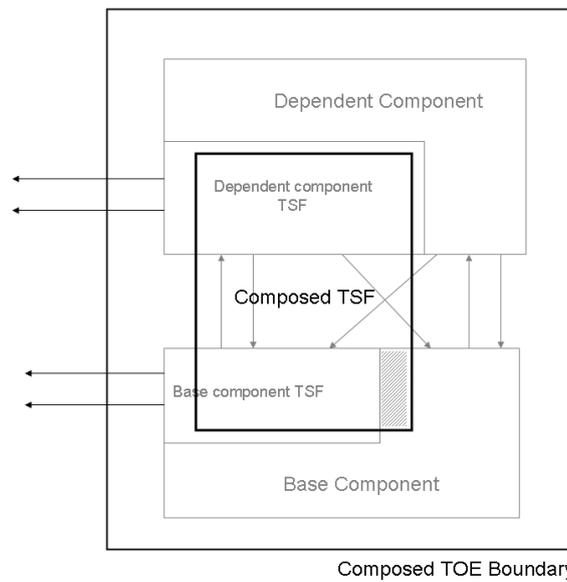
6937

**Figure BB.2 — Dependent component abstraction**

6938 When considering the composition of the base component and the dependent component, if the  
 6939 dependent component's TSF requires services from the base component to support the  
 6940 implementation of the SFR, the interface to the service will need to be defined. If that service is  
 6941 provided by the base component's TSF, then that interface should be a TSFI of the base component and  
 6942 will therefore already be defined within the functional specification of the base component.

6943 If, however, the service called by the dependent component's TSF is not provided by the TSF of the  
 6944 base component (i.e., it is implemented in the non-TSF portion of the base component or possibly even  
 6945 in the non-TOE portion of the base component (not illustrated in Figure B.3), there is unlikely to be a  
 6946 TSFI of the base component relating to the service, unless the service is mediated by the TSF of the  
 6947 base component. The interfaces to these services from the dependent component to the operational  
 6948 environment are considered in the family Reliance of dependent component (ACO\_REL).

6949 The non-TSF portion of the base component is drawn into the TSF of the composed TOE due to the  
 6950 dependencies the dependent component has on the base component to support the SFRs of the  
 6951 dependent component. Therefore, in such cases, the TSF of the composed TOE would be larger than  
 6952 simply the sum of the components' TSFs.



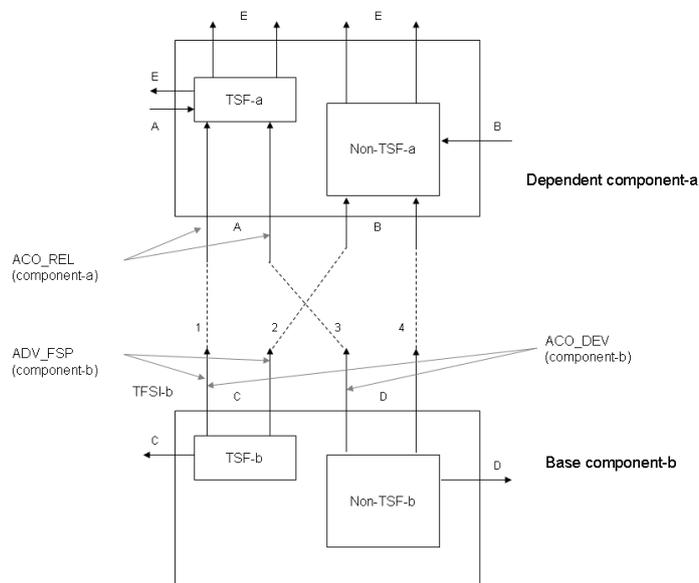
6953

6954

**Figure B.3 — Composed TOE abstraction**

6955 It may be the case that the base component TSFI is being called in a manner that was unforeseen in the  
 6956 base component evaluation. Hence there would be a requirement for further testing of the base  
 6957 component TSFI.

6958 The possible interfaces are further described in the following diagram (Figure B.4) and supporting  
 6959 text.



6960

6961

**Figure B.4 — Composed component interfaces**

6962 a) Arrows going into 'dependent component-a' (A and B) = where the component expects the  
 6963 environment to respond to a service request (responding to calls out from dependent component  
 6964 to the environment);

6965 b) Arrows coming *out* of 'base component-b' (C and D) = interfaces of services provided by the base  
6966 component to the environment;

6967 c) Broken lines between components = types of communication between pairs of interfaces;

6968 d) The other (grey) arrows = interfaces that are described by the given criteria.

6969 The following is a simplification, but explains the considerations that need to be made.

6970 There are components a ('dependent component-a') and b ('base component-b'): the arrows coming  
6971 *out* of TSF-a are services provided by TSF-a and are therefore TSFIs(a); likewise, the arrows coming  
6972 *out* of TSF-b ("C") are TSFIs(b). These are each detailed in their respective functional specs.  
6973 component-a is such that it requires services from its environment: those needed by the TSF(a) are  
6974 labelled "A"; the other (not related to TSF-a) services are labelled "B".

6975 When component-a and component-b are combined, there are four possible combinations of {services  
6976 needed by component-a} and {services provided by component-b}, shown as broken lines (types of  
6977 communication between pairs of interfaces). Any set of these might exist for a particular composition:

6978 a) TSF-a needs those services that are provided by TSF-b ("A" is connected to "C"): this is  
6979 straightforward: the details about "C" are in the FSP for component-b. In this instance the  
6980 interfaces should all be defined in the functional specifications for the component-b.

6981 b) Non-TSF-a needs those services that are provided by TSF-b ("B" is connected to "C"): this is  
6982 straightforward (again, the details about "C" are in the FSP for component-b), but unimportant:  
6983 security-wise.

6984 c) Non-TSF-a needs those services that are provided by non-TSF-b ("B" is connected to "D"): we have  
6985 no details about D, but there are no security implications about the use of these interfaces, so they  
6986 do not need to be considered in the evaluation, although they are likely to be an integration issue  
6987 for the developer.

6988 d) TSF-a needs those services that are provided by non-TSF-b ("A" is connected to "D"): this would  
6989 arise when component-a and component-b have different senses of what a "security service" is.  
6990 Perhaps component-b is making no claims about I&A (has no FIA SFRs in its ST), but component-a  
6991 needs authentication provided by its environment. There are no details about the "D" interfaces  
6992 available (they are not TSFI (b), so they are not in component-b's FSP).

6993 Note: if the kind of interaction described in case d above exists, then the TSF of the composed TOE  
6994 would be TSF-a + TSF-b + Non-TSF-b. Otherwise, the TSF of the composed TOE would be TSF-a + TSF-  
6995 b.

6996 Interfaces types 2 and 4 of Figure B.4 are not directly relevant to the evaluation of the composed TOE.  
6997 Interfaces 1 and 3 will be considered during the application of different families:

6998 a) Functional specification (ADV\_FSP) (for component-b) will describe the C interfaces.

6999 b) Reliance of dependent component (ACO\_REL) will describe the A interfaces.

7000 c) Development evidence (ACO\_DEV) will describe the C interfaces for connection type 1 and the D  
7001 interfaces for connection type 3.

7002 A typical example where composition may be applied is a database management system (DBMS) that  
7003 relies upon its underlying operating system (OS). During the evaluation of the DBMS component, there

7004 will be an assessment made of the security properties of that DBMS (to whatever degree of rigour is  
7005 dictated by the assurance components used in the evaluation): its TSF boundary will be identified, its  
7006 functional specification will be assessed to determine whether it describes the interfaces to the  
7007 security services provided by the TSF, perhaps additional information about the TSF (its design,  
7008 architecture, internal structure) will be provided, the TSF will be tested, aspects of its life-cycle and its  
7009 guidance documentation will be assessed, etc.

7010 However, the DBMS evaluation will not call for any evidence concerning the dependency the DBMS has  
7011 on the OS. The ST of the DBMS will most likely state assumptions about the OS in its Assumptions  
7012 subclause and state security objectives for the OS in its Environment subclause. The DBMS ST may  
7013 even instantiate those objectives for the environment in terms of SFRs for the OS. However, there will  
7014 be no specification for the OS that mirrors the detail in the functional specification, architecture  
7015 description, or other ADV evidence as for the DBMS. Reliance of dependent component (ACO\_REL) will  
7016 fulfil that need.

7017 Reliance of dependent component (ACO\_REL) describes the interfaces of the dependent TOE that make  
7018 the calls to the base component for the provision of services. These are the interfaces to which the  
7019 base component is to respond. The interface descriptions are provided from the dependent  
7020 component's viewpoint.

7021 Development evidence (ACO\_DEV) describes the interfaces provided by the base component, which  
7022 respond to the dependent component service requests. These interfaces are mapped to the relevant  
7023 dependent component interfaces that are identified in the reliance information. (The completeness of  
7024 this mapping, whether the base component interfaces described represent all dependent component  
7025 interfaces, is not verified here, but in Composition rationale (ACO\_COR)). At the higher levels of  
7026 ACO\_DEV the subsystems providing the interfaces are described.

7027 Any interfaces required by the dependent component that have not been described for the base  
7028 component are reported in the rationale for Composition rationale (ACO\_COR). The rationale also  
7029 reports whether the interfaces of the base component on which the dependent component relies were  
7030 considered within the base component evaluation. For any interfaces that were not considered in the  
7031 base component evaluation, a rationale is provided of the impact of using the interface on the base  
7032 component TSF.

## Annex C (informative)

### Cross reference of assurance component dependencies

7037 The dependencies documented in the components of Clauses 7 and 9-15 are the direct dependencies  
7038 between the assurance components.

7039 The following dependency tables for assurance components show their direct, indirect and optional  
7040 dependencies. Each of the components that is a dependency of some assurance component is allocated  
7041 a column. Each assurance component is allocated a row. The value in the table cell indicate whether  
7042 the column label component is directly required (indicated by a cross "X") or indirectly required  
7043 (indicated by a dash "-"), by the row label component. If no character is presented, the component is  
7044 not dependent upon another component.

7045

	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	ALC_TDA.1	ALC_TDA.2	ALC_TDA.3	
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																					

7046

**Table C.1 — Dependency table for Class ADV: Development**

7047

7048

	ADV_FSP.1
AGD_OPE.1	X
AGD_PRE.1	

7049  
7050

**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_CMS.3	ALC_DVS.1	ALC_DVS.2	ALC_LCD.1	ALC_TAT.1
ALC_CMC.1						X					
ALC_CMC.2						X					
ALC_CMC.3						X		X		X	
ALC_CMC.4						X		X		X	
ALC_CMC.5						X			X	X	
ALC_CMS.1											
ALC_CMS.2											
ALC_CMS.3											
ALC_CMS.4											
ALC_CMS.5											
ALC_DEL.1											
ALC_DVS.1											
ALC_DVS.2											
ALC_FLR.1											
ALC_FLR.2											
ALC_FLR.3											
ALC_LCD.1											
ALC_LCD.2											
ALC_TDA.1											
ALC_TDA.2							X				
ALC_TDA.3					-						X
ALC_TAT.1	-	-	X	-	-						-
ALC_TAT.2	-	-	X	-	-						-
ALC_TAT.3	-	-	X	-	-						-
ALC_MOK.1											
ALC_COMP.1											

7051  
7052

**Table C.3 — Dependency table for Class ALC: Life-cycle support**

	APE_SPD.1	APE_REQ.1	APE_OBJ.2	APE_INT.1	APE_ECD.1
APE_CCL.1	X	X			
APE_ECD.1					
APE_INT.1					
APE_OBJ.1					
APE_OBJ.2					X
APE_REQ.1	X				
APE_REQ.2	X		X		-
APE_SPD.1					

Table C.4 — Dependency table for Class APE: Protection Profile evaluation

7053

7054

	ACE_SPD.1	ACE_REQ.1	ACE_OBJ.1	ACE_MCO.2	ACE_INT.1	ACE_ECD.1
ACE_CCL.1	X	X	-			
ACE_CCO.1	-	X	X	-		
ACE_ECD.1						
ACE_INT.1						
ACE_MCO.1	-	X	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

7055

7056

	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									

7057  
7058**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																

7059  
7060**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															

7061

**Table C.8 Dependency table for Class AVA: Vulnerability assessment**



7066

## Annex D