

1 **Conformance Requirements for Specifications**
2 **Version 0.5**
3 **March 1, 2002**

4
5
6 **Editors:**

7 Lynne Rosenthal (lynne.rosenthal@nist.gov)
8 Mark Skall (mark.skall@nist.gov)
9

10 **Contributors:**

11 Lofton Henderson (lofton@rockynet.com)
12 David Marston (david.marston@lotus.com)
13

14 **Abstract**

15 Document describes how to specify software conformance to a specification and
16 identifies the conformance requirements that must to be included and addressed in
17 specifications. Target audience is primarily specification developers, followed by
18 conformance test suite developers and implementation developers.
19

20 **Status of this Document**

21 Working Draft – v0.5
22

23 **Document Version History**

24 28 Feb 2002 updated based on Feb 15 Telcon and issues on capitalization and
25 deprecation; italicize terms that are being defined
26 30 Dec 2001 updated based on Dec 13 F2F, Telcon
27 21 Nov 2001 updated based on input from QAWG
28 25 Oct 2001 updated based on Sept 13 Telecon
29 22 Aug 2001 updated based on Aug 16 Telecon.
30 2 Aug 2001 initial draft
31

32 **Reference Documents**

33 ISO Guide 2: Standardization and related activities – General vocabulary.
34 ISO/IEC Directives Part 3: Rules for the structure and drafting of International Standards
35 [ebXML Technical Architecture Specification](#), Conformance Clause
36 [OASIS/ebXML Registry Services Specification](#)
37 W3C WAI Guidelines
38 W3C XSLT/Xpath Recommendation
39 [UNICODE](#)
40 [SAML specifications](#)
41

41	TABLE of CONTENTS
42	
43	1. Introduction
44	2. Scope and Audience
45	3. Conformance
46	4. Normative references
47	5. Informative references
48	6. Terms and definitions
49	7. Conformance Clause
50	7.1. Rationale for a conformance clause
51	7.2. Conformance keywords
52	7.3. General principles
53	8. What to Address in a Conformance Clause
54	8.1. What needs to conform
55	8.1.1. Modularity of (software)
56	8.1.2. Specifying conformance claims
57	8.2. Profiles and Levels
58	8.3. Extensions
59	8.3.1. Disallow Extensions
60	8.3.2. Allow Extensions
61	8.4. Discretionary Items
62	8.4.1. Implementation dependent values
63	8.4.2. Alternate approaches
64	8.5. Internationalization – Languages and Character sets
65	9. Additional Issues to Address
66	9.1. Implementation conformance statement (questionnaire)
67	9.2. Test Assertions
68	9.3. Specify a testing methodology or program
69	10 Conformance Claim
70	Appendix A: Sample Conformance Claims
71	Appendix B: Profiles
72	Appendix C: Extensions
73	C.1 Mechanism to allow extensions –
74	C.2 Registration of implementer extensions or implementation defined values
75	
76	
77	

77 **1. Introduction**

78
79 The objective of this document is to provide guidance on how to specify conformance
80 and communicate requirements for claiming conformance in specifications. A primary
81 goal is to improve the quality of specifications and resulting implementations. Good
82 specifications lead to better implementations and foster the development of conformance
83 test suites and tools. The document identifies the conformance requirements that shall be
84 included or addressed in specifications. Conformance requirements are the expression
85 that conveys the criteria to be fulfilled in an implementation of a specification [ISO
86 Guide 2]. The conformance requirements are stated in a conformance clause or
87 statements within the specification. This document describes the purpose and scope of a
88 conformance clause, associated issues that a conformance clause shall address as well as
89 issues that a conformance clause may address. Where ever possible, sample text and
90 examples will be given.

91
92 The information contained is produced as the result of extensive experience in the
93 development and implementation of conformance clauses and test suites for consensus
94 standards and specifications. It is based on the principles and requirements prescribed by
95 international standards (e.g., ISO/IEC and IEEE) as well as extractions from ebXML,
96 OASIS and W3C specifications.

98 **2. Scope and Audience**

99
100 This document specifies the general requirements and definitions concerning
101 conformance and related issues. It is intended to fundamentally contribute towards
102 mutual understanding amongst developers of specifications and conformance test suites
103 and tools. It is also intended to provide a suitable source for teaching and for reference,
104 briefly covering basic theoretical and practical principles of conformance.

105
106 This document will not define specific conformance requirements for any specific
107 specification – this is the responsibility of committees chartered to develop specifications.

108
109 This document is intended primarily for the developers of specifications to help enable
110 them to develop conformance requirements within their specification and to create a
111 testable, unambiguous specification. Secondary audiences include, but are not limited to:
112 developers of conformance test suites, software implementers, international standards
113 bodies, and other industry organizations.

115 **3. Conformance**

116
117 A specification that conforms to this document SHALL:

- 118 • contain a conformance clause,
- 119 • use the conformance keywords (section 7.2),
- 120 • address all issues (topics) in section 8 and indicate the applicability and means
121 for achieving conformance to each issue,

- 122 • examine the issues in section 9, determine if each issue is applicable and define
123 the conformance requirements for applicable items.

124
125 The location of the conformance clause SHALL be clearly identifiable from the table of
126 contents and any relevant index. The conformance clause SHOULD exist as a separate
127 section within the specification, so that it is clearly identifiable, allowing a reader to find
128 all conformance provisions from a single starting point.

129
130 Each issue in section 8 SHALL be addressed by the specification. When alternate
131 approaches are allowed, the specification SHALL clearly describe the disposition of each
132 issue. For example, if a specification does not contain levels it should be clear to the
133 reader that levels are not supported. One method to ensure this clarity is to explicitly
134 state that levels are not supported.

135

136 **4. Normative references**

137

138 The following normative documents contain provisions, which through reference in this
139 text constitute provisions of this document. At the time of publications, the editions
140 indicated below were valid. All standards are subject to revision, and parties to
141 agreements based on this document are encouraged to investigate the possibility of
142 applying the most recent editions of the standards indicated below.

143

144 ISO/IEC Guide 2: Standardization and related activities – General vocabulary
145 ISO/IEC Directives Part 3: Rules for the structure and drafting of International Standards.
146 [RFC 2119](#): Keywords for use in RFC's to Indicate Requirement Levels
147 [UNICODE Standard, version 3.0](#), Addison Wesley, Reading MA, 2000, ISBN: 0-201-
148 61633-5

149

150 **5. Informative references**

151

152 Carnahan, Rosenthal, Skall, *Conformance Testing and Certification Model for Software*
153 *Specifications*, ISACC Conference 1998, March 1998.

154

155 Glossary of Conformance Terminology, [http://www.oasis-](http://www.oasis-open.org/committees/ioc/glossary.htm)
156 [open.org/committees/ioc/glossary.htm](http://www.oasis-open.org/committees/ioc/glossary.htm).

157

158 Rosenthal, Brady, *What is this thing called conformance?*, NIST/ITL Bulletin, January
159 2001, <http://www.itl.nist.gov/div897/ctg/conformance/bulletin-conformance.htm>.

160

161 Rosenthal, Skall, *Software Validation*, Encyclopedia of Software Engineering, edited by
162 J. Marciniak, Wiley, December 2001.

163

164 **6. Terms and definitions**

165

166 For the purposes of this document and specifications implementing this document, the
167 following relevant terms and definitions SHALL apply:

168
169 **Accreditation** – procedure by which an authoritative body gives formal recognition that
170 a body or person is competent to carry out specific tasks.

171 **Certification** – the acknowledgement that a validation has been completed and the
172 criteria established by the certifying organization has been met.

173 **Conformance** – the fulfillment of a product, process, or service of specified
174 requirements.

175 **Conformance Testing** – a method of verifying implementations of a specification to
176 determine whether or not deviations from the specification exist.

177 **Implementation** – the realization of a specification – it can be a software product,
178 system, program, protocol, application, or document instance.

179 **Strict Conformance** – conformance of an implementation that employs only the
180 requirements and/or functionality defined in the specification and no more (i.e., no
181 extensions to the specification are implemented).

182 **Validation** – the process of testing software for conformance to a specific specification.

183

184 **7. Conformance Clause**

185

186 Every specification SHALL contain a conformance clause.

187

188 The conformance clause is a part or collection of parts of a specification that defines the
189 requirements, criteria, or conditions that must be satisfied by an implementation in order
190 to claim conformance. The conformance clause identifies *what* must conform and *how*
191 conformance can be met. Typically the conformance clause is a high-level description of
192 what is required of implementers and applications. It may refer to other parts of the
193 standard. It may specify sets of functions, which may take the form of profiles, levels, or
194 other structures. It may specify minimal requirements for certain functions and for
195 implementation-dependent values. Additionally, it may specify the permissibility of
196 extensions, options, and alternative approaches and how they are to be handled.

197 **7.1. Rationale for a conformance clause**

198 A conformance clause:

- 199 • promotes a common understanding of conformance and what is required to claim
200 conformance to a specification,
- 201 • facilitates consistent application of conformance within a specification,
- 202 • facilitates consistent application of conformance across related specifications,
- 203 • promotes interoperability and open interchange,
- 204 • encourages the use of applicable conformance test suites,
- 205 • promotes uniformity in the development of conformance test suites.

206 **7.2. Conformance keywords**

207 There are specific words that are used throughout the specification to denote whether or
208 not requirements are mandatory, optional, or suggested. Using these keywords helps to
209 identify the testable statements in a specification. Although the keywords used within the

210 ISO/IEC community differ from the keywords used within the IETF communities, they
211 achieve the same results. Use of these keywords SHOULD be consistent (i.e., use the
212 ISO keywords or the IETF keywords, but do not use both).

213

214 ISO Keywords:

215 SHALL – to indicate requirements strictly to be followed in order to conform to
216 the standard and in which no deviation is permitted. Equivalent expressions
217 include: is to, is required to, has to, it is necessary. Do not use MUST as an
218 alternative for shall.

219 SHALL NOT - converse of SHALL.

220 SHOULD – to indicate that among several possibilities one is recommended as
221 particularly suitable, without mentioning or excluding others.

222 SHOULD NOT – converse of SHOULD.

223 MAY – to indicate a course of action permissible within the limits of the standard.

224 Equivalent expressions include: is permitted, is allowed.

225 NEED NOT – to indicate a course of action is not required.

226 CAN – statement of possibility and capability, whether material, physical or
227 causal. Equivalent expressions include: be able to, it is possible to.

228 CANNOT – converse of CAN.

229

230 IETF Keywords (RFC2119)

231 MUST - the requirement is an absolute requirement of the specification.

232 MUST NOT – the requirement is an absolute prohibition of the specification.

233 REQUIRED – see MUST.

234 SHALL – see MUST.

235 SHALL NOT – see MUST NOT.

236 SHOULD – there may exist valid reasons in particular circumstances to ignore a
237 particular item, but the full implications must be understood and carefully
238 weighed before choosing a difference course.

239 SHOULD NOT – there may exist valid reasons in particular circumstances when
240 the particular behavior is acceptable or even useful, but the full implications
241 should be understood and the case carefully weighed before implementing any
242 behavior described with this label.

243 RECOMMENDED – see SHOULD.

244 MAY - the item is truly optional. One vendor may choose to include the item
245 because a particular marketplace requires it or because the vendor feels that it
246 enhances the product while another vendor may omit the same item. An
247 implementation that does not include a particular option MUST be prepared to
248 interoperate with another implementation that does include the option, though
249 perhaps with reduced functionality. In the same vein an implementation, which
250 does include a particular option MUST be prepared to interoperate with another
251 implementation that does not include the option (except, of course, for the feature
252 the option provides.)

253

254 Additionally keywords include:

255 NORMATIVE – statements provided for the prescriptive parts of the
256 specification, providing that which is necessary in order to be able to claim
257 conformance to the specification. Note: the conformance scheme of a
258 specification can allow claimants to exempt certain normative provisions as long
259 as the claim discloses the exemption.
260 INFORMATIVE (NON-NORMATIVE) –statements provided for informational
261 purposes, intended to assist the understanding or use of the specification and shall
262 not contain provisions that are required for conformance.

263 **7.3. General principles**

264 An objective of any conformance clause and its related conformance statements is to
265 provide clear and unambiguous statements, so that the reader knows what is required in
266 order to claim conformance and what is optional. To achieve this objective:

- 267 • normative and informative sections SHALL be evident and if necessary, labeled
268 accordingly,
- 269 • uniformity of structure, of style, and terminology SHALL be maintained within the
270 specification,
- 271 • identical wording SHALL be used to express identical provisions and analogous
272 wording SHALL be used to express analogous provisions.

273

274 **8. What to Address in a Conformance Clause**

275 **8.1. What needs to conform**

276 The conformance clause identifies the “class of products” (i.e., object of the claim) that
277 will be developed, where “class of product” may be an implementation, application,
278 service, and/or protocol (e.g., content, user agent, authoring tool). Additionally, the
279 clause specifies the conditions that SHALL be satisfied in order to claim conformance for
280 that class of product (i.e., make a valid claim). It may also specify that which is not a
281 requirement. There may be several classes of products that are identified, each with its
282 own conformance statement or set of conformance criteria.

283

284 Example 1: The [OASIS/ebXML Registry Services Specification](#) (December
285 2001) defines conformance for ebXML Registry Client implementations and
286 ebXML Registry implementations.

287

288 Example 2: The [W3C XSLT Recommendation](#) defines conformance for XSLT
289 processors. It does not define conformance for editors or generators that create
290 stylesheets.

291

292 **8.1.1. Modularity**

293 A class of product may consist of several integrated components rather than a single
294 piece of software (e.g., browser). Conformance may be defined in terms of the integrated
295 components (system) and/or for each component. Any restrictions or constraints on the
296 number or types of components that make up the “subject of a conformance claim”
297 SHALL be specified.

298

299 For systems that are comprised of several components, it may be sufficient to state that
300 conformance to the system is equivalent to conformance to all the required components
301 considered individually, and the system satisfies at least the minimum conformance
302 requirements for each of those components.

303

304 For example, the conformance clause in the ebXML Technical Architecture
305 states, “ebXML conformance is defined as conformance to an ebXML system that
306 is comprised of all the architectural components of the ebXML infrastructure and
307 satisfies at least the minimum conformance requirements for each of the ebXML
308 technical specifications.”

309

310 **8.1.2. Specifying conformance claims**

311 A specification may differentiate conformance claims by designating different degrees of
312 conformance in order to apply and group requirements according to profiles or levels or
313 to indicate the permissibility of extensions. When a conformance claim is linked to
314 functionality, impact and/or incremental degrees of implementation, the term
315 *conformance level* is often used to indicate the varying degrees of conformance. When a
316 conformance claim is linked to extensions, the term *strict conformance* is often used.
317 Strict conformance is defined as conformance of an implementation that employs only
318 the requirements of the specification and no more.

319

320 The conformance clause SHALL identify and define all designations of conformance.

321

322 For example, the W3C Web Accessibility Guideline designates three
323 conformance levels (Level A, Double-A and Triple A) based on the checkpoint
324 priority levels satisfied. Conformance Level A: all Priority 1 checkpoints are
325 satisfied; Conformance Level Double-A: all Priority 1 and 2 checkpoints are
326 satisfied; and Conformance Level Triple-A: all Priority 1, 2, and 3 checkpoints
327 are satisfied.

328

329 The specification MAY provide the specific wording of the claim (Appendix A provides
330 sample conformance claims). It MAY also require specific information to be contained in
331 the claim, such as name/date/version of the specification, test suite, and tested product.

332

333 The specification SHALL impose no restrictions about who can make a conformance
334 claim (e.g., vendor, user, third party) or where the claims may be published. It MAY
335 provide additional information regarding the responsibility of claimants.

336

337 **8.2. Profiles and Levels**

338 Often implementations do not use all the features within a specification. In order to
339 accommodate these implementations it may be desirable to divide a specification into sets
340 of functions. Implementers would still be conforming if they implemented one or more
341 of these sets rather than the entire standard. These sets are commonly implemented as
342 profiles or levels.

343

344 *Profiles* are used as a method for defining subsets of a specification by identifying the
345 functionality, parameters, options, and/or implementation requirements necessary to
346 satisfy the requirements of a particular community of users. Specifications that explicitly
347 recognize profiles should provide rules for profile creation, maintenance, registration and
348 applicability. Appendix B provides additional information on profiles.

349
350 *Levels* are used to indicate nested subsets of functionality, ranging from minimal or core
351 functionality to full or complete functionality. Typically, level 1 is the minimal or core
352 of the specification that must be implemented by all products. Level 2 includes all of
353 level 1 and also additional functionality. This nesting continues until level n, which
354 consists of the entire specification.

355
356 It is possible for a specification to have both profiles and levels. If profiles and/or levels
357 are defined, the conformance clause specifies which (if any) of these profiles and/or
358 levels is mandatory. Additionally, any conditions associated with a particular profile,
359 level or combination of these needs to be specified.

360
361 If profiles and/or levels exist, the specification SHALL indicate the conditions for
362 claiming conformance to a specific profile and/or level. In particular, consider whether
363 or not a claim of conformance to a particular profile/level can include functionality or
364 features of a higher profile/level. Typically, implementations that purport to conform to a
365 specific level of a specification MAY include functionality defined within one of the
366 higher levels.

367
368 Caution should be exercised in creating of profiles and/or levels. Experience has shown
369 that having too many profiles and/or levels can inhibit interoperability as well as add
370 confusion to the marketplace.

371 **8.3. Extensions**

372 An extension to a specification is a mechanism to incorporate functionality beyond what
373 is defined in the specification. Allowing extensions affects how conformance is defined
374 as well as what conformance claims may be made. Care should be exercised in
375 determining the extent to which extensions are allowed or not allowed. Since extensions
376 can seriously compromise interoperability, specification writers should carefully consider
377 whether extensions should be allowed. Appendix C provides additional information
378 about extensions.

379 **8.3.1. Disallow Extensions**

380 If a specification disallows extensions, then the conformance clause SHALL specify that
381 extensions are not allowed and that implementations of the specification SHALL
382 precisely implement the complete specification. This is strict conformance. Strict
383 conformance is often imposed on applications or content of a specification (e.g., a
384 software program or XML document instance). Strict conformance may also be imposed
385 on implementations (e.g., as in Ada). Note, that this prohibition of extensions could be
386 applied to a specific profile or level rather than to the entire specification.

387
388

389 **8.3.2. Allow Extensions**

390 If specification allows extensions, then the conformance clause SHALL state the
391 conditions under which extensions are allowed, the applicability of the extensions, their
392 affect on conformance claims, and any limitations or restrictions on the use of the
393 extension.

394
395 The conformance clause SHALL include the following statements or their equivalent:

- 396 • Each implementation SHALL fully support all required functionality of the
397 specification exactly as specified.
- 398 • The use of extensions SHALL NOT contradict nor cause the non-conformance of
399 functionality defined in the specification.

400

401 Depending on the specification, specification developers MAY want to include the
402 following additional requirements:

- 403 • Extensions SHALL follow the principles and guidelines of the specification they
404 extend, that is, the specifications MUST be extended in a standard manner (see section
405 below).
- 406 • For implementations and/or applications that contain extensions, extensions SHALL
407 be clearly described in supporting documentation and the extensions SHALL be marked
408 as such within the implementation/application.
- 409 • For implementations that contain extensions, there SHALL be a mode under which
410 the implementation can be directed to produce only conformant files (documents) or to
411 operate in a strictly conformant manner.

412 **8.4. Discretionary Items**

413 Specifications SHALL define or allow discretionary behavior by explicitly stating those
414 cases and conditions where discretion is allowed and/or expected. Discretionary items
415 may be warranted because of environmental conditions (e.g., hardware limitations or
416 software configuration, external systems), locality (e.g., time zone or language), optional
417 choices providing flexibility of implementation, dependence on other specifications, etc.
418 Two types of discretionary items are discussed below.

419

420 **8.4.1. Implementation dependent values**

421 In some instances, it may not be possible to define the behavior or values of a function.
422 *Implementation dependent* means that an implementation may determine the effect
423 (rather than having the effect mandated by the specification). However, the specification
424 SHALL make it clear that such effects shall be consistent within a single implementation
425 (e.g., a browser's rendering of a XSL-FO shall be the same for every invocation
426 regardless of the document instance).

427

428 Details in a specification MAY deliberately be omitted (i.e., not specified), so as to
429 provide freedom to adapt implementations to different environments and different
430 requirements. In general this is not a recommended practice. Caution should be exercised
431 if details are omitted and used only in a limited number of instances.

432

433 Specifications SHALL indicate implementation dependencies and where applicable,
434 address allowable differences between implementations, including,

- 435 • implementation dependent ranges, data, minimum or maximum values, etc.,
- 436 • Values that may be different for different conforming implementations of the
437 standard,
- 438 • environmental resources (e.g., memory or disk limitations),
- 439 • environmental values (i.e., language and local settings).

440

441 For example, a specification for a process that generates a numbered list with
442 roman numerals may specify a minimum range that shall be supported, but allow
443 implementations to generate larger numbers.

444

445 **8.4.2. Alternate approaches**

446 Specifications may describe several different ways to accomplish its operation (e.g., a
447 choice of file formats, protocols, or encodings). In such a case, the conformance clause
448 SHALL specify the conditions under which an implementation is considered to be
449 conformant. Some possible ways to define conformance include mandating that an
450 implementation shall:

- 451 1. implement only one approach,
- 452 2. implement every approach,
- 453 3. be allowed to implement none of the approaches.

454

455 Note: if the specification doesn't describe the different approaches, this becomes an
456 implementation detail irrelevant to conformance.

457

458 For example, the [W3C XSLT Recommendation](#) limits the set of situations under
459 which an attribute node is allowed to be produced on the output tree. If an
460 attempt is made to produce an attribute node in any other situation, the
461 Recommendation allows only two course of action: raise an error or ignore the
462 attribute. No other behavior is considered conformant, but either of the
463 enumerated behaviors is equally conformant.

464 **8.5. Deprecation**

465 After the initial publication of a specification, specification developers may be
466 considering the deprecation of a feature (i.e., element or attribute) defined in the
467 specification. A *deprecated feature* is a feature whose use is discouraged because it has
468 been outdated by newer constructs or is no longer viable. Deprecated features may
469 become obsolete and no longer defined in future versions of the specification.
470 Deprecated features warn implementers that the feature was a bad idea and it may be
471 withdrawn in the future.

472

473 Specification developers need to consider the affect of deprecation on all the classes of
474 products that implement the specification (e.g., authoring tools, user agents) as well as
475 the conformance consequences on each class of product For the purpose of backward
476 compatibility, it may be necessary to specify different requirements for the support of



477 deprecated features for each class of product. For example: authoring tools shall not use
478 the deprecated feature, whereas user agents shall support the deprecated feature.

479
480 If a specification contains deprecated features, the specification SHALL identify and
481 clearly mark each deprecated feature. Additionally, the specification SHALL specify, for
482 each class of products, the level of support required for the deprecated feature and the
483 conformance consequences of the deprecation. The specification MAY include a note
484 describing the rationale for the deprecation. The specification MAY include examples
485 that illustrate how to avoid using deprecated features.

486
487 Example 1: [SMIL 2.0](#) addresses deprecated features in the SMIL profiles (SMIL
488 Language, XHTML+SMIL, etc.). SMIL 2.0 Language user agents must support
489 all deprecated features. This ensures backward compatibility with SMIL 1.0
490 content. Since there are no user agents that support XHTML+SMIL 1.0 and very
491 little content, there is no requirement for backward compatibility to this profile.
492 Thus, there is no requirement to support deprecated features.

493
494 Example 2: [MathML 2.0](#) defines what it means for a feature to be deprecated as
495 follows: (a) In order to be MathML-output-compliant, authoring tools may not
496 generate MathML markup containing deprecated features. (b) In order to be
497 MathML-input-compliant, rendering/reading tools must support deprecated
498 features if they are to be MathML 1.x compliant. They do not have to support
499 deprecated features to be considered MathML 2.0 compliant. However, all tools
500 are encouraged to support the old forms as much as possible. (c) In order to be
501 MathML-roundtrip compliant, a processor need only preserve MathML
502 equivalence on expressions containing no deprecated features.

503 **8.6. Internationalization – Languages and Character sets**

504 Every specification SHALL identify, either by default or explicitly, a single natural
505 language or a more formal specification language (e.g., IDL, UML) edition as the
506 normative version.

507
508 Every specification SHALL specify whether it permits multiple or alternative natural
509 languages, language bindings and/or character encodings. If it permits these, it SHALL
510 specify the languages and encodings that SHALL be supported by conforming
511 implementations. Additionally, the error conditions and/or behavior to handle situations
512 in which unsupported languages or encodings are encountered SHALL be defined.

513
514 When specifying characters, the Unicode Standard [ISO 10646] SHALL be used.

515
516

517 **9. Additional Issues to Address**

518 **9.1. Implementation Conformance Statement (questionnaire)**

519 A specification MAY include an Implementation Conformance Statement (ICS) or
520 questionnaire and require its completion as part of a conformance claim. An ICS is

521 useful in clarifying and declaring optional functionality and discretionary behavior and
522 values. The results of the ICS can be used to identify the subset of test cases from a
523 conformance test suite that are applicable to the implementation to be tested. This will
524 allow the implementation to be tested for conformance against only the relevant
525 requirements. The ICS is also helpful in describing the expected interoperability to be
526 achieved with other implementations or applications of the specification.

527

528 If an ICS is included as part of the specification, it SHALL be explicitly identified as
529 either a normative or informative part of the specification.

530

531 For example, a specification that allows the implementation to perform locale-
532 aware processing for locales of the implementor's choosing, could use an ICS to
533 obtain a list of the implemented locales from the implementor. Similarly, a
534 specification that allows an implementation to choose from an enumerated list of
535 behaviors could use an ICS to find out which behavior is implemented.

536

537 **9.2. Test Assertions**

538 A specification MAY include test assertions as part of the specification. A *test assertion*
539 is a statement of behavior, action or condition that can be measured or tested. It is
540 derived from the specification's requirements and bridges the gap between the narrative
541 of the specification and the test cases. Each test assertion is an independent, complete,
542 testable statement for requirements in the specification. Each test assertion results in one
543 or more test cases.

544

545 Including test assertions as part of the specification facilitates and promotes the
546 development of conformance test suites and tools. Specific benefits include:

- 547 • helping to uncover inconsistencies, ambiguities, gaps, and non-testable statements in
548 the specification by developing test assertions in parallel with the specification,
- 549 • ensuring consistency between the specification and assertions,
- 550 • allowing test assertions to be reviewed and accepted by the specification developers
551 and the public,
- 552 • providing a common set of assertions (and thus interpretation of the requirements)
553 from which test developers can develop conformance tests,
- 554 • encouraging the early development of conformance tests that can be used by
555 implementers during the development of their implementation,
- 556 • achieving comparability between the results of corresponding tests developed by
557 different organizations,
- 558 • achieving confidence in the resulting tests as a measure of conformance.

559

560 Examples of specifications that included test assertions as part of their specification
561 include several IEEE and ISO standards, most notably IEEE POSIX and ISO 10303
562 (STEP).

563

564 **9.3. Specify a testing methodology or program**

565 A specification MAY provide a test framework, methodology and/or procedures for
566 testing to the specification. This type of information ensures consistency between testing
567 programs and organizations, and provides confidence in those testing programs. If any of
568 this information is provided, it SHALL be explicitly identified as either normative or
569 informative guidelines.

570
571 The test methodology MAY describe the conformance testing approach – the use of
572 methods involving rigorous proofs of correctness in which conformance can be
573 conclusively and exhaustively demonstrated (e.g., the syntactic validators for HTML,
574 CSS, accessibility of content) or the use of methods involving falsification testing.

575
576 The test method MAY specify the use of XML equivalence mechanisms such as XML
577 Information Sets or Canonical form when comparing test results to expected results.

578
579 The test methodology MAY describe the different types of conformance tests and tools
580 that need to be developed, the type of test materials that need to accompany the tests, and
581 the type of information contained in a test report.

582
583 The procedures for testing MAY describe the organizational structure, activities and
584 responsibilities for external organizations that establish and operate a testing service for
585 the specification.

586
587 The procedures for testing MAY prescribe how testing is conducted (e.g., self-declaration
588 or third party testing laboratories). It MAY also provide a step-by-step guide for using
589 the tests or tools correctly so that the results are repeatable and reproducible.

590
591 This type of information is provided as normative sections in several standards, e.g., ISO
592 10303 (STEP) and ISO 15046 (Geographic Information), and as part of several consortia
593 specifications, e.g., RosettaNet.

594

595 **10. Conformance Claim**

596

597 This section is the conformance claim for how this document conforms to itself. This
598 document conforms to the OASIS Conformance Requirements for Specifications version
599 0.5, 1 March 2002. (*ed note: update this as appropriate*).

600

601 The conformance issues in section 8 apply to this document as follows:

602

- 603 1. This document is applicable to all specifications. In order to claim conformance
604 to this document, all the requirements in section 3.1 SHALL be met.
- 605 2. This document SHALL be implemented in its entirety. It defines no profiles and
606 no levels.
- 607 3. This document allows extensions. Extensions included in a conforming
608 specification would address additional conformance issues and/or contain

609 additional statements contributing to a clearer, more measurable, less ambiguous,
610 specification.
611 4. This document contains no discretionary items.
612 5. This document's normative language is English. Translation into other languages
613 is permitted.
614
615

615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651

Appendix A: Sample Conformance Claims

Informative

In general, a conformance claim should contain the name and version of the tested implementation, the name and version of the specification, name and version of the test suite, date testing was completed, conformance level (or profile) satisfied, and the results of the testing. For example:

Name of Implementation and version has been tested for *Level L* conformance to *Name of Specification* and version using the *Name of Test suite, ver X.X* on YY-MM-DD and no nonconformities were found.

This *Name of Implementation* (fully specified) has been tested for conformance to *Name of Specification*, in accordance with the *XXX Validation Procedures* using the *Test Suite* and testing environment listed below:

- Name of Certificate Holder:
- Implementation Identification:
- Testing Environment (hardware/software):
- Test Suite name and version
- Level of Conformance:
- Nonconformities:
- Test Report: provide a URI

Specific Examples

The Web Content Accessibility Guideline requires a claim to contain the title of the guidelines document, its URI, the conformance level satisfied, and the scope covered by the claim (e.g., page, site), for example:

This page conforms to W3C's "Web Content Accessibility Guidelines 1.0", available at <http://www.w3.org/TR/1999/WAI-WEBCONTENT-19990505>, level Double-A

Appendix B: Profiles

Informative

The following is extracted from ISO 8632 Computer Graphics Metafile Standard

A profile of a specification defines the options, elements, and parameters necessary to accomplish a particular function and maximize the probability of interchange between systems implementing the profile. Profiles are defined to meet the requirements of application constituencies who are asked to adhere to the same subset of the specification.

661 A profile may be a subset of a single specification or may be part of the set of interrelated
662 standards and profiles assembled for the purpose of accomplishing a larger functional
663 purpose. A profile shall not specify any requirement that would contradict or cause non-
664 conformance to its specification.

665
666 A profile may:

- 667 • give the meaning of implementation dependent semantics of some elements,
- 668 • enforce common resolution of ambiguous semantics,
- 669 • ensure that identical use of identical elements and parameter values have the same
670 meaning,
- 671 • specify subsets or groupings of publicly defined extensions,
- 672 • prohibit undefined or ill-defined elements or parameter values.

673
674 Profiles provide a means to:

- 675 • improve interoperability between implementations by inhibiting the proliferation of
676 private subsets of a specification,
- 677 • provide a foundation for testing and promote uniformity of conformance tests,
- 678 • enhance the availability of consistent implementations of a profile.

679
680
681

682 **Appendix C: Extensions**

683 Informative

684

685 An extension may be *private* (often vendor specific) or may be *public* (a full description
686 of the extension is public). Private extensions are usually truly private, i.e., valid for a
687 specific implementation or are only known by prior agreement between implementations.
688 Public extensions are extensions in which the syntax, semantics, identifiers, etc are
689 defined and published allowing anyone to implement the extended functionality.

690

691 **C.1 Mechanism to allow extensions**

692 One mechanism to allow extensions within a specification is to provide a standard way of
693 defining the extension or a “standard way of being non-standard”. This helps to ensure
694 predictable handling of extensions, that is, its recognition as such and the appropriate
695 action (i.e., to ignore or to implement). The nature of the extension may dictate the
696 method for defining the extension. It may be possible to define a generic function or
697 mechanism that indicates external (from the specification) functionality. This external
698 function/mechanism may take the form of an escape or control character or be an
699 identifier, which whenever invoked indicates an extension follows. Another method,
700 especially when extending a list of numeric parameters is to use a scheme where positive
701 values represent standardized values and negative values are reserved for private use.

702

703 Another mechanism that minimizes interoperability problems when extensions are
704 allowed is to have a register for extensions. This document, distinct from the official
705 specification, contains a list of recognized extensions to the standard. See section below.

706 In a language that supports qualified names, like XML with its namespaces, extensions
707 may be required to use names from namespaces other than the one used in the
708 specification. The specification can then define a mechanism by which certain
709 namespaces are denoted to contain extensions rather than any other type of syntactic
710 element.

711

712 For example, the W3C XSLT Recommendation specifies that the outer element of
713 a stylesheet may contain an attribute extension-element-prefixes =
714 “prefix1prefix2prefix3...” and that the given prefixes are mapped to namespaces.
715 All elements in those namespaces are designated as extension elements, as
716 opposed to other uses of elements with qualified names that are described
717 elsewhere in the Recommendation. The namespace for XSLT stylesheets shall not
718 be on the list, and an implementor is also prohibited from adding any elements to
719 the XSLT namespace. (This designation applies locally within the stylesheet and
720 is a “totally private extension”.)

721

722 **C.2 Registration of implementer extensions or implementation defined values**

723 Registration is a procedure that allows extensions to be acknowledged and made
724 available to the public. Registration provides for a degree of rigor and technical review
725 for any proposed extension. Typically, the committee developing the specification is
726 responsible for processing the registration of an extension, thus ensuring adequate quality
727 of a proposed extension and a technical description sufficient to be uniformly
728 implementable. Often, registered extensions may migrate into a later version of the
729 specification

730

731 **C.3 Caution: proceed with care when using extensions**

732 Specifications may allow extensions for various reasons. Extensions allow implementers
733 to include features that are in demand by their customers. Also, extensions, often times,
734 define new features that may migrate into future versions of the specifications. However,
735 the use of extensions can have a severe negative impact on interoperability. Some
736 methods for enabling extensions have less impact on interoperability than other methods.
737 For example, a specification that allows private extensions (e.g., proprietary) is more
738 likely to impede interoperability than a specification that requires extensions to be
739 registered. The table below illustrates various methods for implementing extensions and
740 their impact on interoperability.

741

742

743

744

745

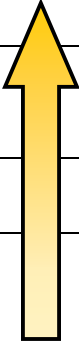
746

747

748

749

750

Impact on Interoperability	Method of Implementing Extension	Examples of specifications containing extensions
Greatest Negative Impact	Totally private extensions	Unknown function references in XSLT
	Totally private extensions, but contained within a standard template	ISO 8632: CGM's Escape or GDP function
	Private, but with ability to inquire	???
	Registered extension	ISO Register of International Character Sets (in accordance with ISO 2375) ISO 9973: Procedures of Registration of Graphical Items.
Least Impact		

751 **Table 1: Extensions and their impact on interoperability**

752