Conformance Requirements for Specifications
Version 0.5
March 1, 2002

Editors:
Lynne Rosenthal (lynne.rosenthal@nist.gov)
Mark Skall (mark.skall@nist.gov)

Contributors:
Lofton Henderson (lofton@rockynet.com)
David Marston (david marston@lotus.com)

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

Status of this Document
Working Draft – v0.5

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

Reference Documents
ISO Guide 2: Standardization and related activities – General vocabulary.
ISO/IEC Directives Part 3: Rules for the structure and drafting of International Standards
ebXML Technical Architecture Specification, Conformance Clause
OASIS/ebXML Registry Services Specification
W3C WAI Guidelines
W3C XSLT/Xpath Recommendation
UNICODE
SAML specifications

TABLE of CONTENTS

1. Introduction

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

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

2. Scope and Audience

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

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

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

3. Conformance

A specification that conforms to this document SHALL:

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

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

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

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

4. Normative references

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

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

5. Informative references

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

Glossary of Conformance Terminology, http://www.oasis-open.org/committees/ioc/glossary.htm.

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

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

6. Terms and definitions

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

Accreditation – procedure by which an authoritative body gives formal recognition that a body or person is competent to carry out specific tasks.
Certification – the acknowledgement that a validation has been completed and the criteria established by the certifying organization has been met.
Conformance – the fulfillment of a product, process, or service of specified requirements.
Conformance Testing – a method of verifying implementations of a specification to determine whether or not deviations from the specification exist.
Implementation – the realization of a specification – it can be a software product, system, program, protocol, application, or document instance.
Strict Conformance – conformance of an implementation that employs only the requirements and/or functionality defined in the specification and no more (i.e., no extensions to the specification are implemented).
Validation – the process of testing software for conformance to a specific specification.

7. Conformance Clause

Every specification SHALL contain a conformance clause.

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

7.1. Rationale for a conformance clause

A conformance clause:

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

7.2. Conformance keywords

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

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

ISO Keywords:

SHALL – to indicate requirements strictly to be followed in order to conform to the standard and in which no deviation is permitted. Equivalent expressions include: is to, is required to, has to, it is necessary. Do not use MUST as an alternative for shall.
SHALL NOT - converse of SHALL.
SHOULD – to indicate that among several possibilities one is recommended as particularly suitable, without mentioning or excluding others.
SHOULD NOT – converse of SHOULD.
MAY – to indicate a course of action permissible within the limits of the standard. Equivalent expressions include: is permitted, is allowed.
NEED NOT – to indicate a course of action is not required.
CAN – statement of possibility and capability, whether material, physical or causal. Equivalent expressions include: be able to, it is possible to.
CANNOT – converse of CAN.

IETF Keywords (RCF2119)

MUST - the requirement is an absolute requirement of the specification.
MUST NOT – the requirement is an absolute prohibition of the specification.
REQUIRED – see MUST.
SHALL – see MUST.
SHALL NOT – see MUST NOT.
SHOULD – there may exist valid reasons in particular circumstances to ignore a particular item, but the full implications must be understood and carefully weighed before choosing a difference course.
SHOULD NOT – there may exist valid reasons in particular circumstances when the particular behavior is acceptable or even useful, but the full implications should be understood and the case carefully weighed before implementing any behavior described with this label.
REOMMENDED – see SHOULD.
MAY - the item is truly optional. One vendor may choose to include the item because a particular marketplace requires it or because the vendor feels that it enhances the product while another vendor may omit the same item. An implementation that does not include a particular option MUST be prepared to interoperate with another implementation that does include the option, though perhaps with reduced functionality. In the same vein an implementation, which does include a particular option MUST be prepared to interoperate with another implementation that does not include the option (except, of course, for the feature the option provides.)

Additionally keywords include:

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

7.3. General principles

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

- normative and informative sections SAHLL be evident and if necessary, labeled accordingly,
- uniformity of structure, of style, and terminology SHALL be maintained within the specification,
- identical wording SHALL be used to express identical provisions and analogous wording SHALL be used to express analogous provisions.

8. What to Address in a Conformance Clause

8.1. What needs to conform

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

Example 1: The OASIS/ebXML Registry Services Specification (December 2001) defines conformance for ebXML Registry Client implementations and ebXML Registry implementations.

Example 2: The W3C XSLT Recommendation defines conformance for XSLT processors. It does not define conformance for editors or generators that create stylesheets.

8.1.1. Modularity

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

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

> For example, the conformance clause in the ebXML Technical Architecture states, "ebXML conformance is defined as conformance to an ebXML system that is comprised of all the architectural components of the ebXML infrastructure and satisfies at least the minimum conformance requirements for each of the ebXML technical specifications."

8.1.2. Specifying conformance claims

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

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

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

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

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

8.2. Profiles and Levels

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

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

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

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

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

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

8.3. Extensions

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

8.3.1. Disallow Extensions

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

8.3.2. Allow Extensions

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

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

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

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

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

8.4. Discretionary Items

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

8.4.1. Implementation dependent values

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

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

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

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

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

8.4.2. Alternate approaches

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

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

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

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

8.5. Deprecation

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

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

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

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

> Example 1: SMIL 2.0 addresses deprecated features in the SMIL profiles (SMIL Language, XHTML+SMIL, etc.). SMIL 2.0 Language user agents must support all deprecated features. This ensures backward compatibility with SMIL 1.0 content. Since there are no user agents that support XHTML+SMIL 1.0 and very little content, there is no requirement for backward compatibility to this profile. Thus, there is no requirement to support deprecated features.
>
> Example 2: MathML 2.0 defines what it means for a feature to be deprecated as follows: (a) In order to be MathML-output-compliant, authoring tools may not generate MathML markup containing deprecated features. (b) In order to be MathML-input-compliant, rendering/reading tools must support deprecated features if they are to be MathML 1.x compliant. They do not have to support deprecated features to be considered MathML 2.0 compliant. However, all tools are encouraged to support the old forms as much as possible. (c) In order to be MathML-roundtrip compliant, a processor need only preserve MathML equivalence on expressions containing no deprecated features.

8.6. Internationalization – Languages and Character sets

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

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

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

9. Additional Issues to Address

9.1. Implementation Conformance Statement (questionnaire)

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

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

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

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

9.2. Test Assertions

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

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

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

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

9.3. Specify a testing methodology or program

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

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

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

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

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

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

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

10. Conformance Claim

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

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

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

additional statements contributing to a clearer, more measurable, less ambiguous, specification.

4. This document contains no discretionary items.
5. This document's normative language is English. Translation into other languages is permitted.

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.

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

A profile may:

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

Profiles provide a means to:

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

Appendix C: Extensions

Informative

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

C.1 Mechanism to allow extensions

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

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

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

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

C.2 Registration of implementer extensions or implementation defined values

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

C.3 Caution: proceed with care when using extensions

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

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	???
Least Impact	Registered extension	ISO Register of International Character Sets (in accordance with ISO 2375) ISO 9973: Procedures of Registration of Graphical Items.

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

752