Workgroup: Network Working Group
Internet-Draft:
draft-bormann-rswg-terminology-00
Published: 27 July 2023
Intended Status: Informational
Expires: 28 January 2024
Authors: C. Bormann
Universität Bremen TZI

Terminology for RFCXML Evolution

Abstract

The canonical format for RFCs is called RFCXML, with the currently effective details originally documented in the RFC 799x series. This format has experienced some uncontrolled evolution since, partially caused by an unwillingness to recognize the need for overt, deliberate evolution.

Controlled RFCXML evolution is going to be complex. Its discussion will need agreed terminology, without which it will devolve into a Tower of Babel.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-bormann-rswg-terminology/.

Discussion of this document takes place on the rswg Working Group mailing list (mailto:rswg@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/rswg/. Subscribe at https://www.ietf.org/mailman/listinfo/rswg/.

Source for this draft and an issue tracker can be found at https://github.com/cabo/rswg-terminology.

Status of This Memo

at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 28 January 2024.

Table of Contents

1. Introduction

The canonical format for RFCs is called RFCXML, with the currently effective details originally documented in the RFC 799x series. This format has experienced some uncontrolled evolution since, partially caused by an unwillingness to recognize the need for overt, deliberate evolution.

Controlled RFCXML evolution is going to be complex. Its discussion will need agreed terminology, without which it will devolve into a Tower of Babel.

1.1. Conventions and Definitions

Although this document is not an IETF Standards Track publication, it adopts the conventions for normative language to provide clarity of instructions to the implementer. The key words "**MUST**", "**MUST**

NOT", "**REQUIRED**", "**SHALL**", "**SHALL NOT**", "**SHOULD**", "**SHOULD NOT**", "**RECOMMENDED**", "**NOT RECOMMENDED**", "**MAY**", and "**OPTIONAL**" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

2. Terminology

Ultimately, this document should turn into a definitions section of some other document. For now, we will use a mix of prose and definition styles.

2.1. Format

XML does not define the meaning of its instances. Saying "this document is in XML" doesn't tell you much more about its semantics than "this document is in ASCII".

When we talk about the specific semantics instilled into an XML document by the RFCXML format, we will therefore always use the term RFCXML. This term can be split into several aspects:

* **syntactic** aspects. As XML is (mostly) a tree, this is often reduced to a (tree) **grammar** of XML elements and XML attributes. However, there are other syntactical aspects, such as for the text in elements and attribute values (**lexical** aspects): the meaning of specific characters (e.g., format effectors; hyphenation semantics) and even whether some text is allowed in certain elements or attribute values.

* **semantic** aspects. The elements and attributes carry specific semantics. These semantics are perceived by document users through **renderings**. Semantic markup is about keeping the semantics mostly at a domain level, with an ability to infer the right kind of **layout** (in a wide sense, e.g., including font choice) in a rendering process. However, there are also semantics that are defined at a rendering level, e.g., those of and . (Officially, these are also semantic markup, but as soon as a "Conventions" section says "Newly defined terms are shown in italics", that is no longer true.)

Semantic aspects that are not rendered are **hidden** semantics. E.g., the <keywords> element is entirely not rendered in today's renderings; it is intended for processes outside of rendering/human consumption (e.g., search). The <sourcecode name= attribute is rendered only in certain cases, but can be used by sourcecode extraction processes (e.g., for CI or for re-use of pseudo-code in other contexts) in other cases, too.

RFCXML currently has three rendering **targets** offered by the RFC editor: **TXT**, **HTML**, **PDF**. HTML and PDF are **typographic** renderings, TXT is a **typewriter** rendering. IETF datatracker also has a forth and fifth rendering target, which uses HTML or PDF, but tries to emulate TXT rendering while doing so.

Some semantics is hidden in some of these renderings, but not others. E.g., the <tt> element serves to identify text as different from normal running text, semantically similar to the way <sourcecode> and <artwork> do, but syntactically more like or . Since xml2rfc 3.10.0, the semantics of <tt> are **suppressed** in TXT renderings, which leads to problems (not just the middle-of-the-river semantic change, but also for new documents: the inability to express certain semantics in a way that they are recognizable in TXT but not distracting in the typographic renderings). A recent poll whether should also be suppressed in TXT ended with a negative result.

Note that suppression of certain semantics in certain rendering targets is fine if the semantics is **ancillary**. Different documents differ in their usage of certain markup semantics, and even different authors of the same document may disagree whether some usage is ancillary or **essential** (i.e., of semantic intent, conveying meaning): From an author's view, usage of specific markup can be for aesthetic purposes, it can increase ease of use of the document, it can help prevent a misunderstanding (which can have very different levels of likelihood to occur), or it can be essential.

2.2. Instances

RFCs are **instances** of RFCXML, specifically the **publishing** subset of RFCXML. As of today, these instances are **immutable**. Format evolution may call for a way to evolve the instances along with an evolved format specification.

Most RFCs are the result of a **consensus** process, either full IETF consensus or maybe just the review of a smaller group whether the document should be published (IAB, IRTF RG, ISE review).

This consensus is almost exclusively formed by review processes that involve reviewing renderings, only very rarely by looking at the RFCXML instance itself. These review processes are often extremely expensive, as they involve contributions from sought-after experts in the field. Their output constitutes much of the value of the RFC series.

During the review processes, the document instance is not an RFC. Specifically, the **authoring** subset of RFCXML is used, which has slightly different characteristics from the publishing subset. As

mentioned, we sometimes also use different renderers during the authoring/reviewing process (e.g., datatracker's distinct HTML/PDF renderings), reducing the congruence of the reviewed document with what its users will see.

2.3. Evolution

The definition of RFCXML will evolve, by adding functionality, or by taking elements and attributes out of service (sometimes called **deprecating**, but see below) that have been obsoleted in some way.

This is relatively straightforward for new documents.

Documents that have been in the authoring process and have already received expensive review generally need a **transition** strategy, such as translation from the format defined by an older RFCXML specification to a newer one. This transition often needs to be synchronized with tool development more than with consensus processes on the format itself, which can give tools a de-facto normative role.

Documents that already have been published cannot benefit from format evolution as long as their XML instances are immutable. This can be accommodated by keeping RFCXML able to process published documents — just those, not the entirety of potential instances of a previous RFCXML specification. This support would be tagged as for backwards compatibility only. (Backwards compatibility for documents in authoring/reviewing stage would reduce disruption.)

The corpus of published RFCXML-form documents is large enough that any translation processes to a new RFCXML specification need to be **automated**. Such automated processes can then also be made available for authoring/reviewing (xml2rfc's --v2v3 process is a nicely carried out example for that) or just focused on the finite set of documents published to a previous RFCXML specification.

A format change can affect the Syntax (grammar, other syntactic details not captured in the grammar), the Semantics, and/or the Rendering (possibly hiding some information in some renderings).

2.4. Types of Evolution

A term that has been used in a non-standard way in the creation of RFCXMLv3 is **deprecation**. In RFC799x, it means that the deprecated feature is no longer available for publishing. It is still available during authoring/reviewing, with an understanding that these

processes provide a way to do a reviewed manual translation or to at least review automated translation.

* Backwards compatibility (**BC**) means the ability of new systems to work with old data.

* Forward compatibility means the ability of old systems to work with new data. Forward compatibility is of little interest for the current discussion, as we generally view tooling as updated in sync with evolution processes. xml2rfc's input validation actively prevents forward compatibility, there is no "ignore-unknown" functionality even for semantics that could be ancillary.

Here, Backwards compatibility often can only be ascertained by manual review: It is not sufficient that the new system does not crash with the old data, the old data **MUST** be useful in the sense that it would survive the same review processes. (These are generally too expensive to be redone just for an RFCXML format change.)

A non-backwards-compatible (**NBC**) change to the RFCXML format can have **detectable** impact on a document, e.g., by now failing its validation. Or the impact can be **non-detectable**, i.e., requiring human review to detect, such as a semantic change that creates a different rendering that (potentially) has a different meaning.

A **semantic refinement** allows instances of the updated RFCXML specification to express more detailed information than previously possible. E.g., the element could be split into usages for term definitions, true emphasis, and other usages of italic type. It could carry hints as to how to emulate it in typewriter renderings.

A semantic refinement can be done in a roughly backwards-compatible way, by retaining the unrefined alternative (e.g.). Giving that alternative more limited semantics (e.g., by adding an attribute with a default value) is no longer truly backwards-compatible, as it is a (usually hidden!) semantic change. Retaining it without "deprecating" it will require some will-power --- but many documents may not have a need for the specific refinement (e.g., proposed in the example) and would be well-served by retaining the unrefined alternative.

2.5. Correcting Errors

If there is a need to translate RFC instances to new format specifications, they are no longer immutable (and/or their names need to be augmented by a revision indicator, possibly with a way added to obtain the most recent revision).

Opening up mutability provides an opportunity to correct errors in the originally published document, such as errata.

Such an **instance update** also can be used to replace now deprecated (in the English sense) markup by modern one.

An example for a detectable NBC change would be to only allow digits and single spaces between them in <rfc updates= attributes. Correcting this in the now failing instances would probably be done by manual intervention, as the number of instances is too small to justify automation.

3. Security Considerations

TBD

4. IANA Considerations

This document has no IANA actions.

5. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/rfc/rfc2119>.

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/rfc/rfc8174>.

Acknowledgments

TBD

Author's Address

Carsten Bormann
Universität Bremen TZI
Postfach 330440
D-28359 Bremen
Germany

Phone: +49-421-218-63921
Email: cabo@tzi.org