IETF ADMINISTRATION LLC

Request for Proposals

RFC Errata Merge Tool

Date of Issuance: April 8, 2019
Proposal Submission Deadline: April 25, 2019, 5:00 P.M. ET

IETF LLC Request for Proposals

RFC Errata Merge Tool

The IETF Administration LLC (IETF LLC) is soliciting proposals ("Proposals") to create a tool that takes, as its input, the source of an RFC and all of its errata, and creates an annotated version of the RFC that incorporates selected types of errata in a highly visible form.

Proposals from any commercial or non-commercial vendor are welcome.

Timeline
08 Apr: RFP Issued
15 Apr: Questions and Inquiries deadline
18 Apr: Answers to questions issued, RFP Addenda and Update issued
25 Apr: Proposals due
02 May: Selection made, negotiations begin
09 May: Contract execution
13 May: Work begins

This is the process for the Request for Bids:

1. The Statement of Work (SOW) is in attached.

2. Any questions about the Work must be submitted by 15 April 2019. A response to all parties shall be provided by 18 April 2019. The response will include the questions asked and the answers, but will not identify the company asking the question.

3. Bids are due by 25 April 2019. The bid must provide a not-to-exceed price, the expected start date, the expected completion date, any assumptions, and a description of any dependencies that might cause delays in the schedule.

4. The IETF LLC will discuss the Bids and may ask questions by email and/or conference call.

5. Once the answers are received a decision will be made to select the bidder to perform the work and a Work Order will be prepared for execution. We anticipate an award on or before 2 May 2019.

6. This is the Bid format:
    a. Executive Summary
    b. Project Approach & Plan
    c. Schedule - When the work will begin and end, as well as dependencies and other milestones.
    d. Test Plan
    e. Cost & Payment Schedule
    f. Warranty & Late Delivery Consequence
    g. Technical Support & Maintenance

h. Miscellaneous

7. Instructions for IETF Software Development Contractors will apply. See https://trac.tools.ietf.org/tools/ietfdb/wiki/ContractorInstructions?version=26

8. Additionally, bidders for the Datatracker Meeting Application Improvements also have the option to be considered as a qualified software vendor for Indefinite Delivery/Indefinite Quantity (IDIQ) Master Services Agreement (MSA) purposes. Please indicate your interest. IDIQ software vendors will be eligible for future software development projects to improve and expand to the existing IETF toolset, which has been substantially developed in Python using the Django framework.

Please reply with questions, if any, and a bid if you are interested in pursuing this opportunity to ietf-rfps@ietf.org.

Thanks in advance.

Portia Wenze-Danley

RFC Errata Merge Tool: Statement of Work

Table of Contents

Overview and Objectives

The RFC series is a suite of documents published by the RFC Editor for the benefit of the Internet community. It contains technical and organizational documents about the Internet, including the specifications and policy documents produced by four document streams within the RFC series: the Internet Engineering Task Force (IETF), the Internet Research Task Force (IRTF), the Internet Architecture Board (IAB), and Independent Submissions. Documents in the RFC series are identified by a number. Once a number is associated with an RFC, the document content referenced by that number is forever frozen and cannot be changed.

The RFC Editor maintains a database of errata that have been submitted by the consumers of RFCs. These errata, once reported, can be verified, rejected, or "held for document update." The visibility of errata to the consumers of RFCs has historically been poor or nonexistent. This project seeks to ameliorate this situation by creating a tool that takes, as its input, the source of an RFC and all of its errata, and creates an annotated version of the RFC that incorporates selected types of errata in a highly visible form.

This tool will initially be used to create pages served somewhere other than the RFC Editor's website (www.rfc-editor.org). After gaining experience and community feedback, the pages might be moved to that website. Thus, the development of the tools should assume that the way the results are deployed will change over time.

Note that, while the traditional singular form of "errata" is "erratum," IETF participants overwhelmingly use the term "errata" for both purposes. This document follows that convention.

List of Deliverables

This statement of work calls for the following deliverables. These deliverables are described in greater detail in the "Description of Tasks and Deliverables" section below.

1. **Errata Merge Tool** that ingests an RFC and corresponding errata, and generates an output document containing the RFC annotated with errata information.
2. **Support Tool** that fetches RFCs and errata, determines which RFCs require invocation of the Errata Merge Tool, invokes the Errata Merge Tool as needed, and uploads the annotated RFCs to one or more remote locations.
3. **Base CSS Stylesheet** that contains formatting common among all formats, as well as formatting suitable for printing.
4. **Color CSS Stylesheet** that formats errata annotation using color coding.
5. **Monochrome CSS Stylesheet** that formats errata annotation using font variations.
6. **JavaScript Support File** that contains any functions necessary for active content in the annotated RFC (e.g., tool-tips, overlays, and content that should be hideable and/or hidden). This file may not be necessary if all of the described features can be satisfactorily implemented using CSS alone. In no case will the HTML files generated by the Errata Merge Tool contain JavaScript files except by including this external file. It is permissible for this file to make use of additional libraries (e.g., React, Angular).

All code deliverables are required to include test-suite covering all functionality. Developers will follow the Contractor Instructions.

Background

In researching the feasibility of creating a tool to merge errata into existing RFCs, a simplistic prototype was developed. Based on experimentation with this prototype, it was determined that significant number of errata — approximately half, on a first-order estimation — can be mechanically placed in a document in such a way that the erroneous text is completely replaced by the corrected text. Over half of the remaining errata can be inserted into the proper section in the document, albeit not at a specific location within the section. The remaining 20% or so of errata cannot be localized to a specific section in the document.

Based on these promising initial results, we now seek the development of a proper tool to generate a "patched" version of RFCs that includes inline indications of errata that readers should be aware of.

Description of Tasks and Deliverables

Support Tool

The support tool will be responsible for providing updated patched versions of RFCs as new errata for that RFC become available. This tool should be designed to be efficient - only generating and uploading new files when they are needed. One reasonable approach would be to keep a local copy of the configured errata.json file and all generated HTML files, using simple HTTP mechanics (HEAD and Last-Modified) to ensure the errata.json file is fresh, and the modified time of the locally stored HTML files to determine which need to be created or regenerated. Another possible approach is using rsync with the deployed service to ensure the tool has the information necessary to determine what new files need to be generated. Any solution must allow the URL to find the current errata.json file and the site to which new generated files are to be uploaded to be configured.

Calculating which HTML files need to be (re)generated requires processing the contents of the errata.json file:

1. For each record in the "errata.json" file in the state "verified" or "held for document update," compare the record's "update_date" against the modification date of the corresponding annotated RFC file. If the errata "update_date" is later than the annotated RFC's modified date (or no annotated RFC file is present), make note of it as a document in need of updating.
2. After each errata record has been evaluated, invoke the Errata Merge Tool on all documents in need of updating.

The support tool must be suitable for running in an automated fashion (e.g., in a `cron` job).

Errata Merge Tool

The Errata Merge Tool will take a canonical RFC in text format and combine it with the information in the "errata.json" file to produce an HTML file with the relevant errata placed inline in the most useful position, as described below. This tool will accept a command line argument to determine which subset of errata types will be placed inline. The initial deployment

is expected to only include "approved" errata, but future deployments may focus on other errata types, particularly "hold for document update".

Errata Placement Criteria

The task of the Errata Merge Tool is summarized as reading a canonical RFC in text format, placing relevant errata at the proper location in the RFC, and writing out an annotated version of the RFC as an HTML file. There are four major steps in realizing this task: (1) finding sections within the document; (2) normalizing errata; (3) locating the most precise placement possible for the errata within the document; and (4) modifying the document.

Finding Document Sections

Most errata are reported against a specific section in a document. When this is true, the tool should examine only the text in that section when attempting to place an errata. Sections heading can generally be identified as lines starting with zero or more spaces, followed by a sequence of digits and dots, and typically ending in a dot. Appendices can be similarly identified as starting with zero or more spaces, followed by the word "Appendix", and then a sequence of digits, letters, and dots.

More recent RFCs are fairly consistent that section and appendix headers start in the first column of a line; however, as errata are reported against older RFCs as well, the tool needs to handle the quirks of older formats. Some interesting examples to test with include RFC 822 (which uses a constant indent for section headers) and RFC 1122 (which uses a variable indent for them).

Some documents omit section numbers altogether. For these documents, the tool should attempt to match the text against the entire document. See, for example, RFC 854: despite the lack of section numbers in the document, and despite the fact that Errata 571 indicates a section number, the tool should still be able to place it inline in the document.

The section-finding text in the Python module, 'rfc2html', available on https://pypi.org/, may prove useful in implementing this logic for the Errata Merge Tool.

Errata Normalization

Errata are entered by hand by their reporters, and frequently have subtle differences from the source material that would otherwise prevent the "original" text from matching the text in an RFC verbatim. Prior to attempting to match an errata against an RFC, its original and replacement text needs to be processed as follows:

- Any lines starting with "|" (the horizontal bar character) should have this bar removed. Several errata use this to designate which lines are being updated in the document. See Errata 2561 for an example.
- Any lines consisting exclusively of "^" should be removed. Several errata use this to designate specific columns in which changes are being made. See, for example, Errata 1701.
- Smartquotes (U+2018, U+2019, U+201C, and U+201D) should be converted to straight quotes (ASCII 34 and ASCII 39, as appropriate).

Locating Errata Placement

When placing errata, the goal of the tool is to find as precise a placement of the errata as possible. The following process will result in one of three outcomes for each errata, in decreasing order of preference: (1) being able to determine exact text to be corrected (an "inline" correction), (2) being able to determine a section to which an errata applies (a "section" correction), and (3) being unable to determine a specific location within the document (an "unplaceable" correction).

Ideally, the "original" text can be matched against a sequence of text in the correct section of the document. To do so, the "original text" is treated as a search string and compared against the indicated section in the RFC text. For the purposes of this search, the following rules are applied:

- Any sequence of one or more whitespace characters (ASCII values 9, 10, 12, 13, and 32) in the original text should be treated as matching any sequence of one or more whitespace characters in the RFC text.
- Any Unicode character sequences remaining in the original text should be treated as matching one to three arbitrary characters in the RFC text (e.g., if implemented using a regular expression engine, these would be treated as "`.{1,3}`").
- Since some errata include literal document headers and footers while other excise such text, an attempt should first be made to match the RFC in its original form (with headers and footers present); if this fails, then the headers and footers should be ignored when performing the match.

If applying these rules results in no match within the section, or if section boundaries cannot be located with the document, then the matching is instead attempted against the entire document instead of a single section.

If the result of the foregoing attempts result in exactly one match, then the errata can be applied "inline."

If the result is no match, or more than one match, then the tool will attempt to locate the proper section within the RFC, using the section number the errata was reported against. If this is possible, then the errata can be applied as the section level.

If the preceding attempts to find a match fail, then the errata will be applied as a combined header / footnote.

Updating the Document

Once the proper location for placing an errata has been determined, the output document is updated to reflect the contents of the errata.

For those "verified" errata that can be placed inline, assuming that rendering "verified" errata has been specified on the command-line, the original text is replaced by the corrected text. This text will be enclosed in a tag with an appropriate class that allows for errata-type-specific CSS styling (e.g., "verified-inline-styling").

Other errata types will be handled similarly, For example, those "held for document update" errata that can be placed inline, again assuming that rendering "hold for document update errata has been specified on the command-line, the original text remains in the RFC, but is enclosed in a tag that allows for errata-type-specific CSS styling (e.g., "held-inline-styling").

For any inline text matches, the text will be followed by text in a tag that indicates the errata number (with a link to the errata), the original text, and any notes associated with the errata. This will be tagged with a class that indicates it is supplementary information (e.g., "supplementary-styling").

Note that some documents have multiple errata that attempt to make conflicting changes to the same set of text. When this occurs, the tool should make an attempt to place the earliest errata using the preceding approaches, and should insert subsequent errata immediately below the placed errata in the same format as section-level errata.

For those errata that can be matched to a section, a will be placed at the beginning of the corresponding section. This will contain the errata number (with a link to the errata) along with the original text, the replacement text, and any notes associated with the errata. The class of the will distinguish the errata type (e.g., "verified-section-styling" and "held-section-styling").

The remaining errata will be placed at the end of the document, in individual elements. These will contain the errata number (with a link to the errata) along with the original text, the replacement text, and any notes associated with the errata. The class of the will distinguish the errata type (e.g., "verified-endnote-styling" and "held-endnote-styling"). Verified errata will also be added to a list that is prepended to the document as its own , with an indication similar to the following: "This document has been updated by the following errata, which cannot be shown in-line in the document. Please see the end of this file for additional details: Technical Errata 832, Technical Errata 1073". This div will have its own styling class (e.g., "verified-banner-styling").

When multiple errata are placed in the same location (e.g., at the beginning of a section or at the end of an RFC), they should be ordered by the date of their most recent change (oldest errata first).

Input Data

RFCs

The main document input format for this version of the tool is the plain-text version of an RFC, which includes headers, footers, and page breaks.

While the format of recent RFCs are fairly well standardized — with sections delineated by numeric section numbers starting in column 1 — there has historically been significant variation in the format of RFCs.

Errata

The preliminary format for the errata is single JSON array of objects, each of which contain the following fields:

- errata_id: Unique number that identifies this report
- doc-id: Indication of the RFC to which the report applies

- errata_status_code: Indication of the errata status. Valid values are "`Reported`", "`Verified`", "`Rejected`", and "`Held for Document Update`".
- errata_type_code: nature of the errata (either "`Technical`" or "`Editorial`")
- section: Section of the document to which the errata applies. Typically digits separated by dots, but may also take the two additional forms "Appendix *X*" (where *X* is some combination of digits and letters, separated by dots), and "GLOBAL".
- orig_text: Reporter-entered indication of the text in the RFC to be replaced.
- correct_text: Reporter-entered indication of the text that should replace the original text.
- notes: Reporter- and verifier-entered annotations explaining the rationale for the change and any additional information associated with the errata.
- submit_date: Date, in "YYYY-MM-DD" format, on which the errata was initially reported.
- submitter_name: Unverified submitter-entered indication of their own name.
- verifier_id: Unique (within the errata database), authenticated numeric identifier of the verifier (that is, the person who placed the errata in the "verified," "rejected," or "held for document update" state).
- verifier_name: Human-readable form of the individual identified by the verifier_id field.
- update_date: Date and time at which the errata record was most recently updated, in the format "YYYY-MM-DD HH:MM:SS".

Dates and times are given in US Pacific Time, using the time in effect that the record was created (i.e., using PDT when Daylight Savings Time is in effect in US Pacific Time Zone, PST otherwise).

Example of a single errata record (line wrapping is present for readability purposes only):


```
{
 "errata_id": 4439,
 "doc-id": "RFC7240",
 "errata_status_code": "Verified",
 "errata_type_code": "Technical",
 "section": 2,
 "orig_text": "    preference = token [ BWS \"=\" BWS word ]\r\n
            *( OWS \";\" [ OWS parameter ] )\r\n     parameter  = token [
       BWS \"=\" BWS word ]",
 "correct_text": "  preference = preference-parameter *( OWS \";\" [ OWS\r\n
                       preference-parameter ] )\r\n  preference-parameter =
       parameter / token\r\n",
 "notes": "Section 1.1 incorrectly states that \"word\" is defined in RFC
       7230.  It is not. Therefore, the syntax is completely under-
       specified.\r\n\r\nThe \"parameter\" rule, as defined in RFC 7231, is
       used in lots of other header field definitions successfully.  The only
       drawback is that \"parameter\" doesn't permit the use of \"OWS\" either
       side of the \"=\" character.\r\n\r\nThis change would also require
       changes to Section 1.1.",
 "submit_date": "2015-08-09",
 "submitter_name": "Martin Thomson",
 "verifier_id": 130,
 "verifier_name": "Barry Leiba",
 "update_date": "2016-02-23 15:34:45"
```

```
}
```

Output Format

The output of the Errata Merge Tool will be a properly-formatted HTML5 document, with the following general characteristics:

- Document text will be output in a fixed-width font, to match the original fixed-width formatting of the text RFCs.
- Page headers, footers, and page-feed (ASCII 12) characters will be stripped. Since the insertion of errata can change the length of the page into which they are inserted, these constructs do not make sense to include. The "`strip()`" function from the Rfcdiff tool may serve as a useful example of how such stripping can be performed.
- The HTML file will contain no styling information. All text styling will be applied using CSS files, described below.

Any errata associated with the document of the types specified on the command line will be incorporated into the document. As described in "Updating the Document," above.

The output of the tool must be verified to generate no console warnings (including deprecation warnings) and no errors in the most recent released versions of at least the four most popular desktop browsers, as determined by Statcounter.

CSS Files

The deliverables include three CSS files, used to style the errata that have been inserted. The first such file a baseline CSS file, will be applied to all documents. The baseline file includes styling suitable for printing (using the "`@media print {...}`" css directive), as well as any styles that are common between the two screen-oriented styles.

The second style is a color-coded CSS file, that will make use of color as a designator of errata-related changes, as described in this section. The third style is a monochrome CSS file that will make use of color-independent font styling (e.g., underlines, italics, and bold formatting), for the benefit of users with atypical color perception. The colored styling will be used be default.

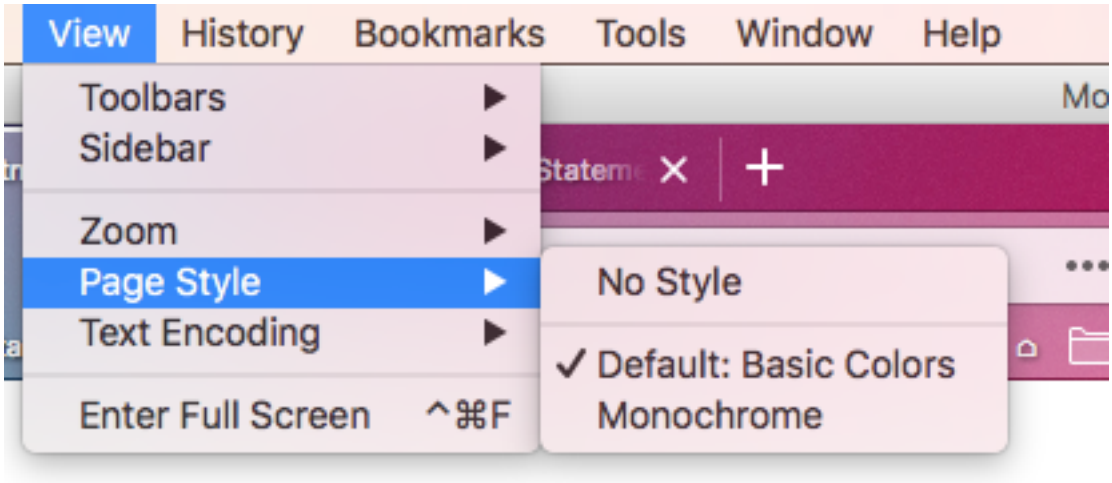
RFC with integrated errata will incorporate these CSS files using a technique similar to the one demonstrated below:

```
<link rel="stylesheet" type="text/css" href="errata-base.css" />

<link rel="stylesheet" type="text/css" href="errata-color.css"
     title="Default: Basic Colors" />

<link rel="alternate stylesheet" type="text/css" href="errata-monochrome.css"
     title="Monochrome" />
```

This approach will allow users of browsers that support the "alternate stylesheet" mechanism to select a style that relies on font variations rather than color to indicate the presence of errata:



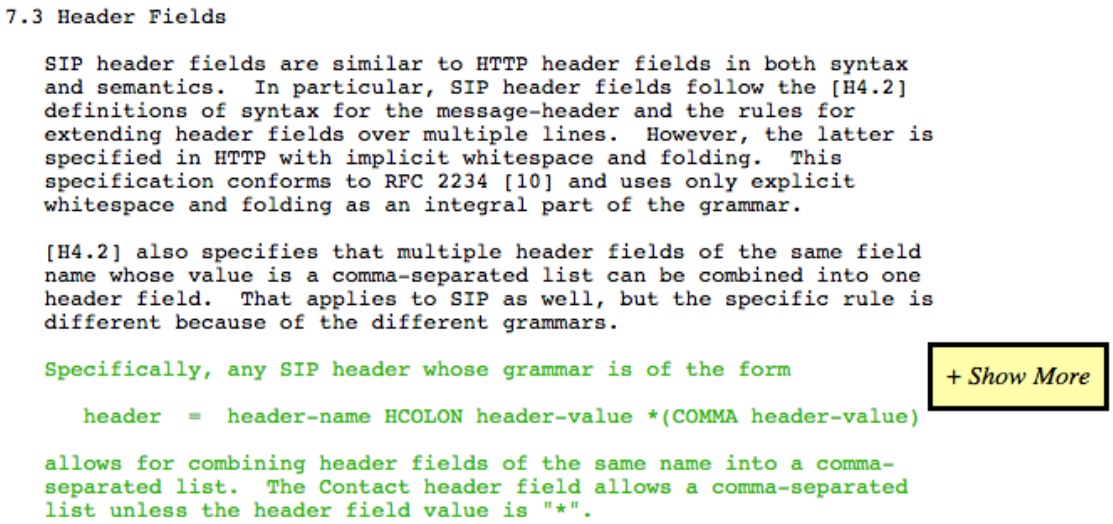
The "Updating the Document" section, above, calls for eight distinct styles within the document. The CSS files will format these as described in the following section. Note that the names used below match the examples from the "Updating the Document" section. The actual style names used are allowed to vary from these names. The specific font style and color schemes are suggestions, and input from the vendor and stakeholders is welcome.

Additional styles may be used to, e.g., highlight exact word changes, and to set corrected text off (e.g., with a strikethrough).

Basic Color Style Sheet

- Verified-inline-styling: Text will be green.
- Held-inline-styling: Text will be blue.
- Supplementary-styling: Text will be black on a light yellow background. Text will be hidden by default, with a small visible widget that allows for its expansion by either hovering or clicking.
- Verified-section-styling: Text will be black on a light green background.
- Held-section-styling: Text will be black on a light blue background. Text will be hidden by default, with a small visible widget that allows for its expansion by either hovering or clicking.
- Verified-endnote-styling: Same as verified-section-styling.
- Held-endnote-styling: Same as held section styling.
- Verified-banner-styling: Text will be black on a yellow background.

The following example shows an example of the placement of a verified inline patch:



Clicking on “Show More” would expand the errata information:

```
7.3 Header Fields

   SIP header fields are similar to HTTP header fields in both syntax
   and semantics.  In particular, SIP header fields follow the [H4.2]
   definitions of syntax for the message-header and the rules for
   extending header fields over multiple lines.  However, the latter is
   specified in HTTP with implicit whitespace and folding.  This
   specification conforms to RFC 2234 [10] and uses only explicit
   whitespace and folding as an integral part of the grammar.

   [H4.2] also specifies that multiple header fields of the same field
   name whose value is a comma-separated list can be combined into one
   header field.  That applies to SIP as well, but the specific rule is
   different because of the different grammars.

   Specifically, any SIP header whose grammar is of the form

      header  =  header-name HCOLON header-value *(COMMA header-value)

   allows for combining header fields of the same name into a comma-
   separated list.  The Contact header field allows a comma-separated
   list unless the header field value is "*".
```

+ *Collapse*

The preceding text has been updated by Technical Errata 3684 . The original text read:

```
   Specifically, any SIP header whose grammar is of the form

      header  =  "header-name" HCOLON header-value *(COMMA header-value)

   allows for combining header fields of the same name into a comma-
   separated list.  The Contact header field allows a comma-separated
   list unless the header field value is "*".
```

Errata Notes: *That is, remove the double quotes around the word `header-name' in the ABNF.*

```
7.3.1 Header Field Format

   Header fields follow the same generic header format as that given in
   Section 2.2 of RFC 2822 [3].  Each header field consists of a field
   name followed by a colon (":") and the field value.
```

Note that the preceding examples are meant to be illustrative, not prescriptive.

Monochrome Style Sheet

- Verified-inline-styling: Text will be bold.
- Held-inline-styling: Text will be italic.
- Supplementary-styling: Text will be black on a light grey background. Text will be hidden by default, with a small visible widget that allows for its expansion by either hovering or clicking.
- Verified-section-styling: Text will be bold and black on a light grey background.
- Held-section-styling: Text will be italic and black on a light grey background. Text will be hidden by default, with a small visible widget that allows for its expansion by either hovering or clicking.
- Verified-endnote-styling: Same as verified-section-styling.

- Held-endnote-styling: Same as held section styling.
- Verified-banner-styling: Text will be black on a light grey background.

Printer Style Sheet

Printer formatting will be identical to monochrome formatting, except that no sections will be hidden by default.

Project Resources and Equipment

Input Material

The source materials to be used for creating the tool are available at the following locations:

- RFC Documents in plain-text format are available individually from https://www.rfc-editor.org/rfc/rfc*XXXX*.txt, where "*XXXX*" represents the variable-length RFC number in question. The entire set of rfcs is also available via rsync at rsync.ietf.org::rfc.

- The entire errata database is available as a single JSON file at https://www.rfc-editor.org/errata.json. The primary intended consumer of this data is the RFC Errata Merge Tool, and so reasonable requests from the developer regarding format, location, and update frequency of this information will be considered.

Appendix

RFC Series	https://www.rfc-editor.org/
Database of Errata	https://www.rfc-editor.org/errata.php
Contractor Instructions	https://trac.tools.ietf.org/tools/ietfdb/wiki/ContractorInstructions?version=26
Simplistic Prototype	https://github.com/adamroach/patch-errata
RFC 822	https://www.ietf.org/rfc/rfc822.txt
RFC 854	https://www.ietf.org/rfc/rfc854.txt
RFC 1122	https://www.ietf.org/rfc/rfc1122.txt
Errata 571	https://www.rfc-editor.org/errata/eid571
Errata 1701	https://www.rfc-editor.org/errata/eid1701
Errata 2561	https://www.rfc-editor.org/errata/eid2561
rfc2html	https://pypi.org/project/rfc2html
The Rfcdiff Tool	https://tools.ietf.org/tools/rfcdiff/code
Statcounter	http://gs.statcounter.com/browser-market-share/desktop/worldwide