Workgroup:	Internet Engineering Task Force
Internet-Draft:	unicode-separated-values
Published:	8 March 2024
Intended Status:	Experimental
Expires:	9 September 2024
Author:	J. Henderson, Ed.

Unicode Separated Values (USV)

Abstract

Unicode Separated Values (USV) is a data format that uses Unicode separator characters.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents 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 9 September 2024.

Copyright Notice

Table of Contents

1. Introduction 3
   1.1. Requirements Language 3
   1.2. Media Type Language 3
   1.3. ABNF Language 4
2. Unicode symbols in use 4
3. Definition of the USV Format 4
   3.1. Data 4
   3.2. Unit 4
   3.3. Record 4
   3.4. Group 5
   3.5. File 5
   3.6. Header 5
   3.7. Escape (ESC) 6
   3.8. End of Transmission (EOT) 6
4. ABNF grammar 7
   4.1. Semantics 7
   4.2. Syntax 7
   4.3. Character classes 7
   4.4. Unicode symbols 8
5. Examples 8
   5.1. Hello World 8
   5.2. Hello World Goodnight Moon 9
   5.3. Units, Records, Groups, Files 9
   5.4. Articles 10
6. Source Code Examples 11
7. MIME media type registration for text/usv 12
   7.1. Optional parameters: charset, header 12
   7.2. Encoding considerations 12

7.3. Security considerations 12
7.4. Interoperability considerations 12
7.5. Published specification 12
7.6. Applications that use this media type 13
7.7. Additional information 13
8. IANA Considerations 13
9. Security Considerations 13
10. References 13
10.1. Normative References 13
10.2. Informative References 14
Appendix A. Appendix 1 14
Acknowledgements 14
Contributors 14
Author's Address 15

1. Introduction

Unicode Separated Values (USV) is a data format useful for exchanging and converting data between various spreadsheet programs, databases, and streaming data services. This RFC explains USV.

Additionally, we propose a new media type "text/usv", to be registered with IANA.

We provide information references for a USV git repository [usv-git-repository] and a programming implementation as a USV Rust crate [usv-rust-crate].

1.1. Requirements Language

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.

1.2. Media Type Language

The media type normative references are RFC 6838 [RFC6838], RFC 2046 [RFC2046], and RFC 4289 [RFC4289].

1.3. ABNF Language

The ABNF normative reference is RFC 5234 [RFC5234].

2. Unicode symbols in use

Separators:

- ␟ U+241F Symbol for Unit Separator (US)
- ␞ U+241E Symbol for Record Separator (RS)
- ␝ U+241D Symbol for Group Separator (GS)
- ␜ U+241C Symbol for File Separator (FS)

Modifiers:

- ␛ U+241B Symbol for Escape (ESC)
- ␄ U+2417 Symbol for End of Transmission (EOT)

3. Definition of the USV Format

3.1. Data

Data is comprises units, records, groups, and files.

3.2. Unit

A unit is comprises content characters. It runs until a unit separator.

Example unit and unit separator:

```
<CODE BEGINS> file "unit-and-unit-separator.usv"

aaa␟

<CODE ENDS>
```

3.3. Record

A record is comprises units. It runs until a record separator.

Example record and record separator:

```
<CODE BEGINS> file "record-and-record-separator.usv"

aaa␟bbb␟␞

<CODE ENDS>
```

3.4. Group

A group is comprises records. It runs until a group separator.

Example group and group separator:

```
<CODE BEGINS> file "group-and-group-separator.usv"

aaa␟bbb␟␞ccc␟ddd␟␞␝

<CODE ENDS>
```

3.5. File

A file is comprises groups. It runs until a file separator.

Example file and file separator:

```
<CODE BEGINS> file "file-and-file-separator.usv"

aaa␟bbb␟␞ccc␟ddd␟␞␝eee␟fff␟␞ggg␟hhh␟␞␝␜

<CODE ENDS>
```

3.6. Header

There may be an optional header appearing as the first item and with the same format as normal items. This header will contain names corresponding to the fields in the data, and should contain the same number of fields as the rest of data. The presence or absence of the header line should be indicated via the optional "header" parameter of this media type.

For example:

```
<CODE BEGINS> file "header.usv"

name␟name␟␞aaa␟bbb␟␞

<CODE ENDS>
```

3.7. Escape (ESC)

The Escape (ESC) symbol flips the purpose of the subsequent character:

- Escape + USV special character: the character is treated as content.
- Escape + USV typical character: the character is ignored.

USV with a unit that contains an Escape + End of Transmission, which is treated as content:

```
<CODE BEGINS> file "header.usv"

  a␛␃␛␄b␟

<CODE ENDS>
```

Escape + newline can be helpful for typical text editor line continuations:

```
<CODE BEGINS> file "header.usv"

          a␛␃␛␄b␟

<CODE ENDS>
```

3.8. End of Transmission (EOT)

The End of Transmission (EOT) symbol tells any reader that it can stop reading, and is especially useful for streaming data, such as to close a connection. EOT can also be useful for providing data files that contain USV data, then EOT, then extra non-USV information such as comments, images, attachments, etc.

- EOT tells the data reader that data streaming is done.
- EOT has no effect on the output content.

Example of a unit then an End of Transmission:

```
<CODE BEGINS> file "header.usv"

abc␞␜␟␄

<CODE ENDS>
```

4. ABNF grammar

4.1. Semantics

usv = *files

file = *groups

group = *records

record = *units

unit = *content-characters

4.2. Syntax

usv = (header-and-body / body) '*' ; anything after the body is chaff

header-and-body = 1*unit-run / 1*record-run / 1*group-run / 1*file-run

body = *unit-run / *record-run / *group-run / *file-run

file-run = *(*liner-character file *liner-character FS)

group-run = *(*liner-character group *liner-character GS)

record-run = *(*liner-character record *liner-character RS)

unit-run = *(*liner-character unit *liner-character US)

4.3. Character classes

content-character = typical-character / ESC '*'

typical-character = '*' - special-character

special-character = US / RS / GS / FS / ESC / EOT

escape-character = ESC (special-character / typical-character)

liner-character = CR / LF

4.4. Unicode symbols

US = U+241F Symbol for Unit Separator (US)

RS = U+241E Symbol for Record Separator (RS)

GS = U+241D Symbol for Group Separator (GS)

FS = U+241C Symbol for File Separator (FS)

ESC = U+241B Symbol for Escape (ESC)

EOT = U+2417 Symbol for End of Transmission (EOT)

CR = U+000D Carriage Return (CR)

LF = U+000A End of Line (EOL, LF, NL)

5. Examples

5.1. Hello World

This kind of data ...

```
<CODE BEGINS> file "hello-world.txt"

hello, world

<CODE ENDS>
```

... is represented in USV as two units:

```
<CODE BEGINS> file "hello-world.usv"

hello␟world␟

<CODE ENDS>
```

If you prefer to see one unit per line:

```
<CODE BEGINS> file "hello-world-with-lines.usv"

hello␟
world␟

<CODE ENDS>
```

5.2. Hello World Goodnight Moon

This kind of data ...

```
<CODE BEGINS> file "hello-world-goodnight-moon.txt"

[ hello, world ], [ goodnight, moon ]

<CODE ENDS>
```

... is represented in USV as two records, each with two units:

```
<CODE BEGINS> file "hello-world-goodnight-moon.usv"

hello␟world␟␞goodnight␟moon␟␞

<CODE ENDS>
```

If you prefer to see one record per line:

```
<CODE BEGINS> file "hello-world-goodnight-moon-with-lines.usv"

hello␟world␟␞
goodnight␟moon␟␞

<CODE ENDS>
```

5.3. Units, Records, Groups, Files

USV with 2 units by 2 records by 2 groups by 2 files:

```
<CODE BEGINS> file "units-records-groups-files.usv"

aᵘˢbᵘˢᴿˢcᵘˢdᵘˢᴿˢᴳˢeᵘˢfᵘˢᴿˢgᵘˢhᵘˢᴿˢᴳˢᶠˢiᵘˢjᵘˢᴿˢkᵘˢlᵘˢᴿˢᴳˢmᵘˢnᵘˢᴿˢoᵘˢpᵘˢᴿˢᴳˢᶠˢ

<CODE ENDS>
```

If you prefer to see one record per line:

```
<CODE BEGINS> file "units-records-groups-files-with-lines.usv"

aᵘˢbᵘˢᴿˢ
cᵘˢdᵘˢᴿˢ
ᴳˢ
eᵘˢfᵘˢᴿˢ
gᵘˢhᵘˢᴿˢ
ᴳˢ
ᶠˢ
iᵘˢjᵘˢᴿˢ
kᵘˢlᵘˢᴿˢ
ᴳˢ
mᵘˢnᵘˢᴿˢ
oᵘˢpᵘˢᴿˢ
ᴳˢ
ᶠˢ

<CODE ENDS>
```

5.4. Articles

USV can format paragraphs, such as in this example data stream of articles; note the units contain leading whitespace and trailing whitespace.

```
<CODE BEGINS> file "articles.usv"


Title One
␟
Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim
veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip.
␟␞
Title Two
␟
Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore
eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident,
sunt in culpa qui officia deserunt mollit anim id est laborum.
␟␞
Title Three
␟
Sed ut perspiciatis unde omnis iste natus error sit voluptatem accusantium
doloremque laudantium, totam rem aperiam, eaque ipsa quae ab illo inventore
veritatis et quasi architecto beatae vitae dicta sunt explicabo.
␟␞


<CODE ENDS>
```

6. Source Code Examples

Hello World using Rust and the USV crate

```
<CODE BEGINS> file "usv-rust-crate-units.rs"


use usv::*;
let input = "hello␟world␟";
let records = input.units().collect();


<CODE ENDS>
```

Hello World Goodnight Moon using Rust and the USV crate

```
<CODE BEGINS> file "usv-rust-crate-records.rs"


use usv::*;
let input = "hello␟world␟␞goodnight␟moon␟␞";
let records = input.records().collect();


<CODE ENDS>
```

7. MIME media type registration for text/usv

This section provides the MIME media type registration application information.

To: ietf-types@iana.org

Subject: Registration of MIME media type text/usv

MIME media type name: text

MIME subtype name: usv

Required parameters: none

7.1. Optional parameters: charset, header

Common usage of USV is UTF-8, but other character sets defined by IANA for the "text" tree may be used in conjunction with the "charset" parameter.

The "header" parameter indicates the presence or absence of the header line. Valid values are "present" or "absent". Implementors choosing not to use this parameter must make their own decisions as to whether the header line is present or absent.

7.2. Encoding considerations

This media type uses LF to denote line breaks. However, implementors should be aware that some implementations may not conform i.e. may incorrectly use other values.

7.3. Security considerations

USV files contain passive text data that should not pose any risks. However, it is possible in theory that malicious binary data may be included in order to exploit potential buffer overruns in the program processing USV data. Additionally, private data may be shared via this format (which of course applies to any text data).

7.4. Interoperability considerations

Implementors should "be conservative in what you do, be liberal in what you accept from others" (RFC 793 [8]) when processing USV data.

Implementations deciding not to use the optional "header" parameter must make their own decision as to whether the header is absent or present.

7.5. Published specification

https://github.com/sixarm/usv

7.6. Applications that use this media type

Spreadsheet programs, such as with import/export. Database programs, such as with loading/saving text. Data conversion utilities.

7.7. Additional information

Magic number(s): none

File extension(s): usv

Apple macOS File Type Code(s): TEXT

Intended usage: COMMON

Author/Change controller: IESG

Contact: Joel Parker Henderson <joel@joelparkerhenderson.com>

8. IANA Considerations

We are requesting IANA to create a standard MIME media type "text/usv".

We have filed an IANA request for this, with same contact information.

9. Security Considerations

This document should not affect the security of the Internet.

10. References

10.1. Normative References

[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/info/rfc8174>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/info/rfc6838>.

[RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <https://www.rfc-editor.org/info/rfc2046>.

[RFC4289] Freed, N. and J. Klensin, "Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures", BCP 13, RFC 4289, DOI 10.17487/RFC4289, December 2005, <https://www.rfc-editor.org/info/rfc4289>.

10.2. Informative References

[usv-git-repository] Henderson, J., "USV repository at https://github.com/sixarm/usv", 2022.

[usv-rust-crate] Henderson, J., "USV rust crate at https://crates.io/crates/usv", 2024.

[usv-to-csv-rust-crate] Henderson, J., "usv-to-csv command line tool at https://crates.io/crates/usv-to-csv", 2024.

[csv-to-usv-rust-crate] Henderson, J., "csv-to-usv command line tool at https://crates.io/crates/csv-to-usv", 2024.

[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/info/rfc2119>.

Appendix A. Appendix 1

This becomes an Appendix

Acknowledgements

The author would like to thank Y. Shafranovich, author of the CSV RFC, which provided guidance for this USV RFC.

A special thank you goes to P.X.V.

Contributors

Thanks to all of the contributors.

Joel Parker Henderson
Email: joel@joelparkerhenderson.com

Author's Address

Joel Parker Henderson (EDITOR)
601 Van Ness Ave #E3-359
San Francisco, CA 94102
United States of America
Phone: 1-415-317-2700
Email: joel@joelparkerhenderson.com
URI: https://linkedin.com/in/joelparkerhenderson