Abstract

The record-jar format provides a method of storing multiple records with a variable repertoire of fields in a text format. This document provides a description of the format. Comments are solicited and should be addressed to the mailing list 'record-jar@yahoogroups.com' and/or the author.
1. Introduction

The record-jar format was originally described by The Art of Unix Programming [AOUP]. This format is useful for storing information in a human-readable text form, while making the data available for machine processing. It is a flexible format, since it provides for an arbitrary range of fields in any given record and can be used to store data with variable length and content.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].
2. Format and Grammar

The record-jar format is described by the following ABNF ([RFC4234]):

record-jar   = [encodingSig] [separator] *record
record       = 1*field separator
field        = (field-name field-sep field-body CRLF)
field-name   = (ALPHA / DIGIT)
  *[ (ALPHA / DIGIT / "-" ) (ALPHA / DIGIT) ]
field-sep    = *SP " : " *SP
field-body   = *(character / continuation)
continuation = ["\"] LWSP
separator    = [blank-line] *("%%" [comment] CRLF)
comment      = SP *69(character)
character    = SP / ASCCHAR / UNICHAR / ESCAPE
encodingSig  = "%%encoding" field-sep
  *(ALPHA / DIGIT / ":" / ":" ) CRLF
blank-line   = WSP CRLF

; ASCII characters except %x26 (&) and %x5C (\)
ASCCHAR      = %x21-25 / %x27-5B / %x5D-7E
UNICHAR      = %x80-%x10FFFF ; Unicode chars
ESCAPE       = "\" ("\" / ":" / ":" / ":" / ":" )
  / "\#" 2*HEXDIG ";"

The record-jar format consists of character data that forms a sequence of records. Each record is separated from other records by at least one line beginning with the sequence "%%" (%x25.25). Records are made up of one or more fields and a record MAY contain as many or as few fields as are necessary to convey the necessary data. Empty records and blank lines are ignored.

A field is a single, logical line of characters from the Universal Character Set [Unicode], comprised of three parts: the field-name, the field-separator, and the field body.

The field-name is an identifier. It MUST be a sequence of letters and digits from the US-ASCII character set [ISO646]. A field-name SHOULD be treated as case sensitive and MUST NOT contain any spaces. Upper and lowercase letters are often used to visually break up the name, for example using CamelCase. It is a common convention that field names use an initial capital letter, although this is not enforced. The hyphen-minus character ("-", %x2D) MAY be used to separate parts of the name visually, however, it MUST NOT appear at the beginning or end of a field-name.

The field separator (field-sep) is the colon character ("":", %x3A). The separator MAY be surrounded on either side by any amount of
horizontal whitespace (tab or space characters). The normal
convention is one space on each side.

The field-body contains the data value. Logically, the field-body
consists of a single line of text using any combination of characters
from the Universal Character Set followed by a CRLF (newline). The
carriage return, newline, and tab characters, when they occur in the
data value stored in the field-body, are represented by their common
backslash escapes ("\r", "\n", and "\t" respectively). See
Section 2.3 for more information on escape sequences.

2.1. Folding of Field Values

For convenience and compatibility with various protocols, individual
lines SHOULD NOT exceed 72 total characters. (Separator lines MUST
NOT exceed 72 characters.) The field-body portion of a field can be
split into a multiple-line representation; this is called "folding".
Successive lines in the same field-body begin with one or more
whitespace characters. When processing the record-jar format, the
linear whitespace (including the newline and any preceding spaces)
is consumed by the processor and the two parts of the field-body
joined to form a single, logical line. For example:

Eulers-Number : 2.718281828459045235360287471
35266249775724709369959574966967627724076630353547
5945713821785251664274274663919320030599218174135...

Figure 2: Example of Folding

Note that a 72-character limit effectively limits the length of the
field-name to no more than 71 characters (since the field separator
MUST appear on the same line with the field-name).

In some cases, the field-body contains spaces that are important to
the data. To accurately preserve whitespace in the document, an
optional line-continuation character (backslash, %x5C) MAY be
included to delimit and separate whitespace to be preserved from
whitespace that will be removed by the processor. The line-
continuation character and any whitespace that follows it (including
whitespace at the beginning of the continuing field-body on the next
line) MUST be consumed by the processor when reading the file.
Whitespace appearing before the line-continuation MUST NOT be
consumed. Use of the line continuation character makes the
whitespace visible in the file.

In other cases, the field-body might contain natural language text,
and, while it is readily apparent that many languages use spaces to
separate words, others, such as Japanese or Thai, do not.
Implementations MAY, in the absence of line continuation characters,
replace the continuation sequence (the line break and surrounding whitespace) in a folded line with a single ASCII space (%x20), however, implementations SHOULD just remove the continuation sequence altogether in order to avoid causing unnatural breaks in the text.

Here are some examples:
SomeField : This is some running text \ that is continued on several lines \ and which preserves spaces between \ the words.
%%% AnotherExample: There are three spaces \ between 'spaces' and 'between' in this record.
%%% SwallowingExample: There are no spaces between \ the numbers one and two in this example 1\ 2.
%%%  
Figure 3: Example of Folding with Preserved Whitespace

Note that blank lines are consumed by the folding rules. Consider this record:
%%% SomeText: \ \ \  
%%%  
Figure 4: Whitespace Folding Example

The field-body of the field "SomeText" is the empty string. On the first line, all of the spaces are contained by the field-separator. The spaces on the subsequent lines are part of the folding whitespace production.

2.2. Comments

Comments MAY be included in the body of the record-jar document by placing them at the end of a separator line. The comment MUST be separated by at least one space from the "%%" sequence that introduces the separator.

Multiple separators MAY appear between records. Logically this appears to result in records that contain no fields: records containing no fields MUST be ignored by a processor.

Folding of comments is not permitted; instead multiple comment lines
MUST be used. Comments can not appear in the body of a record. For example:

%% this is a comment.
Record: goes here
%
%% here is another comment
%% that appears on multiple lines
Record: another record
%% a final comment
%

Figure 5: Comment example

2.3. Characters, Encodings, and Escapes

By default, the record-jar format uses the UTF-8 encoding of Unicode (see [RFC3629]). If an application, protocol, or specification permits an encoding other than UTF-8 to be used in the file, it SHOULD also support reading the encoding from the encoding signature. The encoding signature, when present, MUST be the very first line of the file. If the encoding signature is not present, an application or protocol MAY attempt to infer the encoding using other means. Record-jar files SHOULD include an encoding signature, even if one is not required, whenever the application, protocol, or specification permits one.

A file that uses the UTF-16 or UTF-32 encoding MAY also include a Byte Order Mark (U+FEFF) as the first sequence of two octets (in the case of UTF-16) or four octets (in the case of UTF-32) in the file, just preceding the encoding signature.

Some applications, protocols, or specifications require that the record-jar file use some other, non-Unicode, legacy character set. In particular, an application, protocol, or specification will frequently support only the US-ASCII character set ([ISO646]).

Here is an example of the encoding signature for the UTF-8 encoding of Unicode:

%%encoding:UTF-8

Figure 6: Example of an Encoding Signature

Printable ASCII characters excepting backslash ("\") and ampersand ("&") are represented as themselves.

Non-ASCII values MAY be included in a record-jar file in several ways. For portability, the best mechanism is to use escape sequences in the field-body. Exclusive use of escape sequences results in a
pure ASCII text file.

Non-ASCII characters MAY be represented using the character’s Unicode value represented using the Numeric Character Reference format adapted from XML; the sequence "&#x" (%x26.23.78) is followed by the character’s Unicode scalar value in hex followed directly by the semi-colon character (";", %x3B). Leading zeroes MAY be omitted. For example, the EURO SIGN is U+20AC and could be represented as "&#x20ac;".

Non-ASCII characters MAY also be represented as their associated octet sequence in the file’s character encoding. For example, the EURO SIGN would be represented as %xE2.82.AC in UTF-8.

The characters for carriage return, newline, and tab when considered as part of the data (and not the file format itself) are represented by the traditional escape sequences "\r" (%x5C.72), "\n" (%x5C.6E), and "\t" (%x5C.74) respectively. The character backslash is represented by "\" (%x5C.5C), while the ampersand character is represented by "\&" (%x5C.26). A single backslash at the end of a line indicates continuation, as discussed in Section 2.1. Otherwise a single backslash followed by some other character in the data is an error, although a record-jar processor MAY choose to interpret it as a backslash.
3. Examples

Here is the canonical example from [AOUP]:
Planet: Mercury
Orbital-Radius: 57,910,000 km
Diameter: 4,880 km
Mass: 3.30e23 kg
%
Planet: Venus
Orbital-Radius: 108,200,000 km
Diameter: 12,103.6 km
Mass: 4.869e24 kg
%
Planet: Earth
Orbital-Radius: 149,600,000 km
Diameter: 12,756.3 km
Mass: 5.972e24 kg
Moons: Luna

A more complete example showing more of the various features in the format is described in [RFC4646]. The data shown here is taken from the Language Subtag Registry defined by [RFC4646]:
%
Type: language
Subtag: ia
Description: Interlingua (International Auxiliary Language \ Association)
Added: 2005-08-16
%
Type: language
Subtag: id
Description: Indonesian
Added: 2005-08-16
Suppress-Script: Latn
%
Type: language
Subtag: nb
Description: Norwegian Bokmål
Added: 2005-08-16
Suppress-Script: Latn
%
4. References

4.1. Normative References


4.2. Informative References


Appendix A. Acknowledgements

Thanks to Eris S. Raymond for his gracious permission to both reference and quote The Art of Unix Programming in this document. Without his work, this document would likely not exist.

Contributors to this document include: Stephane Bortzmeyer, John Cowan, Frank Ellerman, Doug Ewell.

The IETF LTRU working group adopted record-jar format on John Cowan’s suggestion. That effort required record-jar to be documented and many people in that group contributed to this work there: the author thanks everyone who participated in that effort, even though names cannot be mustered here.
Author’s Address

Addison Phillips (editor)
Yahoo! Inc.

Email: addison@inter-locale.com
URI: http://www.inter-locale.com
Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.

Disclaimer of Validity

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2006). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.