

Case-Sensitive String Support in ABNF

Abstract

This document extends the base definition of ABNF (Augmented Backus-Naur Form) to include a way to specify US-ASCII string literals that are matched in a case-sensitive manner.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at <http://www.rfc-editor.org/info/rfc7405>.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	2
2. Updates to RFC 5234	2
2.1. Terminal Values - Literal Text Strings	3
2.2. ABNF Definition of ABNF - char-val	4
3. Security Considerations	4
4. Normative References	4
Author's Address	4

1. Introduction

The base definition of ABNF (Augmented Backus-Naur Form) supports US-ASCII string literals. The matching of these literals is done in a case-insensitive manner. While this is often the desired behavior, in some situations, case-sensitive matching of string literals is needed. Literals for case-sensitive matching must be specified using the numeric representation of those characters, which is inconvenient and error prone both to write and read.

This document extends ABNF to have two different types of US-ASCII string literals. One type is matched using case-sensitive matching, while the other is matched using case-insensitive matching. These types are denoted using type prefixes similar to the type prefixes used with numeric values. If no prefix is used, then case-insensitive matching is used (as is consistent with previous behavior).

This document is structured as a set of changes to the full ABNF specification [RFC5234].

2. Updates to RFC 5234

This document makes changes to two parts of [RFC5234]. The two changes are as follows:

- o Replace the last half of Section 2.3 of [RFC5234] (beginning with "ABNF permits the specification of literal text strings") with the contents of Section 2.1.
- o Replace the <char-val> rule in Section 4 of [RFC5234] with the contents of Section 2.2.

2.1. Terminal Values - Literal Text Strings

ABNF permits the specification of literal text strings directly, enclosed in quotation marks. Hence:

```
command      = "command string"
```

Literal text strings are interpreted as a concatenated set of printable characters.

NOTE:

The character set for these strings is US-ASCII.

Literal text strings in ABNF may be either case sensitive or case insensitive. The form of matching used with a literal text string is denoted by a prefix to the quoted string. The following prefixes are allowed:

```
%s          = case-sensitive
%i          = case-insensitive
```

To be consistent with prior implementations of ABNF, having no prefix means that the string is case insensitive and is equivalent to having the "%i" prefix.

Hence:

```
rulename = %i"aBc"
```

and:

```
rulename = "abc"
```

will both match "abc", "Abc", "aBc", "abC", "ABc", "aBC", "AbC", and "ABC".

In contrast:

```
rulename = %s"aBc"
```

will match only "aBc" and will not match "abc", "Abc", "abC", "ABc", "aBC", "AbC", or "ABC".

In the past, the numerical specification of individual characters was used to define a case-sensitive rule.

For example:

```
rulename    = %d97 %d98 %d99
```

or

```
rulename    = %x61.62.63
```

will match only the string that comprises only the lowercase characters, abc. Using a literal text string with a prefix has a clear readability advantage over the old way.

2.2. ABNF Definition of ABNF - char-val

```
char-val    = case-insensitive-string /  
              case-sensitive-string
```

```
case-insensitive-string =  
    [ "%i" ] quoted-string
```

```
case-sensitive-string =  
    "%s" quoted-string
```

```
quoted-string = DQUOTE *(%x20-21 / %x23-7E) DQUOTE  
                ; quoted string of SP and VCHAR  
                ; without DQUOTE
```

3. Security Considerations

Security is truly believed to be irrelevant to this document.

4. Normative References

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

Author's Address

Paul Kyzivat
Massachusetts
United States

EMail: pkyzivat@alum.mit.edu

