Network Working Group
Internet-Draft
Expires: January 10, 2005

J. Klensin

July 12, 2004

A Name Munging Protocol
draft-klensin-name-munging-03.txt

Status of this Memo

This Internet-Draft will expire on January 10, 2005.

Copyright Notice

Abstract

As one works on internationalization issues for DNS, email, and other protocols, it becomes clear that the various encodings and transformations required, while not intrinsically difficult, can be an impediment to rapid conversion of applications to international form and to rapid prototyping of new applications. This document proposes a new, lightweight, protocol that can be used to make such conversions, rather than incorporating the needed tables and algorithms into each application.

Table of Contents

1. Introduction

A variety of new and upcoming protocols, most, but not all, of them associated with internationalization, require that data be presented in, or mapped into, encoding forms that are specialized and largely unique to the Internet or those protocols. The trend arguably started with the introduction of quoted-printable into MIME [RFC1341] and has continued to more recent DNS internationalization work [RFC3490] and developing errors in internationalization of electronic mail [I-D.hoffman-imaa]. These encodings are at least complex enough that testing for interoperability and accuracy is perceived to be needed. Even though they are not, intrinsically, very hard, the process of getting the needed code incorporated and tested may be sufficient to discourage or delay internationalization of some applications, including those that are built around short scripts.

This document describes a protocol -- designed for use over either TCP or UDP -- that can be passed short strings for conversion from one encoding to another. There are various samples, testbeds, and web pages today that can do some of these conversions, but they are not general (few of them handle more than one or two conversions), and they are really not compatible with use in applications implementation (regardless of whether they can be used in testing or not). The core code in those samples and tests could presumably be adapted to support this protocol.

2. The Protocol

The protocol is designed to be as simple as possible, following the general "send packet containing one line, get another line back" model used in finger [RFC1288] and whois [RFC0954]. That model is traditional and well-proven in the Internet, but, by today's standards, sacrifices a high degree of security for performance and should be used with appropriate care. The appendix contains an outline description of a possible variant on this protocol for situations in which it is desired to have, within the protocol itself, some degree of authentication that the intended server was reached and the response received is from it, but, in general some type of authenticated tunnel mechanism will be more satisfactory. See Section 5, Section 4, and Section 7 for additional discussion of these issues. For performance, the protocol is designed to be used over either UDP or TCP, as meets the needs of the application. The TCP variation on the above is, obviously, "open a connection, send a line, remote system sends a line back and closes the connection". The lines are defined as follows:

2.1 Inputs

The input line consists of
- A Version number, "1" for this variation on the protocol.
- An ASCII space (i.e., an octet containing hex 20)
- A source-indication string
- An ASCII space
- A target-indication string
- An ASCII space
- A bit count, expressed as an ASCII numeral
- An ASCII space
- The source bit string

2.2 Element definitions

The version number is a positive integer, defined as "1" in this version of the protocol. Implementations of this version of the protocol are required to check the version number and, if it is not "1", to return a string consisting of "550 bad version number" (see below). The indication strings are positive integers, registered with IANA and described in Section 2.3, below.

The integers for the version number, indicator strings, and bit count are expressed as decimal numbers using ASCII digits. They, and the single ASCII space character that follows each one, are protocol elements and are not intended to be internationalized.

The source string will be a simple string of bits, of length specified by the bit count (with the first bit counted as one). While it will normally be an integral number of octets, some special encodings may not permit this, so any extra bits are ignored. For convenience, the bit count may be specified as an ASCII asterisk ("*", an octet containing hex 2A), in which case the server will examine the string for the first pair of octets containing, respectively, hex 0D and 0A (the usual CRLF convention) and consider it to terminate immediately before those characters.

2.3 Initial List of Encodings

As discussed below, IANA is expected to set up a registry of encoding codes for use in this protocol. That list is initially:

0 Information and debugging option. If 0 appears as the input indicator, the rest of the input line is ignored and the server returns a reply code of "000 " followed by a blank-separated list of the indicator codes it recognizes. If 0 appears as the output indication, the input is copied to the output, also with a reply code of 000, and returned.

1 UCS-4
2 Unicode (UCS-2)
3 IDNA Punycode
4 The IMAA encoding scheme described in [I-D.hoffman-imaa]
5 UTF-8
6 ISO 8859-1
7 Unicode written as a blank-separated list of four or more hexadecimal digit codes (written in ASCII), and with each set of codes optionally preceded by "U+" or "u+". The hexadecimal codes "A"..."F" may be written in either upper or lower case.
8 Nameprep (stringprep profile only, no punycode)
9 SASLprep (stringprep profile only, no punycode)
10 iSCSIprep (stringprep profile only, no punycode)

There is no requirement that every server support every encoding, although it is expected that every server will support the "0" encoding for test purposes. Issues of how a client locates an appropriate server are outside the scope of this specification (see Section 5).

2.4 Outputs

The version 1 output consists of
- a three-digit (ASCII) reply code (codes listed below)
- an ASCII space
- a bit count
- an ASCII space
- a string

The bit count, space, and string are as described above, but the "*" convention will not be used.

2.5 Reply codes

The following reply codes are specified for use in this protocol. If, for some reason (presumably due to a new version of the protocol on the server), the three-digit code returned is not listed below, only the first digit should be examined. A first digit of zero indicates that the string returned contains either the original string or a recoding of it; a first digit of 5 indicates that the recoding failed and the string is either zero-length or contains an explanation in ASCII characters.

000 String translated
001 String not translated
500 Service not available to you

501 Input encoding type not recognized
502 Output encoding type not recognized
503 Bit count exceeds length of line
504 No translation available, i.e., the server recognizes the input encoding and the output encoding, but has no mapping between them.
505 Translation failed or input string invalid, e.g., the input string was not a possible example of the input encoding specified.
506 Input string too long.
550 Wrong version number, i.e., version number specified is not understood by this server.
6yz Authentication, authorization, or other security problem. Reserved for future use.

3. Examples

```
1 6 0 10 teststring
000 10 teststring

1 6 3 9 F ltstr÷m
```

I.e., with the second and eighth characters as a-with-diaeresis (U+00E4) and o-with-diaeresis (U+00F6) respectively.

```
000 12 xn--fltstrm-5wa1o
```

4. Signed Messages and Business Arrangements

In today's sometimes-hostile Internet environment, two questions immediately arise about a protocol that is designed to be this simple. One is how one tells that the returned string is the intended one, i.e., that it came from the designated server and that some is taking responsibility for that server's results. The other is how to get someone to provide this service, especially if it is to be called from production-scale applications protocols. Either or both requirements might be satisfied by sending digitally-signed strings. In the input (business model) case, we might imagine a subscription service with registered users, with the digital signature used to authenticate the query as coming from a subscriber and/or authorize billing. In the output case, we might imagine a family of certified servers (using a certification process that lies outside this specification) able to sign the responses with a key the user or application would trust. Both of these issues, and the protocol changes that would be required, should be examined in depth before this protocol is published.

At least for the TCP version of the protocol, both of these issues could be dealt with independently of the protocol itself, e.g., by running it over fully-authenticated IPSec or SSL.

This specification does not cover identification and location of appropriate servers.

5. Availability

As suggested elsewhere in this document, it is expected that this protocol will be used primarily within controlled environments, or with servers accessed through tunnels that provide both client and server authentication. Sample PERL source for client and server implementations, contributed by Paul Hoffman, will be deposited with the RFC Editor.

6. IANA Considerations

IANA has assigned reserved port number 3950 for both the UDP and TCP variations of this protocol.

A registry of encoding type indicator strings is also required, with a sequential integer to be assigned to each type of encoding registered and the list in Section 2.3 used to initialize that registry. IANA is requested to accept registrations only with contact information and a reference that defines the encoding involved, but, since there is no shortage of integers, checking and evaluation of such requests is not required except to the degree required to prevent denial of service attacks on IANA itself.

The conversions defined and supported are one-to-one mappings only. This protocol, or at least this version of the protocol, does not support any one-many, or otherwise ambiguous, mappings.

No IANA registry is required for version numbers: versions other than the one described here will require a revised version of this specification.

7. Security Considerations

As mentioned in Section 4, there is an attack on this protocol, especially in which it is used over UDP, in which a response is sent to the client application that contains an encoding of a different string than the one that was submitted. If that string is used without inspection or review by the client, various bad things might happen. Signed strings, as discussed above, might protect against that problem, but only if keys are properly protected and verified. If assurances are needed that the server is the intended one, it is recommended that the protocol be operated over an appropriately configured tunnel. An extension for SASL negotiation is possible in principle, but would be incompatible with operation of the protocol over UDP and would be likely to defeat the intent of a very high

performance protocol design.

For those situations in which authentication of the server (and response source) to the client is useful, an alternative version of the protocol is specified with a minimal digest challenge-response mechanism. Since that mechanism depends on a secret shared between the client and server, it is likely to be useful, if at all, in restricted environments such as a small department or group that does not consider whatever group-isolation firewalls or similar mechanisms adequate to protect against server spoofing attacks. For any sort of public use, the mechanism is subject to the well-known problems of a secret known to hundreds of people and is hence likely to be useless. As discussed elsewhere in this document, authenticity and integrity protection when public servers and the public Internet are involved are probably best dealt by running this protocol within an authenticated and cryptographically protected tunnel or, in principle, by extending the protocol to utilize some sort of public key message-signing mechanism.

8. Acknowledgements

The author would like to express appreciation to Patrik Faltstrom and Leslie Dangle, who made some suggestions at a early formative stage of this proposal and, in particular, pointed out the desirability of digitally signing the strings. Paul Hoffman made a number of other useful suggestions and contributed the first implementation. Simon Josefsson suggested the addition of type codes for several additional stringprep profiles. And the decision to modify the protocol to add a version number emerged from a discussion with Harald Alvestrand.

9. References

9.1 Normative References

[RFC2831] Leach, P. and C. Newman, "Using Digest Authentication as a SASL Mechanism", RFC 2831, May 2000.

9.2 Informative References

[I-D.hoffman-imaa]
Hoffman, P. and A. Costello, "Internationalizing Mail Addresses in Applications (IMAA)", draft-hoffman-imaa-03 (work in progress), October 2003.

[RFC0954] Harrenstien, K., Stahl, M. and E. Feinler, "NICNAME/WHOIS", RFC 954, October 1985.

[RFC1288] Zimmerman, D., "The Finger User Information Protocol", RFC

1288, December 1991.

[RFC1341] Borenstein, N. and N. Freed, "MIME (Multipurpose Internet Mail Extensions): Mechanisms for Specifying and Describing the Format of Internet Message Bodies", RFC 1341, June 1992.

[RFC2617] Franks, J., Hallam-Baker, P., Hostetler, J., Lawrence, S., Leach, P., Luotonen, A. and L. Stewart, "HTTP Authentication: Basic and Digest Access Authentication", RFC 2617, June 1999.

[RFC3490] Faltstrom, P., Hoffman, P. and A. Costello, "Internationalizing Domain Names in Applications (IDNA)", RFC 3490, March 2003.

Author's Address

John C Klensin
1770 Massachusetts Ave, #322
Cambridge, MA 02140
USA

Phone: +1 617 491 5735
EMail: john-ietf@jck.com

Intellectual Property Statement

Disclaimer of Validity

Copyright Statement

Acknowledgment

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