Network Working Group
Request for Comments: 5483
Category: Informational

L. Conroy
RMRL
K. Fujiwara
JPRS

March 2009

ENUM Implementation Issues and Experiences

Status of This Memo

Copyright Notice

This document captures experiences in implementing systems based on the ENUM protocol and experiences of ENUM data that have been created by others. As such, it clarifies the ENUM and Dynamic Delegation Discovery System standards. Its aim is to help others by reporting both what is "out there" and potential pitfalls in interpreting the set of documents that specify the ENUM protocol. It does not revise the standards but is intended to provide technical input to future revisions of those documents.

Table of Contents

1. Introduction

1.1. Document Goal

The goal of this document is to clarify the ENUM and Dynamic Delegation Discovery System (DDDS) standards. It does not itself revise ENUM or DDDS standards but is intended to provide technical input to future revisions of those documents. It also serves to advise implementers on the pitfalls that they may find. It highlights areas where ENUM implementations have differed over interpretation of the standards documents or have outright failed to implement some features as specified.

As well as providing clarifications to standards text, this document also mentions potential choices that can be made, in an attempt to help foster interworking between components that use this protocol. The reader is reminded that others may make different choices.

The core specifications for the E.164 Number Mapping (ENUM) protocol [RFC3761] and the Dynamic Delegation Discovery System (DDDS) [RFC3403] [RFC3401] [RFC3402] [RFC3404] [RFC3405] are defined elsewhere. Unfortunately, this document cannot provide an overview of the specifications, so the reader is assumed to have read and understood the complete set of ENUM normative documents.

The Domain Name System (DNS) is ENUM’s database. ENUM uses the NAPTR (Naming Authority Pointer) resource record type to store its DDDS rules into DNS domains. ENUM relies on DNS services. Thus, it is also important for ENUM implementers to carry out a thorough analysis of all of the existing DNS standard documents to understand what services are provided to ENUM and what load ENUM provisioning and queries will place on the DNS.

A great deal of the rationale for making the choices listed in this document is available to those who explore the standards. The trick of course is in understanding those standards and the subtle implications that are involved in some of their features. In almost all cases, the choices presented here are merely selections from values that are permissible within the standards.

1.2. Terminology

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. Character Sets and ENUM

2.1. Character Sets - Non-ASCII Considered Harmful

[RFC3403] and [RFC3761] specify respectively that NAPTR resource records and ENUM support Unicode using the UTF-8 encoding defined in [RFC3629]. This raises an issue when implementations use "single byte" string-processing routines. If there are multi-byte characters within an ENUM NAPTR, incorrect processing may well result from these UTF-8-unaware systems.

The UTF-8 encoding has a US-ASCII equivalent range, so that all characters in US-ASCII [ASCII] from 0x00 to 0x7F hexadecimal have an identity map to the UTF-8 encoding; the encodings are the same. In UTF-8, characters with Unicode code points above this range will be encoded using more than one byte, all of which will be in the range 0x80 to 0xFF hexadecimal. Thus, it is important to consider the different fields of a NAPTR and whether or not multi-byte characters can or should appear in them.

In addition, characters in the non-printable portion of US-ASCII (0x00 to 0x1F hexadecimal, plus 0x7F hexadecimal) are "difficult". Although NAPTRs are processed by machine, they may sometimes need to be written in a human-readable form. Specifically, if NAPTR content is shown to an end user so that he or she may choose, it is imperative that the content is human-readable. Thus, it is unwise to use non-printable characters even if they lie within the US-ASCII range; the ENUM client may have good reason to reject NAPTRs that include these characters as they cannot readily be presented to an end user.

There are two numeric fields in a NAPTR: the ORDER and PREFERENCE/ PRIORITY fields. As these contain binary values, no risk is involved because string processing should not be applied to them. The string-based fields are the Flags, Services, and Regexp fields. The Replacement field holds an uncompressed domain name, encoded according to the standard DNS mechanism [RFC1034][RFC1035]. The Internationalised Domain Name (IDN) can be supported (as specified in [RFC3490], [RFC3491], and [RFC3492]). Any such IDN MUST be further encoded using Punycode [RFC3492]. As the Replacement field holds a domain name that is not subject to replacement or modification (other than Punycode processing), it is not of concern here.

Taking the string fields in turn, the Flags field contains characters that indicate the disposition of the NAPTR. This may be empty, in which case the NAPTR is "non-terminal", or it may include a flag character as specified in [RFC3761]. These characters all fall into the printable US-ASCII equivalent range, so multi-byte characters cannot occur.

The Services field includes the DDDS Application identifier ("E2U") used for ENUM, a set of Enumservice identifiers, any of which may embed the ':' separator character, together with the '+' character used to separate Enumservices from one another and from this DDDS Application identifier. In Section 2.4.2 of [RFC3761], Enumservice identifier tokens are specified as 1*32 ALPHA/DIGIT, so there is no possibility of non-ASCII characters in the Services field.

2.1.1. Non-ASCII in the Regular Expression Field

The Regexp field is more complex. It forms a sed-like substitution expression, defined in [RFC3402], and consists of two sub-fields:

- a POSIX Extended Regular Expression (ERE) sub-field [IEEE.1003-2.1992]

- a replacement (Repl) sub-field [RFC3402].

Additionally, [RFC3402] specifies that a flag character may be appended, but the only flag currently defined there (the 'i' case-insensitivity flag) is not appropriate for ENUM -- see Section 2.2.

The ERE sub-field matches against the "Application Unique String"; for ENUM, this is defined in [RFC3761] to consist of digit characters, with an initial '+' character. It is similar to a global-number-digits production of a tel: URI, as specified in [RFC3966], but with visual-separators removed. In short, it is a telephone number (see [E.164]) in restricted format. All of these characters fall into the US-ASCII equivalent range of UTF-8 encoding, as do the characters significant to the ERE processing.

Strictly, the ERE might include other characters. The ERE could include choice elements matching against different items, some of which might not be an ENUM Application Unique String. Those alternative matching elements might conceivably include non-ASCII characters. As an operational issue, it is not reasonable to include such constructs, as ENUM NAPTRs match against telephone numbers.

In the normal situation in which E2U NAPTRs are provisioned in ENUM domains, there will be no multi-byte characters within this sub-field, as the ERE will be intended to match against telephone numbers. ENUM clients must be able to handle NAPTRs that do contain such multi-byte characters (as the standard does not preclude them), but there is no operational reason for these ever being provisioned in ENUM domains. If NAPTRs provisioned in ENUM domains are encountered containing such multi-byte characters, these could reasonably be discarded.

The Repl sub-field can include a mixture of explicit text used to construct a URI and characters significant to the substitution expression, as defined in [RFC3403]. Whilst the latter set all fall into the US-ASCII equivalent range of UTF-8 encoding, this might not be the case for all conceivable text used to construct a URI. Presence of multi-byte characters could complicate URI generation and processing routines.

URI generic syntax is defined in [RFC3986] as a sequence of characters chosen from a limited subset of the repertoire of US-ASCII characters. The current URIs use the standard URI character escaping rules specified in the URI generic syntax, and so any multi-byte character will be pre-processed; they will not occur in the explicit text used to construct a URI within the Repl sub-field.

2.1.1.1. Impact of Future Support for IRIs

As currently specified, ENUM only permits URIs to be generated in the Regexp field. However, even if this were to be extended in future revisions of the ENUM specification to allow the use of Internationalised Resource Identifiers (IRIs), defined in [RFC3987], further support for non-ASCII characters may be avoided. IRIs are defined as extending the syntax of URIs, and RFC 3987 specifies a mapping from IRIs to URIs. IRI syntax allows characters with multi-byte UTF-8 encoding.

Given that this is the only place within an ENUM NAPTR where such multi-byte encodings might reasonably be found, a simple solution is to use the mapping method specified in Section 3.1 of [RFC3987] to convert any IRI into its equivalent URI.

This process consists of two elements; the domain part of an IRI MUST be processed using Punycode if it has a non-ASCII domain name, and the remainder MUST be processed using the extended escaping rules specified in [RFC3987] if it contains characters outside the normal URI repertoire. Using this process, there will be no non-ASCII characters in any part of any URI, even if it has been converted from an IRI that contains such characters.

2.1.2. Non-ASCII Support - Conclusions

From the analysis just given, the only place within an ENUM NAPTR where non-ASCII characters might be found is the Regexp field. It is possible to remove any requirement to process characters outside the

US-ASCII equivalent range by adding very few operational restrictions. There is no obvious benefit in providing characters outside this range. Handling multi-byte characters complicates development and operation of client programs, and many existing programs do not include such support.

As the gain from permitting characters outside the US-ASCII equivalent range is unclear, and the costs of multi-byte character processing are very clear, ENUM NAPTRs SHOULD NOT include characters outside the printable US-ASCII equivalent range.

2.2. Case Sensitivity

The only place where NAPTR field content is case sensitive is in any static text in the Repl sub-field of the Regexp field. Everywhere else, case-insensitive processing can be used.

The case-insensitivity flag ('i') could be added at the end of the Regexp field. However, in ENUM, the ERE sub-field operates on a string defined as the '+' character, followed by a sequence of digit characters. This flag is redundant for E2U NAPTRs, as it does not act on the Repl sub-field contents.

Thus, the case-sensitivity flag is inappropriate for ENUM, and SHOULD NOT be provisioned into E2U NAPTRs.

2.3. Regexp Field Delimiter

It is not possible to select a delimiter character that cannot appear in one of the sub-fields. The '!' character is used as a delimiter in all of the examples in [RFC3403] and in [RFC3761]. It is the only character seen in existing zones, and a number of different client implementations are still "hardwired" to expect this character as a delimiter.

The '!' character will not normally appear in the ERE sub-field. It may appear in the content of some URIs, as it is a valid character (e.g., in http URLs). If it is present in the Regexp field, then that instance MUST be escaped using the standard technique proposed in Section 3.2 of [RFC3402]: a backslash character (U+005C) should be inserted before it in the string. Otherwise, a client may attempt to process this as a standard delimiter and interpret the Regexp field contents differently from the system that provisioned it.

2.4. Regexp Meta-Character Issue

In ENUM, the ERE sub-field may include a literal character '+', as the Application Unique String on which it operates includes this. However, if it is present, then '+' MUST be escaped using a single backslash character (to produce the sub-string U+005C U+002B), as '+' is a meta-character in POSIX Extended Regular Expression syntax.

Not escaping the '+' character produces an invalid ERE, but is a common mistake. Even standards have given incorrect examples; the obsolete [RFC2916] (Section 3.4.3, example 3) has this problem.

For example, the following NAPTR example is incorrect:

```
* IN NAPTR 100 10 "u" "E2U+sip" "!^+4655(.*)$!sip:\\1@example.net!" .
```

A correct way to write this example is:

```
* IN NAPTR 100 10 "u"
    "E2U+sip" "!^\\+4655(.*)$!sip:\\1@example.net!" .
```

Note that when a NAPTR resource record is shown in DNS master file syntax (as in this example above), the backslash itself must be escaped using a second backslash. The DNS on-the-wire packet will have only a single backslash.

3. Unsupported NAPTRs

An ENUM client MAY discard a NAPTR received in response to an ENUM query because:

- the NAPTR is syntactically or semantically incorrect,

- the NAPTR has a different (non-empty) DDDS Application identifier from the 'E2U' used in ENUM,

- the NAPTR's ERE does not match the Application Unique String for this ENUM query,

- the ENUM client does not recognise any Enumservice held in this NAPTR, or

- this NAPTR (only) contains an Enumservice that is unsupported.

These conditions SHOULD NOT cause the whole ENUM query to terminate, and processing SHOULD continue with the next NAPTR in the returned Resource Record Set (RRSet).

When an ENUM client encounters a compound NAPTR (i.e., one containing more than one Enumservice -- see also Section 4.4.1) and cannot process or cannot recognise one of the Enumservices within it, that ENUM client SHOULD ignore this Enumservice and continue with the next Enumservice within this NAPTR's Services field, discarding the NAPTR only if it cannot handle any of the Enumservices contained. These conditions SHOULD NOT be considered errors.

ENUM uses regular-expression processing when generating URIs from the Regexp field of "terminal" NAPTRs. Just as with all uses of regular expressions, there is a potential for buffer overrun when generating this output. There may be repeated back-reference patterns in a NAPTR's Repl sub-field, and the output these generate may consume a considerable amount of buffer space.

Even if an ENUM client would normally encounter only NAPTRs with short URIs, it may also receive NAPTRs with repeated back-reference patterns in their Repl sub-fields that could generate strings longer than the client's buffer. Such NAPTRs may have been misconfigured accidentally or by design. The client MUST NOT fail in this case. It SHOULD NOT discard the entire ENUM query, but instead just discard the NAPTR that would otherwise have caused this overrun.

If a problem is detected when processing an ENUM query across multiple domains (by following non-terminal NAPTR references), then the ENUM query SHOULD NOT be abandoned, but instead processing SHOULD continue at the next NAPTR after the non-terminal NAPTR that referred to the domain in which the problem would have occurred. See Section 5.2.2 for more details.

3.1. Non-Compliant Client Behaviour

Through monitoring current ENUM clients, a number of non-compliant behaviours have been detected. These behaviours are incorrect, but may be encountered in still-operational client implementations.

ENUM clients have been known to discard NAPTRs in which the Services field holds more than one Enumservice.

ENUM clients have also been known to discard NAPTRs with a "non-greedy" ERE sub-field expression (i.e., EREs that are dissimilar to "^.*$").

ENUM clients have been known to discard NAPTRs that do not use '!' as their Regexp delimiter character.

ENUM clients have been known to discard NAPTRs in which the delimiter is NOT the last character in the Regexp field.