Network Working Group
Internet-Draft
Intended status: Informational
Expires: September 13, 2012

M. Blanchet
Viagenie
A. Sullivan
Dyn, Inc.

March 12, 2012

Stringprep Revision Problem Statement
draft-ietf-precis-problem-statement-05.txt

Abstract

Using Unicode codepoints in protocol strings that expect comparison with other strings requires preparation of the string that contains the Unicode codepoints. Internationalizing Domain Names in Applications (IDNA2003) defined and used Stringprep and Nameprep. Other protocols subsequently defined Stringprep profiles. A new approach different from Stringprep and Nameprep is used for a revision of IDNA2003 (called IDNA2008). Other Stringprep profiles need to be similarly updated or a replacement of Stringprep needs to be designed. This document outlines the issues to be faced by those designing a Stringprep replacement.

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 http://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 September 13, 2012.

Table of Contents

1. Introduction

Internationalizing Domain Names in Applications (IDNA2003) [RFC3490], [RFC3491], [RFC3492], [RFC3454] describes a mechanism for encoding Unicode labels making up Internationalized Domain Names (IDNs) as standard DNS labels. The labels were processed using a method called Nameprep [RFC3491] and Punycode [RFC3492]. That method was specific to IDNA2003, but is generalized as Stringprep [RFC3454]. The general mechanism is used by other protocols with similar needs, but with different constraints than IDNA2003.

Stringprep defines a framework within which protocols define their Stringprep profiles. Known IETF specifications using Stringprep are listed below:

- The Nameprep profile [RFC3490] for use in Internationalized Domain Names (IDNs);
- NFSv4 [RFC3530] and NFSv4.1 [RFC5661];
- The iSCSI profile [RFC3722] for use in Internet Small Computer Systems Interface (iSCSI) Names;
- EAP [RFC3748];
- The Nodeprep and Resourceprep profiles [RFC3920] for use in the Extensible Messaging and Presence Protocol (XMPP), and the XMPP to CPIM mapping [RFC3922] (the latter of these relies on the former);
- The Policy MIB profile [RFC4011] for use in the Simple Network Management Protocol (SNMP);
- The SASLprep profile [RFC4013] for use in the Simple Authentication and Security Layer (SASL), and SASL itself [RFC4422];
- TLS [RFC4279];
- IMAP4 using SASLprep [RFC4314];
- The trace profile [RFC4505] for use with the SASL ANONYMOUS mechanism;
- The LDAP profile [RFC4518] for use with LDAP [RFC4511] and its authentication methods [RFC4513];
- Plain SASL using SASLprep [RFC4616];
- NNTP using SASLprep [RFC4643];
- PKIX subject identification using LDAPprep [RFC4683];
- Internet Application Protocol Collation Registry [RFC4790];
- SMTP Auth using SASLprep [RFC4954];
- POP3 Auth using SASLprep [RFC5034];
- TLS SRP using SASLprep [RFC5054];
- IRI and URI in XMPP [RFC5122];
- PKIX CRL using LDAPprep [RFC5280];
- IAX using Nameprep [RFC5456];
- SASL SCRAM using SASLprep [RFC5802];
- Remote management of Sieve using SASLprep [RFC5804];

- The unicode-casemap Unicode Collation [RFC5051].

However, a review [1] of these protocol specifications found that they are very similar and can be grouped into a short number of classes. Moreover, many reuse the same Stringprep profile, such as the SASL one.

IDNA2003 was replaced because of some limitations described in [RFC4690]. The new IDN specification, called IDNA2008 [RFC5890], [RFC5891], [RFC5892], [RFC5893] was designed based on the considerations found in [RFC5894]. One of the effects of IDNA2008 is that Nameprep and Stringprep are not used at all. Instead, an algorithm based on Unicode properties of codepoints is defined. That algorithm generates a stable and complete table of the supported Unicode codepoints for each Unicode version. This algorithm is based on an inclusion-based approach, instead of the exclusion-based approach of Stringprep/Nameprep.

This document lists the shortcomings and issues found by protocols listed above that defined Stringprep profiles. It also lists the requirements for any potential replacement of Stringprep.

2. Conventions

This document uses the Unicode convention [2] to specify Unicode codepoint with the following syntax: U+ABCD where ABCD is the codepoint in hexadecimal.

3. Stringprep Profiles Limitations

During IETF 77, a BOF [3] discussed the current state of the protocols that have defined Stringprep profiles [NEWPREP]. The main conclusions from that discussion were as follows:

- Stringprep is bound to version 3.2 of Unicode. Stringprep has not been updated to new versions of Unicode. Therefore, the protocols using Stringprep are stuck to Unicode 3.2.
- The protocols need to be updated to support new versions of Unicode. The protocols would like to not be bound to a specific version of Unicode, but rather have better Unicode agility in the way of IDNA2008. This is important partly because it is usually impossible for an application to require Unicode 3.2; the application gets whatever version of Unicode is available on the host.
- The protocols require better bidirectional support (bidi) than currently offered by Stringprep.

- If the protocols are updated to use a new version of Stringprep or another framework, then backward compatibility is an important requirement. For example, Stringprep is based on and profiles may use NFKC [UAX15], while IDNA2008 mostly uses NFC [UAX15].
- Identifiers are passed between protocols. For example, the same username string of codepoints may be passed between SASL, XMPP, LDAP and EAP. Therefore, common set of rules or classes of strings are preferred over specific rules for each protocol. Without real planning in advance, many stringprep profiles reuse other profiles, so this goal was accomplished by accident with Stringprep.

Protocols that use Stringprep profiles use strings for different purposes:

- XMPP uses a different Stringprep profile for each part of the XMPP address (JID): a localpart which is similar to a username and used for authentication, a domainpart which is a domain name and a resource part which is less restrictive than the localpart.
- iSCSI uses a Stringprep profile for the IQN, which is very similar to (often is) a DNS domain name.
- SASL and LDAP uses a Stringprep profile for usernames.
- LDAP uses a set of Stringprep profiles.

The consensus [4] of the BOF attendees is that it would be highly desirable to have a replacement of Stringprep, with similar characteristics to IDNA2008. That replacement should be defined so that the protocols could use internationalized strings without a lot of specialized internationalization work, since internationalization expertise is not available in the respective protocols or working groups.

4. Major Topics for Consideration

This section provides an overview of major topics that a Stringprep replacement needs to address. The headings correspond roughly with categories under which known Stringprep-using protocol RFCs have been evaluated. For the details of those evaluations, see Appendix A.

4.1. Comparison

4.1.1. Types of Identifiers

Following [I-D.iab-identifier-comparison], it is possible to organize identifiers into three classes in respect of how they may be compared with one another:

Absolute Identifiers Identifiers that can be compared byte-by-byte for equality.

Definite Identifiers Identifiers that have a well-defined comparison algorithm on which all parties agree.

Indefinite Identifiers Identifiers that have no single comparison algorithm on which all parties agree.

Definite Identifiers include cases like the comparison of Unicode code points in different encodings: they do not match byte for byte, but can all be converted to a single encoding which then does match byte for byte. Indefinite Identifiers are sometimes algorithmically comparable by well-specified subsets of parties. For more discussion of these categories, see [I-D.iab-identifier-comparison].

The section on treating the existing known cases, Appendix A uses the categories above.

4.1.2. Effect of comparison

The three classes of comparison style outlined in Section 4.1.1 may have different effects when applied. It is necessary to evaluate the effects if a comparison results in a false positive, and what the effects are if a comparison results in a false negative, especially in terms of the consequences to security and usability.

4.2. Dealing with characters

This section outlines a range of issues having to do with characters in the target protocols, and outlines the ways in which IDNA2008 might be a good analogy to other protocols, and ways in which it might be a poor one.

4.2.1. Case folding, case sensitivity, and case preservation

In IDNA2003, labels are always mapped to lower case before the Punycode transformation. In IDNA2008, there is no mapping at all: input is either a valid U-label or it is not. At the same time, upper-case characters are by definition not valid U-labels, because they fall into the Unstable category (category B) of [RFC5892].

If there are protocols that require upper and lower cases be preserved, then the analogy with IDNA2008 will break down. Accordingly, existing protocols are to be evaluated according to the following criteria:

1. Does the protocol use case folding? For all blocks of code points, or just for certain subsets?

2. Is the system or protocol case sensitive?
3. Does the system or protocol preserve case?

4.2.2. Stringprep and NFKC

Stringprep profiles may use normalization. If they do, they use NFKC [UAX15] (most profiles do). It is not clear that NFKC is the right normalization to use in all cases. In [UAX15], there is the following observation regarding Normalization Forms KC and KD: "It is best to think of these Normalization Forms as being like uppercase or lowercase mappings: useful in certain contexts for identifying core meanings, but also performing modifications to the text that may not always be appropriate." For things like the spelling of users' names, then, NFKC may not be the best form to use. At the same time, one of the nice things about NFKC is that it deals with the width of characters that are otherwise similar, by canonicalizing half-width to full-width. This mapping step can be crucial in practice. A replacement for stringprep depends on analyzing the different use profiles and considering whether NFKC or NFC is a better normalization for each profile.

For the purposes of evaluating an existing example of Stringprep use, it is helpful to know whether it uses no normalization, NFKC, or NFC.

4.2.3. Character mapping

Along with the case mapping issues raised in Section 4.2.1, there is the question of whether some characters are mapped either to other characters or to nothing during Stringprep. [RFC3454], Section 3, outlines a number of characters that are mapped to nothing, and also permits Stringprep profiles to define their own mappings.

4.2.4. Prohibited characters

Along with case folding and other character mappings, many protocols have characters that are simply disallowed. For example, control characters and special characters such as "@" or "/" may be prohibited in a protocol.

One of the primary changes of IDNA2008 is in the way it approaches Unicode code points. IDNA2003 created an explicit list of excluded or mapped-away characters; anything in Unicode 3.2 that was not so listed could be assumed to be allowed under the protocol. IDNA2008 begins instead from the assumption that code points are disallowed, and then relies on Unicode properties to derive whether a given code point actually is allowed in the protocol.

Moreover, there is more than one class of "allowed in the protocol"

in IDNA2008 (but not in IDNA2003). While some code points are disallowed outright, some are allowed only in certain contexts. The reasons for the context-dependent rules have to do with the way some characters are used. For instance, the ZERO WIDTH JOINER and ZERO WIDTH NON-JOINER (ZWJ, U+200D and ZWNJ, U+200C) are allowed with contextual rules because they are required in some circumstances, yet are considered punctuation by Unicode and would therefore be DISALLOWED under the usual IDNA2008 derivation rules. The goal of IDNA2008 is to provide the widest repertoire of code points possible and consistent with the traditional DNS LDH rule, trusting to the operators of individual zones to make sensible (and usually more restrictive) policies for their zones.

IDNA2008 may be a poor model for what other protocols ought to do in this case, because it is designed to support an old protocol that is designed to operate on the scale of the entire Internet. Moreover, IDNA2008 is intended to be deployed without any change to the base DNS protocol. Other protocols may aim at deployment in more local environments, or may have protocol version negotiation built in.

4.2.5. Internal structure, delimiters, and special characters

IDNA2008 has a special problem with delimiters, because the delimiter "character" in the DNS wire format is not really part of the data. In DNS, labels are not separated exactly; instead, a label carries with it an indicator that says how long the label is. When the label is presented in presentation format as part of a fully qualified domain name, the label separator FULL STOP, U+002E (.) is used to break up the labels. But because that label separator does not travel with the wire format of the domain name, there is no way to encode a different, "internationalized" separator in IDNA2008.

Other protocols may include characters with similar special meaning within the protocol. Common characters for these purposes include FULL STOP, U+002E (.); COMMERCIAL AT, U+0040 (@); HYPHEN-MINUS, U+002D (-); SOLIDUS, U+002F (/); and LOW LINE, U+005F (_). The mere inclusion of such a character in the protocol is not enough for it to be considered similar to another protocol using the same character; instead, handling of the character must be taken into consideration as well.

An important issue to tackle here is whether it is valuable to map to or from these special characters as part of the Stringprep replacement. In some locales, the analogue to FULL STOP, U+002E is some other character, and users may expect to be able to substitute their normal stop for FULL STOP, U+002E. At the same time, there are predictability arguments in favour of treating identifiers with FULL STOP, U+002E in them just the way they are treated under IDNA2008.

4.2.6. Restrictions because of glyph similarity

Homoglyphs are similarly (or identically) rendered glyphs of different codepoints. For DNS names, homoglyphs may enable phishing. If a protocol requires some visual comparison by end-users, then the issue of homoglyphs are to be considered. In the DNS context, theses issues are documented in [RFC5894] and [RFC4690]. IDNA2008 does not, however, have a mechanism to deal with them, trusting to DNS zone operators to enact sensible policies for the subset of Unicode they wish to support, given their user community. A similar policy/ protocol split may not be desirable in every protocol.

4.3. Where the data comes from and where it goes

4.3.1. User input and the source of protocol elements

Some protocol elements are provided by users, and others are not. Those that are not may presumably be subject to greater restrictions, whereas those that users provide likely need to permit the broadest range of code points. The following questions are helpful:

1. Do users input the strings directly?
2. If so, how? (keyboard, stylus, voice, copy-paste, etc.)
3. Where do we place the dividing line between user interface and protocol? (see [RFC5895])

4.3.2. User output

Just as only some protocol elements are expected to be entered directly by users, only some protocol elements are intended to be consumed directly by users. It is important to know how users are expected to be able to consume the protocol elements, because different environments present different challenges. An element that is only ever delivered as part of a vCard remains in machine-readable format, so the problem of visual confusion is not a great one. Is the protocol element published as part of a vCard, a web directory, on a business card, or on "the side of a bus"? Do users use the protocol element as an identifier (which means that they might enter it again in some other context)? (See also Section 4.2.6.)

4.3.3. Operations

Some strings are useful as part of the protocol but are not used as input to other operations (for instance, purely informative or descriptive text). Other strings are used directly as input to other operations (such as cryptographic hash functions), or are used together with other strings to (such as concatenating a string with some others to form a unique identifier).

4.3.3.1. String classes

Strings often have a similar function in different protocols. For instance, many different protocols contain user identifiers or passwords. A single profile for all such uses might be desirable.

Often, a string in a protocol is effectively a protocol element from another protocol. For instance, different systems might use the same credentials database for authentication.

4.3.3.2. Community Considerations

A Stringprep replacement that does anything more than just update Stringprep to the latest version of Unicode will probably entail some changes. It is important to identify the willingness of the protocol-using community to accept backwards-incompatible changes. By the same token, it is important to evaluate the desire of the community for features not available under Stringprep.

4.3.3.3. Unicode Incompatible Changes

IDNA2008 uses an algorithm to derive the validity of a Unicode code point for use under IDNA2008. It does this by using the properties of each code point to test its validity.

This approach depends crucially on the idea that code points, once valid for a protocol profile, will not later be made invalid. That is not a guarantee currently provided by Unicode. Properties of code points may change between versions of Unicode. Rarely, such a change could cause a given code point to become invalid under a protocol profile, even though the code point would be valid with an earlier version of Unicode. This is not merely a theoretical possibility, because it has occurred ([RFC6452]).

Accordingly, as IDNA2008,a Stringprep replacement that intends to be Unicode version agnostic will need to work out a mechanism to address cases where incompatible changes occur because of new Unicode versions.

4.3.4. Some useful classes of strings

With the above considerations in hand, we can usefully classify strings into the following categories:

DomainClass Strings that are intended for use in a domain name slot, as defined in [RFC5890]. Note that strings of DomainClass could be used outside a domain name slot: the question here is what the eventual intended use for the string is, and not whether the string is actually functioning as a domain name at any moment.

NameClass Strings that are intended for use as identifiers but that are not DomainClass strings. NameClass strings are normally public data within the protocol where they are used: these are intended as identifiers that can be passed around to identify something.

FreeClass Strings that are intended to be used by the protocol as free-form strings, but that have some significant handling within the protocol. This includes things that are normally not public data in a protocol (like passwords), and things that might have additional restrictions within the protocol in question, such as a friendly name in a chat room.

5. Considerations for Stringprep replacement

The above suggests the following guidance for replacing Stringprep:

- A stringprep replacement should be defined.
- The replacement should take an approach similar to IDNA2008, (e.g. by using codepoint properties instead of codepoint whitelisting) in that it enables better Unicode agility.
- Protocols share similar characteristics of strings. Therefore, defining i18n preparation algorithms for the smallest set of string classes may be sufficient for most cases, providing coherence among a set of related protocols or protocols where identifiers are exchanged.
- The sets of string classes need to be evaluated according to the considerations that make up the headings in Section 4
- It is reasonable to limit scope to Unicode code points, and rule the mapping of data from other character encodings outside the scope of this effort.
- Recommendations for handling protocol incompatibilities resulting from changes to Unicode are required.
- Comptability within each protocol between a technique that is stringprep-based and the technique's replacement has to be considered very carefully.

Existing deployments already depend on Stringprep profiles. Therefore, a replacement must consider the effects of any new strategy on existing deployments. By way of comparison, it is worth noting that some characters were acceptable in IDNA labels under IDNA2003, but are not protocol-valid under IDNA2008 (and conversely); disagreement about what to do during the transition has resulted in different approaches to mapping. Different implementers may make

different decisions about what to do in such cases; this could have interoperability effects. It is necessary to trade better support for different linguistic environments against the potential side effects of backward incompatibility.

6. Security Considerations

This document merely states what problems are to be solved, and does not define a protocol. There are undoubtedly security implications of the particular results that will come from the work to be completed.

7. IANA Considerations

This document has no actions for IANA.

8. Discussion home for this draft

Note: RFC-Editor, please remove this section before publication.

This document is intended to define the problem space discussed on the precis@ietf.org mailing list.

9. Acknowledgements

This document is the product of the PRECIS IETF Working Group, and participants in that Working Group were helpful in addressing issues with the text.

Specific contributions came from David Black, Alan DeKok, Bill McQuillan, Alexey Melnikov, Peter Saint-Andre, Dave Thaler, and Yoshiro Yoneya.

Dave Thaler provided the "buckets" insight in Section 4.1.1, central to the organization of the problem.

Evaluations of Stringprep profiles that are included in Appendix B were done by: David Black, Alexey Melnikov, Peter Saint-Andre, Dave Thaler.

10. Informative References

[I-D.iab-identifier-comparison]

Thaler, D., "Issues in Identifier Comparison for Security Purposes", draft-iab-identifier-comparison-00 (work in progress), July 2011.

[NEWPREP] "Newprep BoF Meeting Minutes", March 2010.

[RFC3454] Hoffman, P. and M. Blanchet, "Preparation of Internationalized Strings ("stringprep")", RFC 3454, December 2002.

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

[RFC3491] Hoffman, P. and M. Blanchet, "Nameprep: A Stringprep Profile for Internationalized Domain Names (IDN)", RFC 3491, March 2003.

[RFC3492] Costello, A., "Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)", RFC 3492, March 2003.

[RFC3530] Shepler, S., Callaghan, B., Robinson, D., Thurlow, R., Beame, C., Eisler, M., and D. Noveck, "Network File System (NFS) version 4 Protocol", RFC 3530, April 2003.

[RFC3722] Bakke, M., "String Profile for Internet Small Computer Systems Interface (iSCSI) Names", RFC 3722, April 2004.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, "Extensible Authentication Protocol (EAP)", RFC 3748, June 2004.

[RFC3920] Saint-Andre, P., Ed., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 3920, October 2004.

[RFC3922] Saint-Andre, P., "Mapping the Extensible Messaging and Presence Protocol (XMPP) to Common Presence and Instant Messaging (CPIM)", RFC 3922, October 2004.

[RFC4011] Waldbusser, S., Saperia, J., and T. Hongal, "Policy Based Management MIB", RFC 4011, March 2005.

[RFC4013] Zeilenga, K., "SASLprep: Stringprep Profile for User Names and Passwords", RFC 4013, February 2005.

[RFC4279] Eronen, P. and H. Tschofenig, "Pre-Shared Key Ciphersuites for Transport Layer Security (TLS)", RFC 4279,

December 2005.

[RFC4314] Melnikov, A., "IMAP4 Access Control List (ACL) Extension", RFC 4314, December 2005.

[RFC4422] Melnikov, A. and K. Zeilenga, "Simple Authentication and Security Layer (SASL)", RFC 4422, June 2006.

[RFC4505] Zeilenga, K., "Anonymous Simple Authentication and Security Layer (SASL) Mechanism", RFC 4505, June 2006.

[RFC4511] Sermersheim, J., "Lightweight Directory Access Protocol (LDAP): The Protocol", RFC 4511, June 2006.

[RFC4513] Harrison, R., "Lightweight Directory Access Protocol (LDAP): Authentication Methods and Security Mechanisms", RFC 4513, June 2006.

[RFC4518] Zeilenga, K., "Lightweight Directory Access Protocol (LDAP): Internationalized String Preparation", RFC 4518, June 2006.

[RFC4616] Zeilenga, K., "The PLAIN Simple Authentication and Security Layer (SASL) Mechanism", RFC 4616, August 2006.

[RFC4643] Vinocur, J. and K. Murchison, "Network News Transfer Protocol (NNTP) Extension for Authentication", RFC 4643, October 2006.

[RFC4683] Park, J., Lee, J., Lee, H., Park, S., and T. Polk, "Internet X.509 Public Key Infrastructure Subject Identification Method (SIM)", RFC 4683, October 2006.

[RFC4690] Klensin, J., Faltstrom, P., Karp, C., and IAB, "Review and Recommendations for Internationalized Domain Names (IDNs)", RFC 4690, September 2006.

[RFC4790] Newman, C., Duerst, M., and A. Gulbrandsen, "Internet Application Protocol Collation Registry", RFC 4790, March 2007.

[RFC4954] Siemborski, R. and A. Melnikov, "SMTP Service Extension for Authentication", RFC 4954, July 2007.

[RFC5034] Siemborski, R. and A. Menon-Sen, "The Post Office Protocol (POP3) Simple Authentication and Security Layer (SASL) Authentication Mechanism", RFC 5034, July 2007.

[RFC5051] Crispin, M., "i;unicode-casemap - Simple Unicode Collation Algorithm", RFC 5051, October 2007.

[RFC5054] Taylor, D., Wu, T., Mavrogiannopoulos, N., and T. Perrin, "Using the Secure Remote Password (SRP) Protocol for TLS Authentication", RFC 5054, November 2007.

[RFC5122] Saint-Andre, P., "Internationalized Resource Identifiers (IRIs) and Uniform Resource Identifiers (URIs) for the Extensible Messaging and Presence Protocol (XMPP)", RFC 5122, February 2008.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, May 2008.

[RFC5456] Spencer, M., Capouch, B., Guy, E., Miller, F., and K. Shumard, "IAX: Inter-Asterisk eXchange Version 2", RFC 5456, February 2010.

[RFC5661] Shepler, S., Eisler, M., and D. Noveck, "Network File System (NFS) Version 4 Minor Version 1 Protocol", RFC 5661, January 2010.

[RFC5802] Newman, C., Menon-Sen, A., Melnikov, A., and N. Williams, "Salted Challenge Response Authentication Mechanism (SCRAM) SASL and GSS-API Mechanisms", RFC 5802, July 2010.

[RFC5804] Melnikov, A. and T. Martin, "A Protocol for Remotely Managing Sieve Scripts", RFC 5804, July 2010.

[RFC5890] Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, August 2010.

[RFC5891] Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, August 2010.

[RFC5892] Faltstrom, P., "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)", RFC 5892, August 2010.

[RFC5893] Alvestrand, H. and C. Karp, "Right-to-Left Scripts for Internationalized Domain Names for Applications (IDNA)", RFC 5893, August 2010.

[RFC5894] Klensin, J., "Internationalized Domain Names for

Applications (IDNA): Background, Explanation, and Rationale", RFC 5894, August 2010.

[RFC5895] Resnick, P. and P. Hoffman, "Mapping Characters for Internationalized Domain Names in Applications (IDNA) 2008", RFC 5895, September 2010.

[RFC6452] Faltstrom, P. and P. Hoffman, "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA) - Unicode 6.0", RFC 6452, November 2011.

[UAX15] "Unicode Standard Annex #15: Unicode Normalization Forms", UAX 15, September 2009.

[1] <http://www.ietf.org/proceedings/78/slides/precis-2.pdf>

[2] <http://www.unicode.org/standard/principles.html>

[3] <http://www.ietf.org/proceedings/77/newprep.html>

[4] <http://www.ietf.org/proceedings/77/minutes/newprep.txt>

[5] <http://trac.tools.ietf.org/wg/precis/trac/report/6>

Appendix A. Classification of Stringprep Profiles

A number of the known cases of Stringprep use were evaluated during the preparation of this document. The known cases are here described in two ways. The types of identifiers the protocol uses is first called out in the ID type column (from Section 4.1.1), using the short forms "a" for Absolute, "d" for Definite, and "i" for Indefinite. Next, there is a column that contains an "i" if the protocol string comes from user input, an "o" if the protocol string becomes user-facing output, "b" if both are true, and "n" if neither is true. The remaining columns have an "x" if and only if the protocol uses that class, as described in Section 4.3.4. Values marked "-" indicate that an answer is not useful; in this case, see detailed discussion in Appendix B.

RFC	IDtype	User?	DomainClass	NameClass	FreeClass
3722	a	o		x	x
3748	-	-	-	x	-
3920	a,d	b		x	x
4505	a	i			x
4314	a,d	b		x	x
4954	a,d	b		x	
5034	a,d	b		x	
5804	a,d	b		x	

Table 1

[[anchor22: This table now contains results of any reviews the WG did. Unreviewed things in the tracker are not reflected here. --ajs@anvilwalrusden.com]]

Appendix B. Evaluation of Stringprep Profiles

This section is a summary of the evaluation of Stringprep profiles [5] that was done to get a good understanding of the usage of Stringprep. This summary is by no means normative nor the actual evaluations themselves. A template was used for reviewers to get a coherent view of all evaluations.

B.1. iSCSI Stringprep Profiles: RFC3722, RFC3721, RFC3720

Description: An iSCSI session consists of an Initiator (i.e., host or server that uses storage) communicating with a target (i.e., a storage array or other system that provides storage). Both the iSCSI initiator and target are named by iSCSI Names. The iSCSI stringprep profile is used for iSCSI names.

How it is used iSCSI initiators and targets (see above). They can also be used to identify SCSI ports (these are software entities in the iSCSI protocol, not hardware ports), and iSCSI logical units (storage volumes), although both are unusual in practice.

What entities create these identifiers? Generally a Human user (1) configures an Automated system (2) that generates the names. Advance configuration of the system is required due to the embedded use of external unique identifier (from the DNS or IEEE).

How is the string input in the system? Keyboard and copy-paste are common. Copy-paste is common because iSCSI names are long enough to be problematic for humans to remember, causing use of email, sneaker-net, text files, etc. to avoid mistype mistakes.

Where do we place the dividing line between user interface and protocol? The iSCSI protocol requires that all i18n string preparation occur in the user interface. The iSCSI protocol treats iSCSI names as opaque identifiers that are compared byte-by-byte for equality. iSCSI names are generally not checked for correct formatting by the protocol.

What entities enforce the rules? There are no iSCSI-specific enforcement entities, although the use of unique identifier information in the names relies on DNS registrars and the IEEE Registration Authority.

Comparison Byte-by-byte

Case Folding, Sensitivity, Preservation Case folding is required for the code blocks specified in RFC 3454, Table B.2. The overall iSCSI naming system (UI + protocol) is case-insensitive.

What is the impact if the comparison results in a false positive? Potential access to the wrong storage. - If the initiator has no access to the wrong storage, an authentication failure is the probable result. - If the initiator has access to the worng storage, the resulting mis-identificaiton could result in use of the wrong data and possible corruption of stored data.

What is the impact if the comparison results in a false negative? Denial of authorized storage access.

What are the security impacts? iSCSI names are often used as the authentication identities for storage systems. Comparison problems could result in authentication problems, although note that authentication failure ameliorates some of the false positive cases.

Normalization NFKC, as specified by RFC 3454.

Mapping Yes, as specified by table B.1 in RFC 3454

Disallowed Characters Only the following characters are allowed: - ASCII dash, dot, colon - ASCII lower case letters and digits - Unicode lower case characters as specified by RFC 3454 All other characters are disallowed.

Which other strings or identifiers are these most similar to? None - iSCSI names are unique to iSCSI.

Are these strings or identifiers sometimes the same as strings or identifiers from other protocols? No

Does the identifier have internal structure that needs to be respected? Yes - ASCII dot, dash and colon are used for internal name structure. These are not reserved characters in that they can occur in the name in locations other than those used for structuring purposes (e.g., only the first occurrence of a colon character is structural, others are not).

How are users exposed to these strings? How are they published? iSCSI names appear in server and storage system configuration interfaces. They also appear in system logs.

Is the string / identifier used as input to other operations?
Effectively, no. The rarely used port and logical unit names involve concatenation, which effectively extends a unique iSCSI Name for a target to uniquely identify something within that target.

How much tolerance for change from existing stringprep approach?
Good tolerance; the community would prefer that i18n experts solve i18n problems ;-).

How strong a desire for change (e.g., for Unicode agility)? Unicode agility is desired in principle as long as nothing significant breaks.

B.2. SMTP/POP3/ManageSieve Stringprep Profiles: RFC4954,RFC5034,RFC 5804

Description: Authorization identity (user identifier) exchanged during SASL authentication: AUTH (SMTP/POP3) or AUTHENTICATE (ManageSieve) command.

How It's Used: Used for proxy authorization, e.g. to [lawfully] impersonate a particular user after a privileged authentication

Who Generates It: Typically generated by email system administrators using some tools/conventions, sometimes from some backend database. - In some setups human users can register own usernames (e.g. webmail self registration)

User Input Methods: - Typed by user / selected from a list - Copy-and-paste - Perhaps voice input - Can also be specified in configuration files or on a command line

Enforcement: - Rules enforced by server / add-on service (e.g., gateway service) on registration of account

Comparison Method: "Type 1" (byte-for-byte) or "type 2" (compare by a common algorithm that everyone agrees on (e.g., normalize and then compare the result byte-by-byte))

Case Folding, Sensitivity, Preservation: Most likely case sensitive. Exact requirements on case-sensitivity/case-preservation depend on a specific implementation, e.g. an implementation might treat all user identifiers as case insensitive (or case insensitive for US-ASCII subset only).

Impact of Comparison: False positives: - an unauthorized user is allowed email service access (login) False negatives: - an authorized user is denied email service access

Normalization: NFKC (as per RFC 4013)

Mapping: (see Section 2 of RFC 4013 for the full list): Non ASCII spaces are mapped to space, etc.

Disallowed Characters: (see Section 2 of RFC 4013 for the full list): Unicode Control characters, etc.

String Classes: - simple username. See Section 2 of RFC 4013 for details on restrictions. Note that some implementations allow spaces in these. While implementations are not required to use a specific format, an authorization identity frequently has the same format as an email address (and EAI email address in the future), or as a left hand side of an email address. Note: whatever is recommended for SMTP/POP/ManageSieve authorization identity should also be used for IMAP authorization identities, as IMAP/POP3/SMTP/ManageSieve are frequently implemented together.

Internal Structure: None

User Output: Unlikely, but possible. For example, if it is the same as an email address.

Operations: - Sometimes concatenated with other data and then used as input to a cryptographic hash function

How much tolerance for change from existing stringprep approach? Not sure.

Background information: In RFC 5034, when describing the POP3 AUTH command: The authorization identity generated by the SASL exchange is a simple username, and SHOULD use the SASLprep profile (see [RFC4013]) of the StringPrep algorithm (see [RFC3454]) to prepare these names for matching. If preparation of the authorization identity fails or results in an empty string (unless it was transmitted as the empty string), the server MUST fail the authentication. In RFC 4954, when describing the SMTP AUTH command: The authorization identity generated by this [SASL] exchange is a "simple username" (in the sense defined in [SASLprep]), and both client and server SHOULD (*) use the [SASLprep] profile of the [StringPrep] algorithm to prepare these names for transmission or comparison. If preparation of the authorization identity fails or results in an empty string (unless it was transmitted as the empty string), the server MUST fail the authentication. (*) Note: Future revision of this specification may change this requirement to MUST. Currently, the SHOULD is used in order to avoid breaking the majority of existing implementations. In RFC 5804, when describing the ManageSieve AUTHENTICATE command: The authorization identity generated by this [SASL] exchange is a "simple username" (in the sense defined in [SASLprep]), and both client and server MUST use the [SASLprep] profile of the [StringPrep] algorithm to prepare these names for transmission or comparison. If preparation of the authorization identity fails or results in an empty string (unless it was transmitted as the empty string), the server MUST fail the authentication.

B.3. IMAP Stringprep Profiles: RFC5738, RFC4314: Usernames

Evaluation Note These documents have 2 types of strings (usernames and passwords), so there are two separate templates.

Description: "username" parameter to the IMAP LOGIN command, identifiers in IMAP ACL commands. Note that any valid username is also an IMAP ACL identifier, but IMAP ACL identifiers can include other things like name of group of users.

How It's Used: Used for authentication (Usernames), or in IMAP Access Control Lists (Usernames or Group names)

Who Generates It: - Typically generated by email system administrators using some tools/conventions, sometimes from some backend database. - In some setups human users can register own usernames (e.g. webmail self registration)

User Input Methods: - Typed by user / selected from a list - Copy-and-paste - Perhaps voice input - Can also be specified in configuration files or on a command line

Enforcement: - Rules enforced by server / add-on service (e.g., gateway service) on registration of account

Comparison Method: Type 1" (byte-for-byte) or "type 2" (compare by a common algorithm that everyone agrees on (e.g., normalize and then compare the result byte-by-byte))

Case Folding, Sensitivity, Preservation: - Most likely case sensitive. Exact requirements on case-sensitivity/ case-preservation depend on a specific implementation, e.g. an implementation might treat all user identifiers as case insensitive (or case insensitive for US-ASCII subset only).

Impact of Comparison: False positives: - an unauthorized user is allowed IMAP access (login) - improperly grant privileges (e.g., access to a specific mailbox, ability to manage ACLs for a mailbox) False negatives: - an authorized user is denied IMAP access - unable to use granted privileges (e.g., access to a specific mailbox, ability to manage ACLs for a mailbox)

Normalization: NFKC (as per RFC 4013)

Mapping: (see Section 2 of RFC 4013 for the full list): non ASCII spaces are mapped to space

Disallowed Characters: (see Section 2 of RFC 4013 for the full list): Unicode Control characters, etc.

String Classes: - simple username. See Section 2 of RFC 4013 for details on restrictions. Note that some implementations allow spaces in these. While IMAP implementations are not required to use a specific format, an IMAP username frequently has the same format as an email address (and EAI email address in the future), or as a left hand side of an email address. Note: whatever is recommended for IMAP username should also be used for ManageSieve, POP3 and SMTP authorization identities, as IMAP/POP3/SMTP/ ManageSieve are frequently implemented together.

Internal Structure: None
User Output: Unlikely, but possible. For example, if it is the same as an email address. - access control lists (e.g. in IMAP ACL extension), both when managing membership and listing membership of existing access control lists. - often show up as mailbox names (under Other Users IMAP namespace)
Operations: - Sometimes concatenated with other data and then used as input to a cryptographic hash function
How much tolerance for change from existing stringprep approach? Not sure. Non-ASCII IMAP usernames are currently prohibited by IMAP (RFC 3501). However they are allowed when used in IMAP ACL extension.

B.4. IMAP Stringprep Profiles: RFC5738: Passwords

Description: "Password" parameter to the IMAP LOGIN command
How It's Used: Used for authentication (Passwords)
Who Generates It: Either generated by email system administrators using some tools/conventions, or specified by the human user.
User Input Methods: - Typed by user - Copy-and-paste - Perhaps voice input - Can also be specified in configuration files or on a command line
Enforcement: Rules enforced by server / add-on service (e.g., gateway service or backend databse) on registration of account
Comparison Method: "Type 1" (byte-for-byte)
Case Folding, Sensitivity, Preservation: Most likely case sensitive.
Impact of Comparison: False positives: - an unauthorized user is allowed IMAP access (login) False negatives: - an authorized user is denied IMAP access
Normalization: NFKC (as per RFC 4013)
Mapping: (see Section 2 of RFC 4013 for the full list): non ASCII spaces are mapped to space
Disallowed Characters: (see Section 2 of RFC 4013 for the full list): Unicode Control characters, etc.
String Classes: Currently defined as "simple username" (see Section 2 of RFC 4013 for details on restrictions.), however this is likely to be a different class from usernames. Note that some implementations allow spaces in these. Password in all email related protocols should be treated in the same way. Same passwords are frequently shared with web, IM, etc. applications.
Internal Structure: None
User Output: - text of email messages (e.g. in "you forgot your password" email messages) - web page / directory - side of the bus / in ads -- possible

Operations: Sometimes concatenated with other data and then used as input to a cryptographic hash function. Frequently stored as is, or hashed.

How much tolerance for change from existing stringprep approach? Not sure. Non-ASCII IMAP passwords are currently prohibited by IMAP (RFC 3501), however they are likely to be in widespread use.

Background information: RFC 5738 (IMAP I18N): 5. UTF8=USER Capability If the "UTF8=USER" capability is advertised, that indicates the server accepts UTF-8 user names and passwords and applies SASLprep [RFC4013] to both arguments of the LOGIN command. The server MUST reject UTF-8 that fails to comply with the formal syntax in RFC 3629 [RFC3629] or if it encounters Unicode characters listed in Section 2.3 of SASLprep RFC 4013 [RFC4013]. RFC 4314 (IMAP4 Access Control List (ACL) Extension): 3. Access control management commands and responses Servers, when processing a command that has an identifier as a parameter (i.e., any of SETACL, DELETEACL, and LISTRIGHTS commands), SHOULD first prepare the received identifier using "SASLprep" profile [SASLprep] of the "stringprep" algorithm [Stringprep]. If the preparation of the identifier fails or results in an empty string, the server MUST refuse to perform the command with a BAD response. Note that Section 6 recommends additional identifier's verification steps. and in Section 6: This document relies on [SASLprep] to describe steps required to perform identifier canonicalization (preparation). The preparation algorithm in SASLprep was specifically designed such that its output is canonical, and it is well-formed. However, due to an anomaly [PR29] in the specification of Unicode normalization, canonical equivalence is not guaranteed for a select few character sequences. Identifiers prepared with SASLprep can be stored and returned by an ACL server. The anomaly affects ACL manipulation and evaluation of identifiers containing the selected character sequences. These sequences, however, do not appear in well-formed text. In order to address this problem, an ACL server MAY reject identifiers containing sequences described in [PR29] by sending the tagged BAD response. This is in addition to the requirement to reject identifiers that fail SASLprep preparation as described in Section 3.

B.5. Anonymous SASL Stringprep Profiles: RFC4505

Description: RFC 4505 defines a "trace" field:

Comparison: this field is not intended for comparison (only used for logging)

Case folding; case sensitivity, preserve case: No case folding/case sensitive

Do users input the strings directly? Yes. Possibly entered in configuration UIs, or on a command line. Can also be stored in configuration files. The value can also be automatically generated by clients (e.g. a fixed string is used, or a user's email address).

How users input strings? Keyboard/voice, stylus (pick from a list). Copy-paste - possibly.

Normalization: None

Disallowed Characters Control characters are disallowed. (See Section 3 of RFC 4505)

Which other strings or identifiers are these most similar to? RFC 4505 says that the trace "should take one of two forms: an Internet email address, or an opaque string that does not contain the '@' U+0040) character and that can be interpreted by the system administrator of the client's domain." In practice, this is a freeform text, so it belongs to a different class from "email address" or "username".

Are these strings or identifiers sometimes the same as strings or identifiers from other protocols (e.g., does an IM system sometimes use the same credentials database for authentication as an email system)? Yes: see above. However there is no strong need to keep them consistent in the future.

How are users exposed to these strings, how are they published? No. However, The value can be seen in server logs

Impacts of false positives and false negatives: False positive: a user can be confused with another user. False negative: two distinct users are treated as the same user. But note that the trace field is not authenticated, so it can be easily falsified.

Tolerance of changes in the community The community would be flexible.

Delimiters No internal structure, but see comments above about frequent use of email addresses.

Background information: The Anonymous Mechanism The mechanism consists of a single message from the client to the server. The client may include in this message trace information in the form of a string of [UTF-8]-encoded [Unicode] characters prepared in accordance with [StringPrep] and the "trace" stringprep profile defined in Section 3 of this document. The trace information, which has no semantical value, should take one of two forms: an Internet email address, or an opaque string that does not contain the '@' (U+0040) character and that can be interpreted by the system administrator of the client's domain. For privacy reasons, an Internet email address or other information identifying the user should only be used with permission from the user. 3. The "trace" Profile of "Stringprep" This section defines the "trace" profile of [StringPrep]. This profile is designed for use with

the SASL ANONYMOUS Mechanism. Specifically, the client is to prepare the message production in accordance with this profile. The character repertoire of this profile is Unicode 3.2 [Unicode]. No mapping is required by this profile. No Unicode normalization is required by this profile. The list of unassigned code points for this profile is that provided in Appendix A of [StringPrep]. Unassigned code points are not prohibited. Characters from the following tables of [StringPrep] are prohibited: - C.2.1 (ASCII control characters) - C.2.2 (Non-ASCII control characters) - C.3 (Private use characters) - C.4 (Non-character code points) - C.5 (Surrogate codes) - C.6 (Inappropriate for plain text) - C.8 (Change display properties are deprecated) - C.9 (Tagging characters) No additional characters are prohibited. This profile requires bidirectional character checking per Section 6 of [StringPrep].

B.6. XMPP Stringprep Profiles: RFC3920 Nodeprep

Description: Localpart of JabberID ("JID"), as in: localpart@domainpart/resourcepart
How It's Used: - Usernames (e.g., stpeter@jabber.org) - Chatroom names (e.g., precis@jabber.ietf.org) - Publish-subscribe nodes - Bot names
Who Generates It: - Typically, end users via an XMPP client - Sometimes created in an automated fashion
User Input Methods: - Typed by user - Copy-and-paste - Perhaps voice input - Clicking a URI/IRI
Enforcement: - Rules enforced by server / add-on service (e.g., chatroom service) on registration of account, creation of room, etc.
Comparison Method: "Type 2" (common algorithm)
Case Folding, Sensitivity, Preservation: - Strings are always folded to lowercase - Case is not preserved
Impact of Comparison: False positives: - unable to authenticate at server (or authenticate to wrong account) - add wrong person to buddy list - join the wrong chatroom - improperly grant privileges (e.g., chatroom admin) - subscribe to wrong pubsub node - interact with wrong bot - allow communication with blocked entity False negatives: - unable to authenticate - unable to add someone to buddy list - unable to join desired chatroom - unable to use granted privileges (e.g., chatroom admin) - unable to subscribe to desired pubsub node - unable to interact with desired bot - disallow communication with unblocked entity
Normalization: NFKC

Mapping: Spaces are mapped to nothing
Disallowed Characters: ",&,',/,:,<,>,@
String Classes: - Often similar to generic username - Often similar to localpart of email address - Sometimes same as localpart of email address
Internal Structure: None
User Output: - vCard - email signature - web page / directory - text of message (e.g., in a chatroom)
Operations: - Sometimes concatenated with other data and then used as input to a cryptographic hash function

B.7. XMPP Stringprep Profiles: RFC3920 Resourceprep

Description: - Resourcepart of JabberID ("JID"), as in: localpart@domainpart/resourcepart - Typically free-form text
How It's Used: - Device / session names (e.g., stpeter@jabber.org/Home) - Nicknames (e.g., precis@jabber.ietf.org/StPeter)
Who Generates It: - Often human users via an XMPP client - Often generated in an automated fashion by client or server
User Input Methods: - Typed by user - Copy-and-paste - Perhaps voice input - Clicking a URI/IRI
Enforcement: - Rules enforced by server / add-on service (e.g., chatroom service) on account login, joining a chatroom, etc.
Comparison Method: "Type 2" (byte-for-byte)
Case Folding, Sensitivity, Preservation: - Strings are never folded - Case is preserved
Impact of Comparison: False positives: - interact with wrong device (e.g., for file transfer or voice call) - interact with wrong chatroom participant - improperly grant privileges (e.g., chatroom moderator) - allow communication with blocked entity False negatives: - unable to choose desired chatroom nick - unable to use granted privileges (e.g., chatroom moderator) - disallow communication with unblocked entity
Normalization: NFKC
Mapping: Spaces are mapped to nothing
Disallowed Characters: None
String Classes: Basically a free-form identifier
Internal Structure: None
User Output: - text of message (e.g., in a chatroom) - device names often not exposed to human users
Operations: Sometimes concatenated with other data and then used as input to a cryptographic hash function

B.8. EAP Stringprep Profiles: RFC3748

Description: RFC 3748 section 5 references Stringprep, but the WG did not agree with the text (was added by IESG) and there are no known implementations that use Stringprep. The main problem with that text is that the use of strings is a per-method concept, not a generic EAP concept and so RFC 3748 itself does not really use Stringprep, but individual EAP methods could. As such, the answers to the template questions are mostly not applicable, but a few answers are universal across methods. The list of IANA registered EAP methods is at http://www.iana.org/assignments/eap-numbers/eap-numbers.xml#eap-numbers-3

Comparison Methods: n/a (per-method)

Case Folding, Case Sensitivity, Case Preservation: n/a (per-method)

Impact of comparison: A false positive results in unauthorized network access (and possibly theft of service if some else is billed). A false negative results in lack of authorized network access (no connectivity).

User input: n/a (per-method)

Normalization: n/a (per-method)

Mapping: n/a (per-method)

Disallowed characters: n/a (per-method)

String classes: Although some EAP methods may use a syntax similar to other types of identifiers, EAP mandates that the actual values must not be assumed to be identifiers usable with anything else.

Internal structure: n/a (per-method)

User output: Identifiers are never human displayed except perhaps as they're typed by a human.

Operations: n/a (per-method)

Community considerations: There is no resistance to change for the base EAP protocol (as noted, the WG didn't want the existing text). However actual use of stringprep, if any, within specific EAP methods may have resistance. It is currently unknown whether any EAP methods use stringprep.

Appendix C. Changes between versions

Note to RFC Editor: This section should be removed prior to publication.

C.1. 00

First WG version. Based on draft-blanchet-precis-problem-statement-00.

C.2. 01

- Made clear that the document is talking only about Unicode code points, and not any particular encoding.
- Substantially reorganized the document along the lines of the review template at <http://trac.tools.ietf.org/wg/precis/trac/wiki/StringprepReviewTemplate>.
- Included specific questions for each topic for consideration.
- Moved spot for individual protocol review to appendix. Not populated yet.

C.3. 02

- Cleared up details of comparison classes
- Added a section on changes in Unicode

C.4. 03

- Aligned comparison discussion with identifier discussion from draft-iab-identifier-comparison-00
- Added section on classes of strings ("Namey" and so on)

C.5. 04

Keepalive version

C.6. 05

- Changed classes of strings to align with framework doc
- Altered table in Appendix A
- Added all profiles evaluations from the wg wiki in appendix B

Authors' Addresses

Marc Blanchet
Viagenie
246 Aberdeen
Quebec, QC G1R 2E1
Canada

Email: Marc.Blanchet@viagenie.ca
URI: http://viagenie.ca

Andrew Sullivan
Dyn, Inc.
150 Dow St
Manchester, NH 03101
U.S.A.

Email: asullivan@dyn.com