PRECIS
Internet-Draft
Obsoletes: 4013 (if approved)
Intended status: Standards Track
Expires: April 13, 2015

P. Saint-Andre
&yet
A. Melnikov
Isode Ltd

October 10, 2014

Preparation and Comparison of Internationalized Strings Representing Usernames and Passwords
draft-ietf-precis-saslprepbis-08

Abstract

This document describes methods for handling Unicode strings representing usernames and passwords. This document obsoletes RFC 4013.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

Usernames and passwords are widely used for authentication and authorization on the Internet, either directly when provided in plaintext (as in the SASL PLAIN mechanism [RFC4616] or the HTTP Basic scheme [RFC2617]) or indirectly when provided as the input to a cryptographic algorithm such as a hash function (as in the SASL SCRAM mechanism [RFC5802] or the HTTP Digest scheme [RFC2617]).

To increase the likelihood that the input and comparison of usernames and passwords will work in ways that make sense for typical users throughout the world, this document defines rules for preparing and

comparing internationalized strings that represent usernames and passwords. Such strings consist of characters from the Unicode character set [UNICODE], especially characters outside the ASCII range [RFC20]. The rules for handling such strings are specified through profiles of the string classes defined in the PRECIS framework specification [I-D.ietf-precis-framework].

Profiles of the PRECIS framework enable software to handle Unicode characters outside the ASCII range in an automated way, so that such characters are treated carefully and consistently in application protocols. In large measure, these profiles are designed to protect application developers from the potentially negative consequences of supporting the full range of Unicode characters. For instance, in almost all application protocols it would be dangerous to treat the Unicode character SUPERSCRIPT ONE (U+0089) as equivalent to DIGIT ONE (U+0031), since that would result in false positives during comparison, authentication, and authorization (e.g., an attacker could easy spoof an account "user1@example.com").

Whereas a naive use of Unicode would make such attacks trivially easy, the PRECIS profile defined here for usernames generally protects applications from inadvertently causing such problems. (Similar considerations apply to passwords, although here it is desirable to support a wider range of characters so as to maximize entropy during authentication.)

The methods defined here might be applicable wherever usernames or passwords are used. However, the methods are not intended for use in preparing strings that are not usernames (e.g., email addresses and LDAP distinguished names), nor in cases where identifiers or secrets are not strings (e.g., keys and certificates) or require specialized handling.

This document obsoletes RFC 4013 (the "SASLprep" profile of stringprep [RFC3454]) but can be used by technologies other than the Simple Authentication and Security Layer (SASL) [RFC4422], such as HTTP authentication [RFC2617].

2. Terminology

Many important terms used in this document are defined in [I-D.ietf-precis-framework], [RFC5890], [RFC6365], and [UNICODE]. The term "non-ASCII space" refers to any Unicode code point having a general category of "Zs", with the exception of U+0020 (here called "ASCII space").

As used here, the term "password" is not literally limited to a word; i.e., a password could be a passphrase consisting of more than one word, perhaps separated by spaces or other such characters.

Some SASL mechanisms (e.g., CRAM-MD5, DIGEST-MD5, and SCRAM) specify that the authentication identity used in the context of such mechanisms is a "simple user name" (see Section 2 of [RFC4422] as well as [RFC4013]). Various application technologies also assume that the identity of a user or account takes the form of a username (e.g., authentication for the HyperText Transfer Protocol [RFC2617]), whether or not they use SASL. Note well that the exact form of a username in any particular SASL mechanism or application technology is a matter for implementation and deployment, and that a username does not necessarily map to any particular application identifier (such as the localpart of an email address).

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

3. Preparation, Comparison, and Enforcement

This document distinguishes between three different actions that an entity can take in handling a username or password:

o Enforcement entails applying all of the rules specified for a particular profile (UsernameIdentifierClass or PasswordFreeformClass) to an individual string.

o Comparison entails applying all of the rules specified for a particular profile to two separate strings, for the purpose of determining if the two strings are equivalent.

o Preparation entails only ensuring that the characters in an individual string are allowed by the underlying PRECIS base class (IdentifierClass or FreeformClass).

In most cases, "servers" are responsible for enforcement and "clients" are responsible only for preparation. Although some information regarding these responsibilities (e.g., the protocol slots in which usernames or passwords can be placed) needs to be provided in specifications that use the profiles defined in this document, the general outlines of such responsibilities are explained in the following sections.

4. Usernames

4.1. Definition

This document specifies that a username is a string of Unicode code points [UNICODE], encoded using UTF-8 [RFC3629], and structured either as an ordered sequence of "userparts" (where the complete username can consist of a single userpart or a space-separated sequence of userparts) or as a userpart@domainpart (where the domainpart is an IP literal, an IPv4 address, or a fully-qualified domain name).

The syntax for a username is defined as follows using the Augmented Backus-Naur Form (ABNF) [RFC5234].

```
   username   = userpart [1*(1*SP userpart)]
                / userpart '@' domainpart
   userpart   = 1*(idpoint)
                ;
                ; an "idpoint" is a UTF-8 encoded Unicode code point
                ; that conforms to the PRECIS "IdentifierClass"
                ;
   domainpart = IP-literal / IPv4address / ifqdn
                ;
                ; the "IPv4address" and "IP-literal" rules are
                ; defined in RFC 3986, and the first-match-wins
                ; (a.k.a. "greedy") algorithm described in RFC 3986
                ; applies
                ;
                ; reuse of the IP-literal rule from RFC 3986 implies
                ; that IPv6 addresses are enclosed in square brackets
                ; (i.e., beginning with '[' and ending with ']')
                ;
   ifqdn      = 1*1023(domainpoint)
                ;
                ; a "domainpoint" is a UTF-8 encoded Unicode code
                ; point that conforms to RFC 5890
                ;
```

All code points and blocks not explicitly allowed in the PRECIS IdentifierClass are disallowed; this includes private use characters, surrogate code points, and the other code points and blocks that were defined as "Prohibited Output" in [RFC4013]. In addition, common constructions such as "user@example.com" are allowed as usernames under this specification, as they were under [RFC4013].

Implementation Note: The username construct defined in this document does not necessarily match what all deployed applications

might refer to as a "username" or "userid", but instead provides a relatively safe subset of Unicode characters that can be used in existing SASL mechanisms and SASL-using application protocols, and even in most application protocols that do not currently use SASL.

A username MUST NOT be zero bytes in length. This rule is to be enforced after any normalization and mapping of code points.

In protocols that provide usernames as input to a cryptographic algorithm such as a hash function, the client will need to perform proper preparation of the username before applying the algorithm.

4.2. Preparation

An entity that prepares a string for inclusion in a username slot MUST ensure that the string consists only of Unicode code points that conform to the "IdentifierClass" base string class defined in [I-D.ietf-precis-framework]. In addition, the string MUST be encoded as UTF-8 [RFC3629].

4.3. Enforcement

An entity that performs enforcement in username slots MUST prepare a string as described in the previous section and MUST also apply the rules specified below for the UsernameIdentifierClass profile (these rules MUST be applied in the order shown).

1. Width Mapping Rule: Fullwidth and halfwidth characters MUST be mapped to their decomposition mappings.

2. Additional Mapping Rule: There is no additional mapping rule.

3. Case Mapping Rule: There is no case mapping rule (although see Section 4.5 below).

4. Normalization Rule: Unicode Normalization Form C (NFC) MUST be applied to all characters.

5. Exclusion Rule: There is no exclusion rule.

6. Directionality Rule: The "Bidi Rule" provided in [RFC5893] applies.

4.4. Comparison

An entity that performs comparison of two strings before or after their inclusion in username slots MUST prepare each string and enforce the rules specified in the previous two sections. The two

strings are to be considered equivalent if they are an exact octet-for-octet match (sometimes called "bit-string identity").

4.5. Case Mapping

Case mapping is a matter for the application protocol, protocol implementation, or end deployment. In general, this document suggests that it is preferable to perform case mapping, since not doing so can lead to false positives during authentication and authorization (as described in [RFC6943]) and can result in confusion among end users given the prevalence of case mapping in many existing protocols and applications. However, there can be good reasons to not perform case mapping, such as backward compatibility with deployed infrastructure.

In particular:

- o SASL mechanisms that directly re-use this profile MUST specify whether and when case mapping is to be applied to authentication identifiers. SASL mechanisms SHOULD delay any case mapping to the last possible moment, such as when doing a lookup by username, username comparisons, or generating a cryptographic salt from a username (if the last possible moment happens on the server, then decisions about case mapping can be a matter of deployment policy). In keeping with [RFC4422], SASL mechanisms are not to apply this or any other profile to authorization identifiers.

- o Application protocols that use SASL (such as IMAP [RFC3501] and XMPP [RFC6120]) and that directly re-use this profile MUST specify whether case mapping is to be applied to authorization identifiers. Such "SASL application protocols" SHOULD delay any case mapping of authorization identifiers to the last possible moment, which happens to necessarily be on the server side (this enables decisions about case mapping to be a matter of deployment policy). In keeping with [RFC4422], SASL application protocols are not to apply this or any other profile to authentication identifiers.

- o Application protocols that do not use SASL (such as HTTP authentication with the Basic and Digest schemes [RFC2617]) MUST specify whether and when case mapping is to be applied to authentication identifiers and authorization identifiers. Such "non-SASL application protocols" SHOULD delay any case mapping to the last possible moment, such as when doing a lookup by username, username comparisons, or generating a cryptographic salt from a username (if the last possible moment happens on the server, then decisions about case mapping can be a matter of deployment policy).

If the specification for a SASL mechanism, SASL application protocol, or non-SASL application protocol specifies the handling of case mapping for strings that conform to the UsernameIdentifierClass, it MUST clearly describe whether case mapping is required, recommended, or optional at the level of the protocol itself, implementations thereof, or service deployments.

4.6. Examples

The following examples illustrate a small number of usernames that are consistent with the format defined above (note that the characters < and > are used here to delineate the actual usernames and are not part of the username strings).

Table 1: A sample of legal usernames

#	Username	Notes
1	<juliet>	A userpart only
2	<fussball@example.com>	A userpart and domainpart
3	<fu&#xDF;ball@example.com>	The third character is LATIN SMALL LETTER SHARP S (U+00DF)
4	<&#x3C0;@example.com>	A userpart of GREEK SMALL LETTER PI (U+03C0)
5	<&#x3A3;@example.com>	A userpart of GREEK CAPITAL LETTER SIGMA (U+03A3)
6	<&#x3C3;@example.com>	A userpart of GREEK SMALL LETTER SIGMA (U+03C3)
7	<&#x3C2;@example.com>	A userpart of GREEK SMALL LETTER FINAL SIGMA (U+03C2)

Several points are worth noting. Regarding examples 2 and 3: although in German the character esszett (LATIN SMALL LETTER SHARP S, U+00DF) can mostly be used interchangeably with the two characters "ss", the userparts in these examples are different and (if desired) a server would need to enforce a registration policy that disallows one of them if the other is registered. Regarding examples 5, 6, and 7: optional case-mapping of GREEK CAPITAL LETTER SIGMA (U+03A3) to lowercase (i.e., to GREEK SMALL LETTER SIGMA, U+03C3) during comparison would result in matching the usernames in examples 5 and

6; however, because the PRECIS mapping rules do not account for the special status of GREEK SMALL LETTER FINAL SIGMA (U+03C2), the usernames in examples 5 and 7 or examples 6 and 7 would not be matched.

The following examples illustrate strings that are not valid usernames because they violate the format defined above.

Table 2: A sample of strings that violate the username rules

#	Non-Username string	Notes
8	<"juliet"@example.com>	Quotation marks (U+0022) in userpart
9	<foo bar@example.com>	Space (U+0020) in userpart
10	<@example.com>	Zero-length userpart
11	<henry&#x2163;@example.com>	The sixth character is ROMAN NUMERAL FOUR (U+2163)
12	<&#x265A;@example.com>	A localpart of BLACK CHESS KING (U+265A)

Here again, several points are worth noting. Regarding example 11, the Unicode character ROMAN NUMERAL FOUR (U+2163) has a compatibility equivalent of the string formed of LATIN CAPITAL LETTER I (U+0049) and LATIN CAPITAL LETTER V (U+0056), but characters with compatibility equivalents are not allowed in the PRECIS IdentiferClass. Regarding example 12: symbol characters such as BLACK CHESS KING (U+265A) are not allowed in the PRECIS IdentifierClass.

5. Passwords

5.1. Definition

This document specifies that a password is a string of Unicode code points [UNICODE], encoded using UTF-8 [RFC3629], and conformant to the PRECIS FreeformClass.

The syntax for a password is defined as follows using the Augmented Backus-Naur Form (ABNF) [RFC5234].

```
   password       = 1*(freepoint)
                    ;
                    ; a "freepoint" is a UTF-8 encoded
                    ; Unicode code point that conforms to
                    ; the PRECIS "FreeformClass"
                    ;
```

All code points and blocks not explicitly allowed in the PRECIS FreeformClass are disallowed; this includes private use characters, surrogate code points, and the other code points and blocks defined as "Prohibited Output" in Section 2.3 of RFC 4013.

A password MUST NOT be zero bytes in length. This rule is to be enforced after any normalization and mapping of code points.

In protocols that provide passwords as input to a cryptographic algorithm such as a hash function, the client will need to perform proper preparation of the password before applying the algorithm, since the password is not available to the server in plaintext form.

5.2. Preparation

An entity that prepares a string for inclusion in a password slot MUST ensure that the string consists only of Unicode code points that conform to the "FreeformClass" base string class defined in [I-D.ietf-precis-framework]. In addition, the string MUST be encoded as UTF-8 [RFC3629].

5.3. Enforcement

An entity that performs enforcement in password slots MUST prepare a string as described in the previous section and MUST also apply the rules specified below for the PasswordFreeformClass (these rules MUST be applied in the order shown).

1. Width Mapping Rule: Fullwidth and halfwidth characters MUST NOT be mapped to their decomposition mappings.

2. Additional Mapping Rule: Any instances of non-ASCII space MUST be mapped to ASCII space (U+0020); such an instance is any Unicode code point that has a compatibility mapping of any kind to U+0020 SPACE (including but not limited to <compat> as for U+0384 GREEK TONOS, <noBreak> as for U+2007 FIGURE SPACE, and <wide> as for U+3000 IDEOGRAPHIC SPACE).

3. Case Mapping Rule: Uppercase and titlecase characters MUST NOT be mapped to their lowercase equivalents.

4. Normalization Rule: Unicode Normalization Form C (NFC) MUST be applied to all characters.

5. Exclusion Rule: There is no exclusion rule.

6. Directionality Rule: There is no directionality rule.

With regard to directionality, the "Bidi Rule" (defined in [RFC5893]) and similar rules are unnecessary and inapplicable to passwords, since they can reduce the range of characters that are allowed in a string and therefore reduce the amount of entropy that is possible in a password. Furthermore, such rules are intended to minimize the possibility that the same string will be displayed differently on a system set for right-to-left display and a system set for left-to-right display; however, passwords are typically not displayed at all and are rarely meant to be interoperable across different systems in the way that non-secret strings like domain names and usernames are.

5.4. Comparison

An entity that performs comparison of two strings before or after their inclusion in password slots MUST prepare each string and enforce the rules specified in the previous two sections. The two strings are to be considered equivalent if they are an exact octet-for-octet match (sometimes called "bit-string identity").

5.5. Examples

The following examples illustrate a small number of passwords that are consistent with the format defined above (note that the characters < and > are used here to delineate the actual passwords and are not part of the username strings).

Table 3: A sample of legal passwords

#	Password	Notes
13	<correct horse battery staple>	ASCII space is allowed
14	<Correct Horse Battery Staple>	
15	<&#x3C0;&#xDF;&#xE5;>	Non-ASCII letters are OK (e.g., GREEK SMALL LETTER PI, U+03C0)
16	<Jack of &#x2666;s>	Symbols are OK (e.g., BLACK DIAMOND SUIT, U+2666)

The following examples illustrate strings that are not valid passwords because they violate the format defined above.

Table 4: A sample of strings that violate the password rules

#	Password	Notes
17	<foo&#x1680;bar>	Non-ASCII space (here, OGHAM SPACE MARK, U+1680) is not allowed
18	<my cat is a &#x9;by>	Controls are disallowed

6. Migration

The rules defined in this specification differ slightly from those defined by the SASLprep specification [RFC4013]. The following sections describe these differences, along with their implications for migration, in more detail.

6.1. Usernames

Deployments that currently use SASLprep for handling usernames might need to scrub existing data when migrating to use of the rules defined in this specification. In particular:

o SASLprep specified the use of Unicode Normalization Form KC (NFKC), whereas this usage of the PRECIS IdentifierClass employs Unicode Normalization Form C (NFC). In practice this change is

unlikely to cause significant problems, because NFKC provides methods for mapping Unicode code points with compatibility equivalents to those equivalents, whereas the PRECIS IdentifierClass entirely disallows Unicode code points with compatibility equivalents (i.e., during comparison NFKC is more "aggressive" about finding matches than is NFC). A few examples might suffice to indicate the nature of the problem: (1) U+017F LATIN SMALL LETTER LONG S is compatibility equivalent to U+0073 LATIN SMALL LETTER S (2) U+2163 ROMAN NUMERAL FOUR is compatibility equivalent to U+0049 LATIN CAPITAL LETTER I and U+0056 LATIN CAPITAL LETTER V (3) U+FB01 LATIN SMALL LIGATURE FI is compatibility equivalent to U+0066 LATIN SMALL LETTER F and U+0069 LATIN SMALL LETTER I. Under SASLprep, the use of NFKC also handled the mapping of fullwidth and halfwidth code points to their decomposition mappings. Although it is expected that code points with compatibility equivalents are rare in existing usernames, for migration purposes deployments might want to search their database of usernames for Unicode code points with compatibility equivalents and map those code points to their compatibility equivalents.

- SASLprep mapped the "characters commonly mapped to nothing" from Appendix B.1 of [RFC3454]) to nothing, whereas the PRECIS IdentifierClass entirely disallows most of these characters, which correspond to the code points from the "M" category defined under Section 6.13 of [I-D.ietf-precis-framework] (with the exception of U+1806 MONGOLIAN TODO SOFT HYPHEN, which was "commonly mapped to nothing" in Unicode 3.2 but at the time of this writing does not have a derived property of Default_Ignorable_Code_Point in Unicode 6.2). For migration purposes, deployments might want to remove code points contained in the PRECIS "M" category from usernames.

- SASLprep allowed uppercase and titlecase characters, whereas this usage of the PRECIS IdentifierClass maps uppercase and titlecase characters to their lowercase equivalents. For migration purposes, deployments can either convert uppercase and titlecase characters to their lowercase equivalents in usernames (thus losing the case information) or preserve uppercase and titlecase characters and ignore the case difference when comparing usernames.

6.2. Passwords

Depending on local service policy, migration from RFC 4013 to this specification might not involve any scrubbing of data (since passwords might not be stored in the clear anyway); however, service providers need to be aware of possible issues that might arise during migration. In particular:

- SASLprep specified the use of Unicode Normalization Form KC (NFKC), whereas this usage of the PRECIS FreeformClass employs Unicode Normalization Form C (NFC). Because NFKC is more aggressive about finding matches than NFC, in practice this change is unlikely to cause significant problems and indeed has the security benefit of probably resulting in fewer false positives when comparing passwords. A few examples might suffice to indicate the nature of the problem: (1) U+017F LATIN SMALL LETTER LONG S is compatibility equivalent to U+0073 LATIN SMALL LETTER S (2) U+2163 ROMAN NUMERAL FOUR is compatibility equivalent to U+0049 LATIN CAPITAL LETTER I and U+0056 LATIN CAPITAL LETTER V (3) U+FB01 LATIN SMALL LIGATURE FI is compatibility equivalent to U+0066 LATIN SMALL LETTER F and U+0069 LATIN SMALL LETTER I. Under SASLprep, the use of NFKC also handled the mapping of fullwidth and halfwidth code points to their decomposition mappings. Although it is expected that code points with compatibility equivalents are rare in existing passwords, some passwords that matched when SASLprep was used might no longer work when the rules in this specification are applied.

- SASLprep mapped the "characters commonly mapped to nothing" from Appendix B.1 of [RFC3454]) to nothing, whereas the PRECIS FreeformClass entirely disallows such characters, which correspond to the code points from the "M" category defined under Section 6.13 of [I-D.ietf-precis-framework] (with the exception of U+1806 MONGOLIAN TODO SOFT HYPHEN, which was commonly mapped to nothing in Unicode 3.2 but at the time of this writing is allowed by Unicode 6.2). In practice, this change will probably have no effect on comparison, but user-oriented software might reject such code points instead of ignoring them during password preparation.

7. IANA Considerations

The IANA shall add the following entries to the PRECIS Profiles Registry.

7.1. UsernameIdentifierClass

Name: UsernameIdentifierClass.

Applicability: Usernames in security and application protocols.

Base Class: IdentifierClass.

Replaces: The SASLprep profile of Stringprep.

Width Mapping Rule: Map fullwidth and halfwidth characters to their decomposition mappings.

Additional Mapping Rule: None.

Case Mapping Rule: To be defined by security or application protocols that use this profile.

Normalization Rule: NFC.

Exclusion Rule: None.

Directionality Rule: The "Bidi Rule" defined in RFC 5893 applies.

Enforcement: To be defined by security or application protocols that use this profile.

Specification: RFC XXXX. [Note to RFC Editor: please change XXXX to the number issued for this specification.]

7.2. PasswordFreeformClass

Name: PasswordFreeformClass.

Applicability: Passwords in security and application protocols.

Base Class: FreeformClass

Replaces: The SASLprep profile of Stringprep.

Width Mapping Rule: None.

Additional Mapping Rule: Map non-ASCII space characters to ASCII space.

Case Mapping Rule: None.

Normalization Rule: NFC.

Exclusion Rule: None.

Directionality Rule: None.

Enforcement: To be defined by security or application protocols that use this profile.

Specification: RFC XXXX. [Note to RFC Editor: please change XXXX to the number issued for this specification.]

8. Security Considerations

8.1. Password/Passphrase Strength

The ability to include a wide range of characters in passwords and passphrases can increase the potential for creating a strong password with high entropy. However, in practice, the ability to include such characters ought to be weighed against the possible need to reproduce them on various devices using various input methods.

8.2. Identifier Comparison

The process of comparing identifiers (such as SASL simple user names, authentication identifiers, and authorization identifiers) can lead to either false negatives or false positives, both of which have security implications. A more detailed discussion can be found in [RFC6943].

8.3. Reuse of PRECIS

The security considerations described in [I-D.ietf-precis-framework] apply to the "IdentifierClass" and "FreeformClass" base string classes used in this document for usernames and passwords, respectively.

8.4. Reuse of Unicode

The security considerations described in [UTS39] apply to the use of Unicode characters in usernames and passwords.

9. References

9.1. Normative References

[I-D.ietf-precis-framework]
Saint-Andre, P. and M. Blanchet, "Precis Framework: Handling Internationalized Strings in Protocols", draft-ietf-precis-framework-18 (work in progress), September 2014.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, November 2003.

[RFC5234] Crocker, D. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, January 2008.

[UNICODE] The Unicode Consortium, "The Unicode Standard, Version 6.3", 2013, <http://www.unicode.org/versions/Unicode6.3.0/>.

9.2. Informative References

[RFC20] Cerf, V., "ASCII format for network interchange", RFC 20, October 1969.

[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.

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

[RFC3501] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION 4rev1", RFC 3501, March 2003.

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

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

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

[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.

[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.

[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.

[RFC6120] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Core", RFC 6120, March 2011.

[RFC6365] Hoffman, P. and J. Klensin, "Terminology Used in Internationalization in the IETF", BCP 166, RFC 6365, September 2011.

[RFC6943] Thaler, D., "Issues in Identifier Comparison for Security Purposes", RFC 6943, May 2013.

[UTS39] The Unicode Consortium, "Unicode Technical Standard #39: Unicode Security Mechanisms", July 2012, <http://unicode.org/reports/tr39/>.

Appendix A. Differences from RFC 4013

This document builds upon the PRECIS framework defined in [I-D.ietf-precis-framework], which differs fundamentally from the stringprep technology [RFC3454] used in SASLprep [RFC4013]. The primary difference is that stringprep profiles allowed all characters except those which were explicitly disallowed, whereas PRECIS profiles disallow all characters except those which are explicitly allowed (this "inclusion model" was originally used for internationalized domain names in [RFC5891]; see [RFC5894] for further discussion). It is important to keep this distinction in mind when comparing the technology defined in this document to SASLprep [RFC4013].

The following substantive modifications were made from RFC 4013.

- A single SASLprep algorithm was replaced by two separate algorithms: one for usernames and another for passwords.

- The new preparation algorithms use PRECIS instead of a stringprep profile. The new algorithms work independenctly of Unicode versions.

- As recommended in the PRECIS framwork, changed the Unicode normalization form from NFKC to NFC.

- Some Unicode code points that were mapped to nothing in RFC 4013 are simply disallowed by PRECIS.

Appendix B. Acknowledgements

The following individuals provided helpful feedback on this document: Marc Blanchet, Alan DeKok, Joe Hildebrand, Jeffrey Hutzelman, Simon Josefsson, Jonathan Lennox, Matt Miller, Chris Newman, Yutaka OIWA, Pete Resnick, Andrew Sullivan, and Nico Williams. Nico in particular deserves special recognition for providing text that was used in Section 4.5. Thanks also to Yoshiro YONEYA and Takahiro NEMOTO for implementation feedback.

This document borrows some text from [RFC4013] and [RFC6120].

Authors' Addresses

Peter Saint-Andre
&yet

Email: peter@andyet.com
URI: https://andyet.com/

Alexey Melnikov
Isode Ltd
5 Castle Business Village
36 Station Road
Hampton, Middlesex TW12 2BX
UK

Email: Alexey.Melnikov@isode.com