      Preparation, Enforcement, and Comparison of Internationalized Strings
                      Representing Usernames and Passwords

Abstract

   This document describes updated methods for handling Unicode strings
   representing usernames and passwords. The previous approach was
   known as SASLprep (RFC 4013) and was based on stringprep (RFC 3454).
   The methods specified in this document provide a more sustainable
   approach to the handling of internationalized usernames and
   passwords. The preparation, enforcement, and comparison of
   internationalized strings (PRECIS) framework, RFC 7564, obsoletes RFC
   3454, and this document obsoletes RFC 4013.

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 PLAIN Simple Authentication and Security Layer
   (SASL) mechanism [RFC4616] and the HTTP Basic scheme
   [HTTP-BASIC-AUTH]) or indirectly when provided as the input to a
   cryptographic algorithm such as a hash function (as in the Salted
   Challenge Response Authentication Mechanism (SCRAM) SASL mechanism
   [RFC5802] and the HTTP Digest scheme [HTTP-DIGEST-AUTH]).

   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,
   enforcing, and comparing internationalized strings that represent
   usernames and passwords. Such strings consist of characters from the
   Unicode character set [Unicode], with special attention to characters
   outside the ASCII range [RFC20]. The rules for handling such strings
   are specified through profiles of the string classes defined in the
   preparation, enforcement, and comparison of internationalized strings
   (PRECIS) framework specification [RFC7564].

   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+00B9) as equivalent to DIGIT ONE
   (U+0031), because 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 for purposes of 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., Lightweight Directory
   Access Protocol (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 SASL [RFC4422],
such as HTTP authentication as specified in [HTTP-BASIC-AUTH] and
[HTTP-DIGEST-AUTH].

This document does not modify the handling of internationalized
strings in usernames and passwords as prescribed by existing
application protocols that use SASLprep. If the community that uses
such an application protocol wishes to modernize its handling of
internationalized strings to use PRECIS instead of stringprep, it
needs to explicitly update the existing application protocol
definition (one example is [XMPP-ADDR], which is intended to obsolete
[RFC6122]). Non-coordinated updates to protocol implementations are
discouraged because they can have a negative impact on
interoperability and security.

2. Terminology

Many important terms used in this document are defined in [RFC5890],
[RFC6365], [RFC7564], and [Unicode]. The term "non-ASCII space"
refers to any Unicode code point having a Unicode 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, punctuation, or other
non-alphanumeric 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 as
specified in [HTTP-BASIC-AUTH] and [HTTP-DIGEST-AUTH]), 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. Usernames

3.1. Definition

   This document specifies that a username is a string of Unicode code
   points [Unicode], encoded using UTF-8 [RFC3629], and structured as an
   ordered sequence of "userparts". A userpart is allowed to contain
   only code points that are in turn allowed by the PRECIS
   IdentifierClass defined in Section 4.2 of [RFC7564], and thus
   consists almost exclusively of letters and digits. A username can
   consist of a single userpart or a space-separated sequence of
   userparts.

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

       username   = userpart *(1*SP userpart)
       userpart   = 1*(idbyte)
                  ;
                  ; an "idbyte" is a byte used to represent a
                  ; UTF-8 encoded Unicode code point that can be
                  ; contained in a string that conforms to the
                  ; PRECIS IdentifierClass
                  ;

   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" (e.g., the Network Access
   Identifier from [RFC7542]) 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 in application protocols that use
      SASL, 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
   enforcement of the rules for the UsernameCaseMapped or
   UsernameCasePreserved profile before applying the algorithm.

This specification defines two profiles for usernames: one that performs case mapping and one that performs case preservation (see further discussion under Section 3.4).

3.2. UsernameCaseMapped Profile

The definition of the UsernameCaseMapped profile of the IdentifierClass is provided in the following sections, including detailed information about preparation, enforcement, and comparison (for details on the distinction between these actions, refer to [RFC7564]).

3.2.1. Preparation

An entity that prepares a string according to this profile MUST first map fullwidth and halfwidth characters to their decomposition mappings (see Unicode Standard Annex #11 [UAX11]). This is necessary because the PRECIS "HasCompat" category specified in Section 9.17 of [RFC7564] would otherwise forbid fullwidth and halfwidth characters. After applying this width-mapping rule, the entity then MUST ensure that the string consists only of Unicode code points that conform to the PRECIS IdentifierClass defined in Section 4.2 of [RFC7564]. In addition, the entity then MUST encode the string as UTF-8 [RFC3629].

3.2.2. Enforcement

An entity that performs enforcement according to this profile MUST prepare a string as described in Section 3.2.1 and MUST also apply the rules specified below for the UsernameCaseMapped profile (these rules MUST be applied in the order shown):

1. Width-Mapping Rule: Applied as part of preparation (see above).

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

3. Case-Mapping Rule: Uppercase and titlecase characters MUST be mapped to their lowercase equivalents, preferably using Unicode Default Case Folding as defined in the Unicode Standard [Unicode] (at the time of this writing, the algorithm is specified in Chapter 3 of [Unicode7.0], but the chapter number might change in a future version of the Unicode Standard); see further discussion in Section 3.4.

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

   5. Directionality Rule: Applications MUST apply the "Bidi Rule"
       defined in [RFC5893] to strings that contain right-to-left
       characters (i.e., each of the six conditions of the Bidi Rule
       must be satisfied).

3.2.3. Comparison

   An entity that performs comparison of two strings according to this
   profile MUST prepare each string as specified in Section 3.2.1 and
   then enforce the rules specified in Section 3.2.2. The two strings
   are to be considered equivalent if they are an exact octet-for-octet
   match (sometimes called "bit-string identity").

3.3. UsernameCasePreserved Profile

   The definition of the UsernameCasePreserved profile of the
   IdentifierClass is provided in the following sections, including
   detailed information about preparation, enforcement, and comparison
   (for details on the distinction between these actions, refer to
   [RFC7564]).

3.3.1. Preparation

   An entity that prepares a string according to this profile MUST first
   map fullwidth and halfwidth characters to their decomposition
   mappings (see Unicode Standard Annex #11 [UAX11]). This is necessary
   because the PRECIS "HasCompat" category specified in Section 9.17 of
   [RFC7564] would otherwise forbid fullwidth and halfwidth characters.
   After applying this width-mapping rule, the entity then MUST ensure
   that the string consists only of Unicode code points that conform to
   the PRECIS IdentifierClass defined in Section 4.2 of [RFC7564]. In
   addition, the entity then MUST encode the string as UTF-8 [RFC3629].

3.3.2. Enforcement

   An entity that performs enforcement according to this profile MUST
   prepare a string as described in Section 3.3.1 and MUST also apply
   the rules specified below for the UsernameCasePreserved profile
   (these rules MUST be applied in the order shown):

   1. Width-Mapping Rule: Applied as part of preparation (see above).

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

   3. Case-Mapping Rule: Uppercase and titlecase characters MUST NOT be
       mapped to their lowercase equivalents; see further discussion in
       Section 3.4.

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

   5. Directionality Rule: Applications MUST apply the "Bidi Rule"
       defined in [RFC5893] to strings that contain right-to-left
       characters (i.e., each of the six conditions of the Bidi Rule
       must be satisfied).

3.3.3. Comparison

   An entity that performs comparison of two strings according to this
   profile MUST prepare each string as specified in Section 3.3.1 and
   then enforce the rules specified in Section 3.3.2. The two strings
   are to be considered equivalent if they are an exact octet-for-octet
   match (sometimes called "bit-string identity").

3.4. Case Mapping vs. Case Preservation

   In order to accommodate the widest range of username constructs in
   applications, this document defines two username profiles:
   UsernameCaseMapped and UsernameCasePreserved. These two profiles
   differ only in the Case-Mapping Rule and are otherwise identical.

   Case mapping is a matter for the application protocol, protocol
   implementation, or end deployment. In general, this document
   suggests that it is preferable to apply the UsernameCaseMapped
   profile and therefore perform case mapping, because 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 apply the
   UsernameCasePreserved profile and thus not perform case mapping, such
   as backward compatibility with deployed infrastructure.

   In particular:

   o SASL mechanisms that follow the recommendations in this document
      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, performing 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, only to authentication identifiers.

   o Application protocols that use SASL (such as IMAP [RFC3501] and
      the Extensible Messaging and Presence Protocol (XMPP) [RFC6120])
      and that directly reuse this profile MUST specify whether or not
      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,
      only to authorization identifiers.

   o Application protocols that do not use SASL (such as HTTP
      authentication with the HTTP Basic and Digest schemes as specified
      in [HTTP-BASIC-AUTH] and [HTTP-DIGEST-AUTH]) but that directly
      reuse this profile MUST specify whether and when case mapping is
      to be applied to authentication identifiers or authorization
      identifiers, or both. Such "non-SASL application protocols"
      SHOULD delay any case mapping to the last possible moment, such as
      when doing a lookup by username, performing 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 uses the UsernameCaseMapped profile,
   it MUST clearly describe whether case mapping is to be applied at the
   level of the protocol itself, implementations thereof, or service
   deployments (each of these approaches can be legitimate, depending on
   the application in question).

3.5. Application-Layer Constructs

   Both the UsernameCaseMapped and UsernameCasePreserved profiles enable
   an application protocol, implementation, or deployment to create
   application-layer constructs such as a username that is a space-
   separated set of userparts like "Firstname Middlename Lastname".
   Although such a construct is not a profile of the PRECIS
   IdentifierClass (because U+0020 SPACE is not allowed in the
   IdentifierClass), it can be created at the application layer because
   U+0020 SPACE can be used as a separator between instances of the
   PRECIS IdentifierClass (e.g., userparts as defined in this
   specification).

3.6. Examples

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

   +---------------------------+--------------------------------+
   | # | Userpart | Notes |
   +---------------------------+--------------------------------+
   | 1 | <juliet@example.com> | The at-sign is allowed in the |
   | | | PRECIS IdentifierClass |
   +---------------------------+--------------------------------+
   | 2 | <fussball> | |
   +---------------------------+--------------------------------+
   | 3 | <fußball> | The third character is LATIN |
   | | | SMALL LETTER SHARP S (U+00DF) |
   +---------------------------+--------------------------------+
   | 4 | <π> | A userpart of GREEK SMALL |
   | | | LETTER PI (U+03C0) |
   +---------------------------+--------------------------------+
   | 5 | <Σ> | A userpart of GREEK CAPITAL |
   | | | LETTER SIGMA (U+03A3) |
   +---------------------------+--------------------------------+
   | 6 | <σ> | A userpart of GREEK SMALL |
   | | | LETTER SIGMA (U+03C3) |
   +---------------------------+--------------------------------+
   | 7 | <ς> | A userpart of GREEK SMALL |
   | | | LETTER FINAL SIGMA (U+03C2) |
   +---------------------------+--------------------------------+

                  Table 1: A Sample of Legal Userparts

   Several points are worth noting. Regarding examples 2 and 3:
   although in German the character eszett (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 userparts 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
   userparts in examples 5 and 7 or examples 6 and 7 would not be
   matched during comparison.

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

```
+-------------------------+-------------------------------+
| # | Non-Userpart String | Notes                         |
+-------------------------+-------------------------------+
| 8 | <foo bar>           | Space (U+0020) is disallowed in |
|   |                     | the userpart                  |
+-------------------------+-------------------------------+
| 9 | <>                  | Zero-length userpart          |
+-------------------------+-------------------------------+
| 10| <henry&#x2163;>     | The sixth character is ROMAN  |
|   |                     | NUMERAL FOUR (U+2163)         |
+-------------------------+-------------------------------+
| 11| <&#x265A;>          | A localpart of BLACK CHESS KING |
|   |                     | (U+265A)                      |
+-------------------------+-------------------------------+
```

       Table 2: A Sample of Strings That Violate the Userpart Rule

Here again, several points are worth noting. Regarding example 8:
although this is not a valid userpart, it is a valid username because
it is a space-separated sequence of userparts. Regarding example 10:
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
IdentifierClass. Regarding example 11: symbol characters such as
BLACK CHESS KING (U+265A) are not allowed in the PRECIS
IdentifierClass.

4. Passwords

4.1. Definition

   This document specifies that a password is a string of Unicode code
   points [Unicode], encoded using UTF-8 [RFC3629], and conformant to
   the OpaqueString profile (specified below) of the PRECIS
   FreeformClass defined in Section 4.3 of [RFC7564].

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

```
      password  = 1*(freebyte)
                  ;
                  ; a "freebyte" is a byte used to represent a
                  ; UTF-8 encoded Unicode code point that can be
                  ; contained in a string 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 (when corrected per
   [Err1812]).

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

      Note: Some existing systems allow an empty string in places where
      a password would be expected (e.g., command-line tools that might
      be called from an automated script, or servers that might need to
      be restarted without human intervention). From the perspective of
      this document (and RFC 4013 before it), these empty strings are
      not passwords but are workarounds for the practical difficulty of
      using passwords in certain scenarios. The prohibition of
      zero-length passwords is not a recommendation regarding password
      strength (because a password of only one byte is highly insecure)
      but is meant to prevent applications from mistakenly omitting a
      password entirely; such an outcome is possible when
      internationalized characters are accepted, because a non-empty
      sequence of characters can result in a zero-length password after
      canonicalization.

   In protocols that provide passwords as input to a cryptographic
   algorithm such as a hash function, the client will need to perform
   enforcement of the rules for the OpaqueString profile before applying
   the algorithm, because the password is not available to the server in
   plaintext form.

4.2. OpaqueString Profile

   The definition of the OpaqueString profile is provided in the
   following sections, including detailed information about preparation,
   enforcement, and comparison (for details on the distinction between
   these actions, refer to [RFC7564]).

4.2.1. Preparation

   An entity that prepares a string according to this profile MUST
   ensure that the string consists only of Unicode code points that
   conform to the FreeformClass base string class defined in [RFC7564].
   In addition, the entity MUST encode the string as UTF-8 [RFC3629].

4.2.2. Enforcement

   An entity that performs enforcement according to this profile MUST
   prepare a string as described in Section 4.2.1 and MUST also apply
   the rules specified below for the OpaqueString profile (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 (see Unicode Standard
       Annex #11 [UAX11]).

   2. Additional Mapping Rule: Any instances of non-ASCII space MUST be
       mapped to ASCII space (U+0020); a non-ASCII space is any Unicode
       code point having a Unicode general category of "Zs" (with the
       exception of U+0020).

   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. Directionality Rule: There is no directionality rule. The "Bidi
       Rule" (defined in [RFC5893]) and similar rules are unnecessary
       and inapplicable to passwords, because 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. Such rules
       are intended to minimize the possibility that the same string

will be displayed differently on a layout system set for
right-to-left display and a layout system set for left-to-right
display; however, passwords are typically not displayed at all
and are rarely meant to be interoperable across different layout
systems in the way that non-secret strings like domain names and
usernames are. Furthermore, it is perfectly acceptable for
opaque strings other than passwords to be presented differently
in different layout systems, as long as the presentation is
consistent in any given layout system.

4.2.3. Comparison

An entity that performs comparison of two strings according to this
profile MUST prepare each string as specified in Section 4.2.1 and
then enforce the rules specified in Section 4.2.2. The two strings
are to be considered equivalent if they are an exact octet-for-octet
match (sometimes called "bit-string identity").

4.3. 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 password strings).

#	Password	Notes
12	<correct horse battery staple>	ASCII space is allowed
13	<Correct Horse Battery Staple>	Differs by case from example 12
14	<πßå>	Non-ASCII letters are OK (e.g., GREEK SMALL LETTER PI (U+03C0))
15	<Jack of ♦s>	Symbols are OK (e.g., BLACK DIAMOND SUIT (U+2666))
16	<foo bar>	OGHAM SPACE MARK (U+1680) is mapped to U+0020, and thus the full string is mapped to <foo bar>

Table 3: A Sample of Legal Passwords

The following example illustrates a string that is not a valid
password because it violates the format defined above.

```
+----------------------------------+----------------------------+
| # | Password                     | Notes                      |
+----------------------------------+----------------------------+
| 17| <my cat is a &#x9;by>        | Controls are disallowed    |
+----------------------------------+----------------------------+
```

Table 4: A String That Violates the Password Rules

5. Use in Application Protocols

This specification defines only the PRECIS-based rules for the
handling of strings conforming to the UsernameCaseMapped and
UsernameCasePreserved profiles of the PRECIS IdentifierClass, and
strings conforming to the OpaqueString profile of the PRECIS
FreeformClass. It is the responsibility of an application protocol
to specify the protocol slots in which such strings can appear, the
entities that are expected to enforce the rules governing such
strings, and at what points during protocol processing or interface
handling the rules need to be enforced. See Section 6 of [RFC7564]
for guidelines on using PRECIS profiles in applications.

Above and beyond the PRECIS-based rules specified here, application
protocols can also define application-specific rules governing such
strings (rules regarding minimum or maximum length, further
restrictions on allowable characters or character ranges, safeguards
to mitigate the effects of visually similar characters, etc.),
application-layer constructs (see Section 3.5), and related matters.

Some PRECIS profile definitions encourage entities that enforce the
rules to be liberal in what they accept. However, for usernames and
passwords such a policy can be problematic, because it can lead to
false positives. An in-depth discussion can be found in [RFC6943].

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 they migrate to the rules defined in
   this specification. In particular:

   o SASLprep specified the use of Unicode Normalization Form KC
      (NFKC), whereas the UsernameCaseMapped and UsernameCasePreserved
      profiles employ 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 NFC). A few examples
      might suffice to indicate the nature of the problem:

      1. LATIN SMALL LETTER LONG S (U+017F) is compatibility equivalent
          to LATIN SMALL LETTER S (U+0073).

      2. ROMAN NUMERAL FOUR (U+2163) is compatibility equivalent to
          LATIN CAPITAL LETTER I (U+0049) and LATIN CAPITAL LETTER V
          (U+0056).

      3. LATIN SMALL LIGATURE FI (U+FB01) is compatibility equivalent
          to LATIN SMALL LETTER F (U+0066) and LATIN SMALL LETTER I
          (U+0069).

      Under SASLprep, the use of NFKC also handled the mapping of
      fullwidth and halfwidth code points to their decomposition
      mappings.

      For migration purposes, operators might want to search their
      database of usernames for names containing Unicode code points
      with compatibility equivalents and, where there is no conflict,
      map those code points to their equivalents. Naturally, it is
      possible that during this process the operator will discover
      conflicting usernames (e.g., HENRYIV with the last two characters
      being LATIN CAPITAL LETTER I (U+0049) and LATIN CAPITAL LETTER V
      (U+0056) vs. "HENRYIV" with the last character being ROMAN NUMERAL
      FOUR (U+2163), which is compatibility equivalent to U+0049 and
      U+0056); in these cases, the operator will need to determine how
      to proceed -- for instance, by disabling the account whose name
      contains a Unicode code point with a compatibility equivalent.
      Such cases are probably rare, but it is important for operators to
      be aware of them.

   o 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 PRECIS "M" category defined
      under Section 9.13 of [RFC7564] (with the exception of MONGOLIAN
      TODO SOFT HYPHEN (U+1806), 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 7.0).
      For migration purposes, the operator might want to remove from
      usernames any code points contained in the PRECIS "M" category
      (e.g., SOFT HYPHEN (U+00AD)). Because these code points would
      have been "mapped to nothing" in stringprep, in practice a user
      would not notice the difference if, upon migration to PRECIS, the
      code points are removed.

   o SASLprep allowed uppercase and titlecase characters, whereas the
      UsernameCaseMapped profile maps uppercase and titlecase characters
      to their lowercase equivalents (by contrast, the
      UsernameCasePreserved profile matches SASLprep in this regard).
      For migration purposes, the operator can use either the
      UsernameCaseMapped profile (thus losing the case information) or
      the UsernameCasePreserved profile (thus ignoring 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 (because
   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:

   o SASLprep specified the use of Unicode Normalization Form KC
      (NFKC), whereas the OpaqueString profile 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. LATIN SMALL LETTER LONG S (U+017F) is compatibility equivalent
          to LATIN SMALL LETTER S (U+0073).

      2. ROMAN NUMERAL FOUR (U+2163) is compatibility equivalent to
          LATIN CAPITAL LETTER I (U+0049) and LATIN CAPITAL LETTER V
          (U+0056).

   3. LATIN SMALL LIGATURE FI (U+FB01) is compatibility equivalent
       to LATIN SMALL LETTER F (U+0066) and LATIN SMALL LETTER I
       (U+0069).

   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.

   o 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 PRECIS "M" category defined under
      Section 9.13 of [RFC7564] (with the exception of MONGOLIAN TODO
      SOFT HYPHEN (U+1806), which was commonly mapped to nothing in
      Unicode 3.2 but at the time of this writing is allowed by
      Unicode 7.0). 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

   IANA has made the updates described below.

7.1. UsernameCaseMapped Profile

   IANA has added the following entry to the "PRECIS Profiles" registry.

   Name: UsernameCaseMapped.

   Base Class: IdentifierClass.

   Applicability: Usernames in security and application protocols.

   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: Map uppercase and titlecase characters to
      lowercase.

   Normalization Rule: NFC.

   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 7613 (this document), Section 3.2.

7.2. UsernameCasePreserved Profile

   IANA has added the following entry to the "PRECIS Profiles" registry.

   Name: UsernameCasePreserved.

   Base Class: IdentifierClass.

   Applicability: Usernames in security and application protocols.

   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: None.

   Normalization Rule: NFC.

   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 7613 (this document), Section 3.3.

7.3. OpaqueString Profile

   IANA has added the following entry to the "PRECIS Profiles" registry.

   Name: OpaqueString.

   Base Class: FreeformClass.

   Applicability: Passwords and other opaque strings in security and
      application protocols.

   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.

   Directionality Rule: None.

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

   Specification: RFC 7613 (this document), Section 4.2.

7.4. Stringprep Profile

   The stringprep specification [RFC3454] did not provide for entries in
   the "Stringprep Profiles" registry to have any state except "Current"
   or "Not Current". Because this document obsoletes RFC 4013, which
   registered the SASLprep profile of stringprep, IANA has marked that
   profile as "Not Current" and cited this document as an additional
   reference.

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 [RFC7564] 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

   [RFC2119]   Bradner, S., "Key words for use in RFCs to Indicate
               Requirement Levels", BCP 14, RFC 2119,
               DOI 10.17487/RFC2119, March 1997,
               <http://www.rfc-editor.org/info/rfc2119>.

   [RFC3629]   Yergeau, F., "UTF-8, a transformation format of
               ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629,
               November 2003, <http://www.rfc-editor.org/info/rfc3629>.

   [RFC5234]   Crocker, D., Ed., and P. Overell, "Augmented BNF for
               Syntax Specifications: ABNF", STD 68, RFC 5234,
               DOI 10.17487/RFC5234, January 2008,
               <http://www.rfc-editor.org/info/rfc5234>.

   [RFC5890]   Klensin, J., "Internationalized Domain Names for
               Applications (IDNA): Definitions and Document Framework",
               RFC 5890, DOI 10.17487/RFC5890, August 2010,
               <http://www.rfc-editor.org/info/rfc5890>.

   [RFC6365]   Hoffman, P. and J. Klensin, "Terminology Used in
               Internationalization in the IETF", BCP 166, RFC 6365,
               DOI 10.17487/RFC6365, September 2011,
               <http://www.rfc-editor.org/info/rfc6365>.

   [RFC7564]   Saint-Andre, P. and M. Blanchet, "PRECIS Framework:
               Preparation, Enforcement, and Comparison of
               Internationalized Strings in Application Protocols",
               RFC 7564, DOI 10.17487/RFC7564, May 2015,
               <http://www.rfc-editor.org/info/rfc7564>.

   [UAX11]   Unicode Standard Annex #11, "East Asian Width", edited by
               Ken Lunde. An integral part of The Unicode Standard,
               <http://unicode.org/reports/tr11/>.

   [Unicode] The Unicode Consortium, "The Unicode Standard",
              <http://www.unicode.org/versions/latest/>.

   [Unicode7.0]
              The Unicode Consortium, "The Unicode Standard,
              Version 7.0.0", (Mountain View, CA: The Unicode
              Consortium, 2014 ISBN 978-1-936213-09-2),
              <http://www.unicode.org/versions/Unicode7.0.0/>.

9.2. Informative References

   [Err1812] RFC Errata, Erratum ID 1812, RFC 4013,
              <http://www.rfc-editor.org>.

   [HTTP-BASIC-AUTH]
              Reschke, J., "The 'Basic' HTTP Authentication Scheme",
              Work in Progress, draft-ietf-httpauth-basicauth-update-07,
              February 2015.

   [HTTP-DIGEST-AUTH]
              Shekh-Yusef, R., Ed., Ahrens, D., and S. Bremer, "HTTP
              Digest Access Authentication", Work in Progress,
              draft-ietf-httpauth-digest-19, April 2015.

   [RFC20]   Cerf, V., "ASCII format for network interchange", STD 80,
              RFC 20, DOI 10.17487/RFC0020, October 1969,
              <http://www.rfc-editor.org/info/rfc20>.

   [RFC3454] Hoffman, P. and M. Blanchet, "Preparation of
              Internationalized Strings ("stringprep")", RFC 3454,
              DOI 10.17487/RFC3454, December 2002,
              <http://www.rfc-editor.org/info/rfc3454>.

   [RFC3501] Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL -
              VERSION 4rev1", RFC 3501, DOI 10.17487/RFC3501,
              March 2003, <http://www.rfc-editor.org/info/rfc3501>.

   [RFC4013] Zeilenga, K., "SASLprep: Stringprep Profile for User Names
              and Passwords", RFC 4013, DOI 10.17487/RFC4013,
              February 2005, <http://www.rfc-editor.org/info/rfc4013>.

   [RFC4422] Melnikov, A., Ed., and K. Zeilenga, Ed., "Simple
              Authentication and Security Layer (SASL)", RFC 4422,
              DOI 10.17487/RFC4422, June 2006,
              <http://www.rfc-editor.org/info/rfc4422>.

   [RFC4616]   Zeilenga, K., Ed., "The PLAIN Simple Authentication and
               Security Layer (SASL) Mechanism", RFC 4616,
               DOI 10.17487/RFC4616, August 2006,
               <http://www.rfc-editor.org/info/rfc4616>.

   [RFC5802]   Newman, C., Menon-Sen, A., Melnikov, A., and N. Williams,
               "Salted Challenge Response Authentication Mechanism
               (SCRAM) SASL and GSS-API Mechanisms", RFC 5802,
               DOI 10.17487/RFC5802, July 2010,
               <http://www.rfc-editor.org/info/rfc5802>.

   [RFC5891]   Klensin, J., "Internationalized Domain Names in
               Applications (IDNA): Protocol", RFC 5891,
               DOI 10.17487/RFC5891, August 2010,
               <http://www.rfc-editor.org/info/rfc5891>.

   [RFC5893]   Alvestrand, H., Ed., and C. Karp, "Right-to-Left Scripts
               for Internationalized Domain Names for Applications
               (IDNA)", RFC 5893, DOI 10.17487/RFC5893, August 2010,
               <http://www.rfc-editor.org/info/rfc5893>.

   [RFC5894]   Klensin, J., "Internationalized Domain Names for
               Applications (IDNA): Background, Explanation, and
               Rationale", RFC 5894, DOI 10.17487/RFC5894, August 2010,
               <http://www.rfc-editor.org/info/rfc5894>.

   [RFC6120]   Saint-Andre, P., "Extensible Messaging and Presence
               Protocol (XMPP): Core", RFC 6120, DOI 10.17487/RFC6120,
               March 2011, <http://www.rfc-editor.org/info/rfc6120>.

   [RFC6122]   Saint-Andre, P., "Extensible Messaging and Presence
               Protocol (XMPP): Address Format", RFC 6122,
               DOI 10.17487/RFC6122, March 2011,
               <http://www.rfc-editor.org/info/rfc6122>.

   [RFC6943]   Thaler, D., Ed., "Issues in Identifier Comparison for
               Security Purposes", RFC 6943, DOI 10.17487/RFC6943,
               May 2013, <http://www.rfc-editor.org/info/rfc6943>.

   [RFC7542]   DeKok, A., "The Network Access Identifier", RFC 7542,
               DOI 10.17487/RFC7542, May 2015,
               <http://www.rfc-editor.org/info/rfc7542>.

   [UTS39]   Unicode Technical Standard #39, "Unicode Security
               Mechanisms", edited by Mark Davis and Michel Suignard,
               <http://unicode.org/reports/tr39/>.

   [XMPP-ADDR]
               Saint-Andre, P., "Extensible Messaging and Presence
               Protocol (XMPP): Address Format", Work in Progress,
               draft-ietf-xmpp-6122bis-24, June 2015.

Appendix A. Differences from RFC 4013

   This document builds upon the PRECIS framework defined in [RFC7564],
   which differs fundamentally from the stringprep technology [RFC3454]
   used in SASLprep [RFC4013]. The primary difference is that
   stringprep profiles allowed all characters except those characters
   that were explicitly disallowed, whereas PRECIS profiles disallow all
   characters except those characters that 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.

   o A single SASLprep algorithm was replaced by three separate
      algorithms: one for usernames with case mapping, one for usernames
      with case preservation, and one for passwords.

   o The new preparation algorithms use PRECIS instead of a stringprep
      profile. The new algorithms work independently of Unicode
      versions.

   o As recommended in the PRECIS framework, changed the Unicode
      normalization form from NFKC to NFC.

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

Acknowledgements

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
   United Kingdom

   Email: Alexey.Melnikov@isode.com