PRECIS
Internet-Draft
Obsoletes: 3454 (if approved)
Intended status: Standards Track
Expires: August 23, 2015

P. Saint-Andre
&yet
M. Blanchet
Viagenie

February 19, 2015

PRECIS Framework: Preparation, Enforcement, and Comparison of Internationalized Strings in Application Protocols
draft-ietf-precis-framework-23

Abstract

Application protocols using Unicode characters in protocol strings need to properly handle such strings in order to enforce internationalization rules for strings placed in various protocol slots (such as addresses and identifiers) and to perform valid comparison operations (e.g., for purposes of authentication or authorization). This document defines a framework enabling application protocols to perform the preparation, enforcement, and comparison of internationalized strings ("PRECIS") in a way that depends on the properties of Unicode characters and thus is agile with respect to versions of Unicode. As a result, this framework provides a more sustainable approach to the handling of internationalized strings than the previous framework, known as Stringprep (RFC 3454). This document obsoletes RFC 3454.

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 August 23, 2015.

Table of Contents

1. Introduction

Application protocols using Unicode characters [Unicode7.0] in protocol strings need to properly handle such strings in order to enforce internationalization rules for strings placed in various protocol slots (such as addresses and identifiers) and to perform valid comparison operations (e.g., for purposes of authentication or authorization). This document defines a framework enabling application protocols to perform the preparation, enforcement, and comparison of internationalized strings ("PRECIS") in a way that depends on the properties of Unicode characters and thus is agile with respect to versions of Unicode.

As described in the PRECIS problem statement [RFC6885], many IETF protocols have used the Stringprep framework [RFC3454] as the basis for preparing, enforcing, and comparing protocol strings that contain Unicode characters, especially characters outside the ASCII range [RFC20]. The Stringprep framework was developed during work on the original technology for internationalized domain names (IDNs), here called "IDNA2003" [RFC3490], and Nameprep [RFC3491] was the Stringprep profile for IDNs. At the time, Stringprep was designed as a general framework so that other application protocols could define their own Stringprep profiles. Indeed, a number of application protocols defined such profiles.

After the publication of [RFC3454] in 2002, several significant issues arose with the use of Stringprep in the IDN case, as documented in the IAB's recommendations regarding IDNs [RFC4690] (most significantly, Stringprep was tied to Unicode version 3.2). Therefore, the newer IDNA specifications, here called "IDNA2008" ([RFC5890], [RFC5891], [RFC5892], [RFC5893], [RFC5894]), no longer use Stringprep and Nameprep. This migration away from Stringprep for IDNs prompted other "customers" of Stringprep to consider new approaches to the preparation, enforcement, and comparison of internationalized strings, as described in [RFC6885].

This document defines a framework for a post-Stringprep approach to the preparation, enforcement, and comparison of internationalized strings in application protocols, based on several principles:

1. Define a small set of string classes that specify the Unicode characters (i.e., specific "code points") appropriate for common application protocol constructs.

2. Define each PRECIS string class in terms of Unicode code points and their properties so that an algorithm can be used to determine whether each code point or character category is (a)

valid, (b) allowed in certain contexts, (c) disallowed, or (d) unassigned.

3. Use an "inclusion model" such that a string class consists only of code points that are explicitly allowed, with the result that any code point not explicitly allowed is forbidden.

4. Enable application protocols to define profiles of the PRECIS string classes if necessary (addressing matters such as width mapping, case mapping, Unicode normalization, and directionality) but strongly discourage the multiplication of profiles beyond necessity in order to avoid violations of the Principle of Least User Astonishment.

It is expected that this framework will yield the following benefits:

- Application protocols will be agile with regard to Unicode versions.

- Implementers will be able to share code point tables and software code across application protocols, most likely by means of software libraries.

- End users will be able to acquire more accurate expectations about the characters that are acceptable in various contexts. Given this more uniform set of string classes, it is also expected that copy/paste operations between software implementing different application protocols will be more predictable and coherent.

Whereas the string classes define the "baseline" code points for a range of applications, profiling enables application protocols to apply the string classes in ways that are appropriate for common constructs such as usernames [I-D.ietf-precis-saslprepbis], opaque strings such as passwords [I-D.ietf-precis-saslprepbis], and nicknames [I-D.ietf-precis-nickname]. Profiles are responsible for defining the handling of right-to-left characters as well as various mapping operations of the kind also discussed for IDNs in [RFC5895], such as case preservation or lowercasing, Unicode normalization, mapping of certain characters to other characters or to nothing, and mapping of full-width and half-width characters.

When an application applies a profile of a PRECIS string class, it transforms an input string (which might or might not be conforming) into an output string that definitively conforms to the profile. In particular, this document focuses on the resulting ability to achieve the following objectives:

a. Enforcing all the the rules of a profile for a single output string (e.g., to determine if a string can be included in a protocol slot, communicated to another entity within a protocol, stored in a retrieval system, etc.).

b. Comparing two output strings to determine if they are equivalent, typically through octet-for-octet matching to test for "bit-string identity" (e.g., to make an access decision for purposes of authentication or authorization as further described in [RFC6943]).

The opportunity to define profiles naturally introduces the possibility of a proliferation of profiles, thus potentially mitigating the benefits of common code and violating user expectations. See Section 5 for a discussion of this important topic.

In addition, it is extremely important for protocol designers and application developers to understand that the transformation of an input string to an output string is rarely reversible. As one relatively simple example, case mapping would transform an input string of "StPeter" to "stpeter", and information about the capitalization of the first and third characters would be lost. Similar considerations apply to other forms of mapping and normalization.

Although this framework is similar to IDNA2008 and includes by reference some of the character categories defined in [RFC5892], it defines additional character categories to meet the needs of common application protocols other than DNS.

The character categories and calculation rules defined under Section 8 and Section 9 are normative and apply to all Unicode code points. The code point table that results from applying the character categories and calculation rules to the latest version of Unicode can be found in an IANA registry.

2. Terminology

Many important terms used in this document are defined in [RFC5890], [RFC6365], [RFC6885], and [Unicode7.0]. The terms "left-to-right" (LTR) and "right-to-left" (RTL) are defined in Unicode Standard Annex #9 [UAX9].

As of the date of writing, the version of Unicode published by the Unicode Consortium is 7.0 [Unicode7.0]; however, PRECIS is not tied to a specific version of Unicode. The latest version of Unicode is always available [UnicodeCurrent].

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, Enforcement, and Comparison

This document distinguishes between three different actions that an entity can take with regard to a string:

- Enforcement entails applying all of the rules specified for a particular string class or profile thereof to an individual string, for the purpose of determining if the string can be used in a given protocol slot.

- Comparison entails applying all of the rules specified for a particular string class or profile thereof to two separate strings, for the purpose of determining if the two strings are equivalent.

- Preparation entails only ensuring that the characters in an individual string are allowed by the underlying PRECIS string class.

In most cases, authoritative entities such as servers are responsible for enforcement, whereas subsidiary entities such as clients are responsible only for preparation. The rationale for this distinction is that clients might not have the facilities (in terms of device memory and processing power) to enforce all the rules regarding internationalized strings (such as width mapping and Unicode normalization), although they can more easily limit the repertoire of characters they offer to an end user. By contrast, it is assumed that a server would have more capacity to enforce the rules, and in any case acts as an authority regarding allowable strings in protocol slots such as addresses and endpoint identifiers. In addition, a client cannot necessarily be trusted to properly generate such strings, especially for security-sensitive contexts such as authentication and authorization.

4. String Classes

4.1. Overview

Starting in 2010, various "customers" of Stringprep began to discuss the need to define a post-Stringprep approach to the preparation and comparison of internationalized strings other than IDNs. This community analyzed the existing Stringprep profiles and also weighed the costs and benefits of defining a relatively small set of Unicode

characters that would minimize the potential for user confusion caused by visually similar characters (and thus be relatively "safe") vs. defining a much larger set of Unicode characters that would maximize the potential for user creativity (and thus be relatively "expressive"). As a result, the community concluded that most existing uses could be addressed by two string classes:

IdentifierClass: a sequence of letters, numbers, and some symbols that is used to identify or address a network entity such as a user account, a venue (e.g., a chatroom), an information source (e.g., a data feed), or a collection of data (e.g., a file); the intent is that this class will minimize user confusion in a wide variety of application protocols, with the result that safety has been prioritized over expressiveness for this class.

FreeformClass: a sequence of letters, numbers, symbols, spaces, and other characters that is used for free-form strings, including passwords as well as display elements such as human-friendly nicknames for devices or for participants in a chatroom; the intent is that this class will allow nearly any Unicode character, with the result that expressiveness has been prioritized over safety for this class. Note well that protocol designers, application developers, service providers, and end users might not understand or be able to enter all of the characters that can be included in the FreeformClass - see Section 12.3 for details.

Future specifications might define additional PRECIS string classes, such as a class that falls somewhere between the IdentifierClass and the FreeformClass. At this time, it is not clear how useful such a class would be. In any case, because application developers are able to define profiles of PRECIS string classes, a protocol needing a construct between the IdentiferClass and the FreeformClass could define a restricted profile of the FreeformClass if needed.

The following subsections discuss the IdentifierClass and FreeformClass in more detail, with reference to the dimensions described in Section 3 of [RFC6885]. Each string class is defined by the following behavioral rules:

Valid: Defines which code points are treated as valid for the string.

Contextual Rule Required: Defines which code points are treated as allowed only if the requirements of a contextual rule are met (i.e., either CONTEXTJ or CONTEXTO).

Disallowed: Defines which code points need to be excluded from the string.

Unassigned: Defines application behavior in the presence of code points that are unknown (i.e., not yet designated) for the version of Unicode used by the application.

This document defines the valid, contextual rule required, disallowed, and unassigned rules for the IdentifierClass and FreeformClass. As described under Section 5, profiles of these string classes are responsible for defining the width mapping, additional mappings, case mapping, normalization, and directionality rules.

4.2. IdentifierClass

Most application technologies need strings that can be used to refer to, include, or communicate protocol strings like usernames, file names, data feed identifiers, and chatroom names. We group such strings into a class called "IdentifierClass" having the following features.

4.2.1. Valid

- Code points traditionally used as letters and numbers in writing systems, i.e., the LetterDigits ("A") category first defined in [RFC5892] and listed here under Section 9.1.

- Code points in the range U+0021 through U+007E, i.e., the (printable) ASCII7 ("K") rule defined under Section 9.11. These code points are "grandfathered" into PRECIS and thus are valid even if they would otherwise be disallowed according to the property-based rules specified in the next section.

  Note: Although the PRECIS IdentifierClass re-uses the LetterDigits category from IDNA2008, the range of characters allowed in the IdentifierClass is wider than the range of characters allowed in IDNA2008. The main reason is that IDNA2008 applies the Unstable category before the LetterDigits category, thus disallowing uppercase characters, whereas the IdentifierClass does not apply the Unstable category.

4.2.2. Contextual Rule Required

- A number of characters from the Exceptions ("F") category defined under Section 9.6 (see Section 9.6 for a full list).

- Joining characters, i.e., the JoinControl ("H") category defined under Section 9.8.

4.2.3. Disallowed

- Old Hangul Jamo characters, i.e., the OldHangulJamo ("I") category defined under Section 9.9.
- Control characters, i.e., the Controls ("L") category defined under Section 9.12.
- Ignorable characters, i.e., the PrecisIgnorableProperties ("M") category defined under Section 9.13.
- Space characters, i.e., the Spaces ("N") category defined under Section 9.14.
- Symbol characters, i.e., the Symbols ("O") category defined under Section 9.15.
- Punctuation characters, i.e., the Punctuation ("P") category defined under Section 9.16.
- Any character that has a compatibility equivalent, i.e., the HasCompat ("Q") category defined under Section 9.17. These code points are disallowed even if they would otherwise be valid according to the property-based rules specified in the previous section.
- Letters and digits other than the "traditional" letters and digits allowed in IDNs, i.e., the OtherLetterDigits ("R") category defined under Section 9.18.

4.2.4. Unassigned

Any code points that are not yet designated in the Unicode character set are considered Unassigned for purposes of the IdentifierClass, and such code points are to be treated as Disallowed. See Section 9.10.

4.2.5. Examples

As described in the Introduction to this document, the string classes do not handle all issues related to string preparation and comparison (such as case mapping); instead, such issues are handled at the level of profiles. Examples for two profiles of the IdentifierClass can be found in [I-D.ietf-precis-saslprepbis] (the UsernameIdentifierClass profile) and in [I-D.ietf-xmpp-6122bis] (the LocalpartIdentifierClass profile).

4.3. FreeformClass

Some application technologies need strings that can be used in a free-form way, e.g., as a password in an authentication exchange (see [I-D.ietf-precis-saslprepbis]) or a nickname in a chatroom (see [I-D.ietf-precis-nickname]). We group such things into a class called "FreeformClass" having the following features.

> Security Warning: As mentioned, the FreeformClass prioritizes expressiveness over safety; Section 12.3 describes some of the security hazards involved with using or profiling the FreeformClass.
>
> Security Warning: Consult Section 12.6 for relevant security considerations when strings conforming to the FreeformClass, or a profile thereof, are used as passwords.

4.3.1. Valid

- Traditional letters and numbers, i.e., the LetterDigits ("A") category first defined in [RFC5892] and listed here under Section 9.1.
- Letters and digits other than the "traditional" letters and digits allowed in IDNs, i.e., the OtherLetterDigits ("R") category defined under Section 9.18.
- Code points in the range U+0021 through U+007E, i.e., the (printable) ASCII7 ("K") rule defined under Section 9.11.
- Any character that has a compatibility equivalent, i.e., the HasCompat ("Q") category defined under Section 9.17.
- Space characters, i.e., the Spaces ("N") category defined under Section 9.14.
- Symbol characters, i.e., the Symbols ("O") category defined under Section 9.15.
- Punctuation characters, i.e., the Punctuation ("P") category defined under Section 9.16.

4.3.2. Contextual Rule Required

- A number of characters from the Exceptions ("F") category defined under Section 9.6 (see Section 9.6 for a full list).

- Joining characters, i.e., the JoinControl ("H") category defined under Section 9.8.

4.3.3. Disallowed

- Old Hangul Jamo characters, i.e., the OldHangulJamo ("I") category defined under Section 9.9.

- Control characters, i.e., the Controls ("L") category defined under Section 9.12.

- Ignorable characters, i.e., the PrecisIgnorableProperties ("M") category defined under Section 9.13.

4.3.4. Unassigned

Any code points that are not yet designated in the Unicode character set are considered Unassigned for purposes of the FreeformClass, and such code points are to be treated as Disallowed.

4.3.5. Examples

As described in the Introduction to this document, the string classes do not handle all issues related to string preparation and comparison (such as case mapping); instead, such issues are handled at the level of profiles. Examples for two profiles of the FreeformClass can be found in [I-D.ietf-precis-nickname] (the NicknameFreeformClass profile) and in [I-D.ietf-xmpp-6122bis] (the ResourcepartIdentifierClass profile).

5. Profiles

This framework document defines the valid, contextual-rule-required, disallowed, and unassigned rules for the IdentifierClass and the FreeformClass. A profile of a PRECIS string class MUST define the width mapping, additional mappings (if any), case mapping, normalization, and directionality rules. A profile MAY also restrict the allowable characters above and beyond the definition of the relevant PRECIS string class (but MUST NOT add as valid any code points that are disallowed by the relevant PRECIS string class). These matters are discussed in the following subsections.

Profiles of the PRECIS string classes are registered with the IANA as described under Section 11.3. Profile names use the following convention: they are of the form "Profilename of BaseClass", where the "Profilename" string is a differentiator and "BaseClass" is the name of the PRECIS string class being profiled; for example, the

profile of the Freeform used for opaque strings such as passwords is the "OpaqueString" profile [I-D.ietf-precis-saslprepbis].

5.1. Profiles Must Not Be Multiplied Beyond Necessity

The risk of profile proliferation is significant because having too many profiles will result in different behavior across various applications, thus violating what is known in user interface design as the Principle of Least Astonishment.

Indeed, we already have too many profiles. Ideally we would have at most two or three profiles. Unfortunately, numerous application protocols exist with their own quirks regarding protocol strings. Domain names, email addresses, instant messaging addresses, chatroom nicknames, filenames, authentication identifiers, passwords, and other strings are already out there in the wild and need to be supported in existing application protocols such as DNS, SMTP, XMPP, IRC, NFS, iSCSI, EAP, and SASL among others.

Nevertheless, profiles must not be multiplied beyond necessity.

To help prevent profile proliferation, this document recommends sensible defaults for the various options offered to profile creators (such as width mapping and Unicode normalization). In addition, the guidelines for designated experts provided under Section 10 are meant to encourage a high level of due diligence regarding new profiles.

5.2. Rules

5.2.1. Width Mapping Rule

The width mapping rule of a profile specifies whether width mapping is performed on the characters of a string, and how the mapping is done. Typically such mapping consists of mapping fullwidth and halfwidth characters, i.e., code points with a Decomposition Type of Wide or Narrow, to their decomposition mappings; as an example, FULLWIDTH DIGIT ZERO (U+FF10) would be mapped to DIGIT ZERO (U+0030).

The normalization form specified by a profile (see below) has an impact on the need for width mapping. Because width mapping is performed as a part of compatibility decomposition, a profile employing either normalization form KD (NFKD) or normalization form KC (NFKC) does not need to specify width mapping. However, if Unicode normalization form C (NFC) is used (as is recommended) then the profile needs to specify whether to apply width mapping; in this case, width mapping is in general RECOMMENDED because allowing fullwidth and halfwidth characters to remain unmapped to their compatibility variants would violate the Principle of Least

Astonishment. For more information about the concept of width in East Asian scripts within Unicode, see Unicode Standard Annex #11 [UAX11].

5.2.2. Additional Mapping Rule

The additional mapping rule of a profile specifies whether additional mappings is performed on the characters of a string, such as:

Mapping of delimiter characters (such as '@', ':', '/', '+', and '-')

Mapping of special characters (e.g., non-ASCII space characters to ASCII space or control characters to nothing).

The PRECIS mappings document [I-D.ietf-precis-mappings] describes such mappings in more detail.

5.2.3. Case Mapping Rule

The case mapping rule of a profile specifies whether case mapping (instead of case preservation) is performed on the characters of a string, and how the mapping is applied (e.g., mapping uppercase and titlecase characters to their lowercase equivalents).

If case mapping is desired (instead of case preservation), it is RECOMMENDED to use Unicode Default Case Folding as defined in Chapter 3 of the Unicode Standard [Unicode7.0].

Note: Unicode Default Case Folding is not designed to handle various localization issues (such as so-called "dotless i" in several Turkic languages). The PRECIS mappings document [I-D.ietf-precis-mappings] describes these issues in greater detail and defines a "local case mapping" method that handles some locale-dependent and context-dependent mappings.

In order to maximize entropy and minimize the potential for false positives, it is NOT RECOMMENDED for application protocols to map uppercase and titlecase code points to their lowercase equivalents when strings conforming to the FreeformClass, or a profile thereof, are used in passwords; instead, it is RECOMMENDED to preserve the case of all code points contained in such strings and then perform case-sensitive comparison. See also the related discussion in [I-D.ietf-precis-saslprepbis].

5.2.4. Normalization Rule

The normalization rule of a profile specifies which Unicode normalization form (D, KD, C, or KC) is to be applied (see Unicode Standard Annex #15 [UAX15] for background information).

In accordance with [RFC5198], normalization form C (NFC) is RECOMMENDED.

5.2.5. Directionality Rule

The directionality rule of a profile specifies how to treat strings containing what are often called "right-to-left" (RTL) characters (see Unicode Standard Annex #9 [UAX9]). RTL characters come from scripts that are normally written from right to left and are considered by Unicode to, themselves, have right-to-left directionality. Some strings containing RTL characters also contain "left-to-right" (LTR) characters, such as numerals, as well as characters without directional properties. Consequently, such strings are known as "bidirectional strings".

Presenting bidirectional strings in different layout systems (e.g., a user interface that is configured to handle primarily an RTL script vs. an interface that is configured to handle primarily an LTR script) can yield display results that, while predictable to those who understand the display rules, are counter-intuitive to casual users. In particular, the same bidirectional string (in PRECIS terms) might not be presented in the same way to users of those different layout systems, even though the presentation is consistent within any particular layout system. In some applications, these presentation differences might be considered problematic and thus the application designers might wish to restrict the use of bidirectional strings by specifying a directionality rule. In other applications, these presentation differences might not be considered problematic (this especially tends to be true of more "free-form" strings) and thus no directionality rule is needed.

The PRECIS framework does not directly address how to deal with bidirectional strings across all string classes and profiles, and does not define any new directionality rules, since at present there is no widely accepted and implemented solution for the safe display of arbitrary bidirectional strings beyond the Unicode bidirectional algorithm [UAX9]. Although rules for management and display of bidirectional strings have been defined for domain name labels and similar identifiers through the "Bidi Rule" specified in the IDNA2008 specification on right-to-left scripts [RFC5893], those rules are quite restrictive and are not necessarily applicable to all bidirectional strings.

The authors of a PRECIS profile might believe that they need to define a new directionality rule of their own. Because of the complexity of the issues involved, such a belief is almost always misguided, even if the authors have done a great deal of careful research into the challenges of displaying bidirectional strings. This document strongly suggests that profile authors who are thinking about defining a new directionality rule think again, and instead consider using the "Bidi Rule" [RFC5893] (for profiles based on the IdentifierClass) or following the Unicode bidirectional algorithm [UAX9] (for profiles based on the FreeformClass or in situations where the IdentifierClass is not appropriate).

5.3. A Note about Spaces

With regard to the IdentiferClass, the consensus of the PRECIS Working Group was that spaces are problematic for many reasons, including:

- Many Unicode characters are confusable with ASCII space.
- Even if non-ASCII space characters are mapped to ASCII space (U+0020), space characters are often not rendered in user interfaces, leading to the possibility that a human user might consider a string containing spaces to be equivalent to the same string without spaces.
- In some locales, some devices are known to generate a character other than ASCII space (such as ZERO WIDTH JOINER, U+200D) when a user performs an action like hitting the space bar on a keyboard.

One consequence of disallowing space characters in the IdentifierClass might be to effectively discourage their use within identifiers created in newer application protocols; given the challenges involved with properly handling space characters (especially non-ASCII space characters) in identifiers and other protocol strings, the PRECIS Working Group considered this to be a feature, not a bug.

However, the FreeformClass does allow spaces, which enables application protocols to define profiles of the FreeformClass that are more flexible than any profiles of the IdentifierClass. In addition, as explained in the previous section, application protocols can also define application-layer constructs containing spaces.

6. Applications

6.1. How to Use PRECIS in Applications

Although PRECIS has been designed with applications in mind, internationalization is not suddenly made easy though the use of PRECIS. Application developers still need to give some thought to how they will use the PRECIS string classes, or profiles thereof, in their applications. This section provides some guidelines to application developers (and to expert reviewers of application protocol specifications).

- Don't define your own profile unless absolutely necessary (see Section 5.1). Existing profiles have been design for wide re-use. It is highly likely that an existing profile will meet your needs, especially given the ability to specify further excluded characters (Section 6.2) and to build application-layer constructs (see Section 6.3).

- Do specify:

  * Exactly which entities are responsible for preparation, enforcement, and comparison of internationalized strings (e.g., servers or clients).

  * Exactly when those entities need to complete their tasks (e.g., a server might need to enforce the rules of a profile before allowing a client to gain network access).

  * Exactly which protocol slots need to be checked against which profiles (e.g., checking the address of a message's intended recipient against the UsernameCaseMapped profile [I-D.ietf-precis-saslprepbis] of the IdentifierClass, or checking the password of a user against the OpaqueString profile [I-D.ietf-precis-saslprepbis] of the FreeformClass).

  See [I-D.ietf-precis-saslprepbis] and [I-D.ietf-xmpp-6122bis] for definitions of these matters for several applications.

6.2. Further Excluded Characters

An application protocol that uses a profile MAY specify particular code points that are not allowed in relevant slots within that application protocol, above and beyond those excluded by the string class or profile.

That is, an application protocol MAY do either of the following:

1. Exclude specific code points that are allowed by the relevant string class.

2. Exclude characters matching certain Unicode properties (e.g., math symbols) that are included in the relevant PRECIS string class.

As a result of such exclusions, code points that are defined as valid for the PRECIS string class or profile will be defined as disallowed for the relevant protocol slot.

Typically, such exclusions are defined for the purpose of backward-compatibility with legacy formats within an application protocol. These are defined for application protocols, not profiles, in order to prevent multiplication of profiles beyond necessity (see Section 5.1).

6.3. Building Application-Layer Constructs

Sometimes, an application-layer construct does not map in a straightforward manner to one of the base string classes or a profile thereof. Consider, for example, the "simple user name" construct in the Simple Authentication and Security Layer (SASL) [RFC4422]. Depending on the deployment, a simple user name might take the form of a user's full name (e.g., the user's personal name followed by a space and then the user's family name). Such a simple user name cannot be defined as an instance of the IdentifierClass or a profile thereof, since space characters are not allowed in the IdentifierClass; however, it could be defined using a space-separated sequence of IdentifierClass instances, as in the following ABNF [RFC5234] from [I-D.ietf-precis-saslprepbis]:

```
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"
             ;
```

Similar techniques could be used to define many application-layer constructs, say of the form "user@domain" or "/path/to/file".

7. Order of Operations

To ensure proper comparison, the rules specified for a particular string class or profile MUST be applied in the following order:

1. Width Mapping Rule
2. Additional Mapping Rule
3. Case Mapping Rule
4. Normalization Rule
5. Directionality Rule
6. Behavioral rules for determining whether a code point is valid, allowed under a contextual rule, disallowed, or unassigned

As already described, the width mapping, additional mapping, case mapping, normalization, and directionality rules are specified for each profile, whereas the behavioral rules are specified for each string class. Some of the logic behind this order is provided under Section 5.2.1 (see also the PRECIS mappings document [I-D.ietf-precis-mappings]).

8. Code Point Properties

In order to implement the string classes described above, this document does the following:

1. Reviews and classifies the collections of code points in the Unicode character set by examining various code point properties.
2. Defines an algorithm for determining a derived property value, which can vary depending on the string class being used by the relevant application protocol.

This document is not intended to specify precisely how derived property values are to be applied in protocol strings. That information is the responsibility of the protocol specification that uses or profiles a PRECIS string class from this document. The value of the property is to be interpreted as follows.

PROTOCOL VALID Those code points that are allowed to be used in any PRECIS string class (currently, IdentifierClass and FreeformClass). The abbreviated term "PVALID" is used to refer to this value in the remainder of this document.

SPECIFIC CLASS PROTOCOL VALID Those code points that are allowed to be used in specific string classes. In the remainder of this document, the abbreviated term *_PVAL is used, where * = (ID | FREE), i.e., either "FREE_PVAL" or "ID_PVAL". In practice, the derived property ID_PVAL is not used in this specification, since every ID_PVAL code point is PVALID.

CONTEXTUAL RULE REQUIRED Some characteristics of the character, such as its being invisible in certain contexts or problematic in others, require that it not be used in labels unless specific other characters or properties are present. As in IDNA2008, there are two subdivisions of CONTEXTUAL RULE REQUIRED, the first for Join_controls (called "CONTEXTJ") and the second for other characters (called "CONTEXTO"). A character with the derived property value CONTEXTJ or CONTEXTO MUST NOT be used unless an appropriate rule has been established and the context of the character is consistent with that rule. The most notable of the CONTEXTUAL RULE REQUIRED characters are the Join Control characters U+200D ZERO WIDTH JOINER and U+200C ZERO WIDTH NON-JOINER, which have a derived property value of CONTEXTJ. See Appendix A of [RFC5892] for more information.

DISALLOWED Those code points that are not permitted in any PRECIS string class.

SPECIFIC CLASS DISALLOWED Those code points that are not to be included in one of the string classes but that might be permitted in others. In the remainder of this document, the abbreviated term *_DIS is used, where * = (ID | FREE), i.e., either "FREE_DIS" or "ID_DIS". In practice, the derived property FREE_DIS is not used in this specification, since every FREE_DIS code point is DISALLOWED.

UNASSIGNED Those code points that are not designated (i.e. are unassigned) in the Unicode Standard.

The algorithm to calculate the value of the derived property is as follows (implementations MUST NOT modify the order of operations within this algorithm, since doing so would cause inconsistent results across implementations):

```
If .cp. .in. Exceptions Then Exceptions(cp);
Else If .cp. .in. BackwardCompatible Then BackwardCompatible(cp);
Else If .cp. .in. Unassigned Then UNASSIGNED;
Else If .cp. .in. ASCII7 Then PVALID;
Else If .cp. .in. JoinControl Then CONTEXTJ;
Else If .cp. .in. OldHangulJamo Then DISALLOWED;
Else If .cp. .in. PrecisIgnorableProperties Then DISALLOWED;
Else If .cp. .in. Controls Then DISALLOWED;
Else If .cp. .in. HasCompat Then ID_DIS or FREE_PVAL;
Else If .cp. .in. LetterDigits Then PVALID;
Else If .cp. .in. OtherLetterDigits Then ID_DIS or FREE_PVAL;
Else If .cp. .in. Spaces Then ID_DIS or FREE_PVAL;
Else If .cp. .in. Symbols Then ID_DIS or FREE_PVAL;
Else If .cp. .in. Punctuation Then ID_DIS or FREE_PVAL;
Else DISALLOWED;
```

The value of the derived property calculated can depend on the string class; for example, if an identifier used in an application protocol is defined as profiling the PRECIS IdentifierClass then a space character such as U+0020 would be assigned to ID_DIS, whereas if an identifier is defined as profiling the PRECIS FreeformClass then the character would be assigned to FREE_PVAL. For the sake of brevity, the designation "FREE_PVAL" is used herein, instead of the longer designation "ID_DIS or FREE_PVAL". In practice, the derived properties ID_PVAL and FREE_DIS are not used in this specification, since every ID_PVAL code point is PVALID and every FREE_DIS code point is DISALLOWED.

Use of the name of a rule (such as "Exceptions") implies the set of code points that the rule defines, whereas the same name as a function call (such as "Exceptions(cp)") implies the value that the code point has in the Exceptions table.

The mechanisms described here allow determination of the value of the property for future versions of Unicode (including characters added after Unicode 5.2 or 7.0 depending on the category, since some categories mentioned in this document are simply pointers to IDNA2008 and therefore were defined at the time of Unicode 5.2). Changes in Unicode properties that do not affect the outcome of this process therefore do not affect this framework. For example, a character can have its Unicode General_Category value (see Chapter 4 of the Unicode Standard [Unicode7.0]) change from So to Sm, or from Lo to Ll, without affecting the algorithm results. Moreover, even if such changes were to result, the BackwardCompatible list (Section 9.7) can be adjusted to ensure the stability of the results.

9. Category Definitions Used to Calculate Derived Property

The derived property obtains its value based on a two-step procedure:

1. Characters are placed in one or more character categories either (1) based on core properties defined by the Unicode Standard or (2) by treating the code point as an exception and addressing the code point based on its code point value. These categories are not mutually exclusive.

2. Set operations are used with these categories to determine the values for a property specific to a given string class. These operations are specified under Section 8.

   Note: Unicode property names and property value names might have short abbreviations, such as "gc" for the General_Category property and "Ll" for the Lowercase_Letter property value of the gc property.

In the following specification of character categories, the operation that returns the value of a particular Unicode character property for a code point is designated by using the formal name of that property (from the Unicode PropertyAliases.txt [1]) followed by '(cp)' for "code point". For example, the value of the General_Category property for a code point is indicated by General_Category(cp).

The first ten categories (A-J) shown below were previously defined for IDNA2008 and are referenced from [RFC5892] to ease the understanding of how PRECIS handles various characters. Some of these categories are reused in PRECIS and some of them are not; however, the lettering of categories is retained to prevent overlap and to ease implementation of both IDNA2008 and PRECIS in a single software application. The next eight categories (K-R) are specific to PRECIS.

9.1. LetterDigits (A)

This category is defined in Section 2.1 of [RFC5892] and is included by reference for use in PRECIS.

9.2. Unstable (B)

This category is defined in Section 2.2 of [RFC5892]. However, it is not used in PRECIS.

9.3. IgnorableProperties (C)

This category is defined in Section 2.3 of [RFC5892]. However, it is not used in PRECIS.

Note: See the "PrecisIgnorableProperties (M)" category below for a more inclusive category used in PRECIS identifiers.

9.4. IgnorableBlocks (D)

This category is defined in Section 2.4 of [RFC5892]. However, it is not used in PRECIS.

9.5. LDH (E)

This category is defined in Section 2.5 of [RFC5892]. However, it is not used in PRECIS.

Note: See the "ASCII7 (K)" category below for a more inclusive category used in PRECIS identifiers.

9.6. Exceptions (F)

This category is defined in Section 2.6 of [RFC5892] and is included by reference for use in PRECIS.

9.7. BackwardCompatible (G)

This category is defined in Section 2.7 of [RFC5892] and is included by reference for use in PRECIS.

Note: Management of this category is handled via the processes specified in [RFC5892]. At the time of this writing (and also at the time that RFC 5892 was published), this category consisted of the empty set; however, that is subject to change as described in RFC 5892.

9.8. JoinControl (H)

This category is defined in Section 2.8 of [RFC5892] and is included by reference for use in PRECIS.

9.9. OldHangulJamo (I)

This category is defined in Section 2.9 of [RFC5892] and is included by reference for use in PRECIS.

9.10. Unassigned (J)

This category is defined in Section 2.10 of [RFC5892] and is included by reference for use in PRECIS.

9.11. ASCII7 (K)

This PRECIS-specific category consists of all printable, non-space characters from the 7-bit ASCII range. By applying this category, the algorithm specified under Section 8 exempts these characters from other rules that might be applied during PRECIS processing, on the assumption that these code points are in such wide use that disallowing them would be counter-productive.

K: cp is in {0021..007E}

9.12. Controls (L)

This PRECIS-specific category consists of all control characters.

L: Control(cp) = True

9.13. PrecisIgnorableProperties (M)

This PRECIS-specific category is used to group code points that are discouraged from use in PRECIS string classes.

M: Default_Ignorable_Code_Point(cp) = True or
Noncharacter_Code_Point(cp) = True

The definition for Default_Ignorable_Code_Point can be found in the DerivedCoreProperties.txt [2] file.

9.14. Spaces (N)

This PRECIS-specific category is used to group code points that are space characters.

N: General_Category(cp) is in {Zs}

9.15. Symbols (O)

This PRECIS-specific category is used to group code points that are symbols.

O: General_Category(cp) is in {Sm, Sc, Sk, So}

9.16. Punctuation (P)

This PRECIS-specific category is used to group code points that are punctuation characters.

P: General_Category(cp) is in {Pc, Pd, Ps, Pe, Pi, Pf, Po}

9.17. HasCompat (Q)

This PRECIS-specific category is used to group code points that have compatibility equivalents as explained in Chapter 2 and Chapter 3 of the Unicode Standard [Unicode7.0].

Q: toNFKC(cp) != cp

The toNFKC() operation returns the code point in normalization form KC. For more information, see Section 5 of Unicode Standard Annex #15 [UAX15].

9.18. OtherLetterDigits (R)

This PRECIS-specific category is used to group code points that are letters and digits other than the "traditional" letters and digits grouped under the LetterDigits (A) class (see Section 9.1).

R: General_Category(cp) is in {Lt, Nl, No, Me}

10. Guidelines for Designated Experts

Experience with internationalization in application protocols has shown that protocol designers and application developers usually do not understand the subtleties and tradeoffs involved with internationalization and that they need considerable guidance in making reasonable decisions with regard to the options before them.

Therefore:

- Protocol designers are strongly encouraged to question the assumption that they need to define new profiles, since existing profiles are designed for wide re-use (see Section 5 for further discussion).

- Those who persist in defining new profiles are strongly encouraged to clearly explain a strong justification for doing so, and to publish a stable specification that provides all of the information described under Section 11.3.

- The designated experts for profile registration requests ought to seek answers to all of the questions provided under Section 11.3 and to encourage applicants to provide a stable specification documenting the profile (even though the registration policy for PRECIS profiles is Expert Review and a stable specification is not strictly required).

- Developers of applications that use PRECIS are strongly encouraged to apply the guidelines provided under Section 6 and to seek out the advice of the designated experts or other knowledgeable individuals in doing so.

- All parties are strongly encouraged to help prevent the multiplication of profiles beyond necessity, as described under Section 5.1, and to use PRECIS in ways that will minimize user confusion and insecure application behavior.

Internationalization can be difficult and contentious; designated experts, profile registrants, and application developers are strongly encouraged to work together in a spirit of good faith and mutual understanding to achieve rough consensus on profile registration requests and the use of PRECIS in particular applications. They are also encouraged to bring additional expertise into the discussion if that would be helpful in adding perspective or otherwise resolving issues.

11. IANA Considerations

11.1. PRECIS Derived Property Value Registry

IANA is requested to create a PRECIS-specific registry with the Derived Properties for the versions of Unicode that are released after (and including) version 7.0. The derived property value is to be calculated in cooperation with a designated expert [RFC5226] according to the rules specified under Section 8 and Section 9.

The IESG is to be notified if backward-incompatible changes to the table of derived properties are discovered or if other problems arise during the process of creating the table of derived property values or during expert review. Changes to the rules defined under Section 8 and Section 9 require IETF Review.

11.2. PRECIS Base Classes Registry

IANA is requested to create a registry of PRECIS string classes. In accordance with [RFC5226], the registration policy is "RFC Required".

The registration template is as follows:

Base Class: [the name of the PRECIS string class]

Description: [a brief description of the PRECIS string class and its intended use, e.g., "A sequence of letters, numbers, and symbols that is used to identify or address a network entity."]

Specification: [the RFC number]

The initial registrations are as follows:

Base Class: FreeformClass.
Description: A sequence of letters, numbers, symbols, spaces, and other code points that is used for free-form strings.
Specification: Section 4.3 of this document.
[Note to RFC Editor: please change "this document" to the RFC number issued for this specification.]

Base Class: IdentifierClass.
Description: A sequence of letters, numbers, and symbols that is used to identify or address a network entity.
Specification: Section 4.2 of this document.
[Note to RFC Editor: please change "this document" to the RFC number issued for this specification.]

11.3. PRECIS Profiles Registry

IANA is requested to create a registry of profiles that use the PRECIS string classes. In accordance with [RFC5226], the registration policy is "Expert Review". This policy was chosen in order to ease the burden of registration while ensuring that "customers" of PRECIS receive appropriate guidance regarding the sometimes complex and subtle internationalization issues related to profiles of PRECIS string classes.

The registration template is as follows:

Name: [the name of the profile]

Base Class: [which PRECIS string class is being profiled]

Applicability: [the specific protocol elements to which this profile applies, e.g., "Localparts in XMPP addresses."]

Replaces: [the Stringprep profile that this PRECIS profile replaces, if any]

Width Mapping Rule: [the behavioral rule for handling of width, e.g., "Map fullwidth and halfwidth characters to their compatibility variants."]

Additional Mapping Rule: [any additional mappings are required or recommended, e.g., "Map non-ASCII space characters to ASCII space."]

Case Mapping Rule: [the behavioral rule for handling of case, e.g., "Unicode Default Case Folding"]

Normalization Rule: [which Unicode normalization form is applied, e.g., "NFC"]

Directionality Rule: [the behavioral rule for handling of right-to-left code points, e.g., "The 'Bidi Rule' defined in RFC 5893 applies."]

Enforcement: [which entities enforce the rules, and when that enforcement occurs during protocol operations]

Specification: [a pointer to relevant documentation, such as an RFC or Internet-Draft]

In order to request a review, the registrant shall send a completed template to the precis@ietf.org list or its designated successor.

Factors to focus on while defining profiles and reviewing profile registrations include the following:

- Would an existing PRECIS string class or profile solve the problem? If not, why not? (See Section 5.1 for related considerations.)
- Is the problem being addressed by this profile well-defined?
- Does the specification define what kinds of applications are involved and the protocol elements to which this profile applies?
- Is the profile clearly defined?
- Is the profile based on an appropriate dividing line between user interface (culture, context, intent, locale, device limitations, etc.) and the use of conformant strings in protocol elements?
- Are the width mapping, case mapping, additional mappings, normalization, and directionality rules appropriate for the intended use?

- Does the profile explain which entities enforce the rules, and when such enforcement occurs during protocol operations?

- Does the profile reduce the degree to which human users could be surprised or confused by application behavior (the "Principle of Least Astonishment")?

- Does the profile introduce any new security concerns such as those described under Section 12 of this document (e.g., false positives for authentication or authorization)?

12. Security Considerations

12.1. General Issues

If input strings that appear "the same" to users are programmatically considered to be distinct in different systems, or if input strings that appear distinct to users are programmatically considered to be "the same" in different systems, then users can be confused. Such confusion can have security implications, such as the false positives and false negatieves discussed in [RFC6943]. One starting goal of work on the PRECIS framework was to limit the number of times that users are confused (consistent with the "Principle of Least Astonishment"). Unfortunately, this goal has been difficult to achieve given the large number of application protocols already in existence. Despite these difficulties, profiles should not be multiplied beyond necessity (see Section 5.1. In particular, application protocol designers should think long and hard before defining a new profile instead of using one that has already been defined, and if they decide to define a new profile then they should clearly explain their reasons for doing so.

The security of applications that use this framework can depend in part on the proper preparation, enforcement, and comparison of internationalized strings. For example, such strings can be used to make authentication and authorization decisions, and the security of an application could be compromised if an entity providing a given string is connected to the wrong account or online resource based on different interpretations of the string (again, see [RFC6943]).

Specifications of application protocols that use this framework are strongly encouraged to describe how internationalized strings are used in the protocol, including the security implications of any false positives and false negatives that might result from various enforcement and comparison operations. For some helpful guidelines, refer to [RFC6943], [RFC5890], [UTR36], and [UTS39].

12.2. Use of the IdentifierClass

Strings that conform to the IdentifierClass and any profile thereof are intended to be relatively safe for use in a broad range of applications, primarily because they include only letters, digits, and "grandfathered" non-space characters from the ASCII range; thus they exclude spaces, characters with compatibility equivalents, and almost all symbols and punctuation marks. However, because such strings can still include so-called confusable characters (see Section 12.5), protocol designers and implementers are encouraged to pay close attention to the security considerations described elsewhere in this document.

12.3. Use of the FreeformClass

Strings that conform to the FreeformClass and many profiles thereof can include virtually any Unicode character. This makes the FreeformClass quite expressive, but also problematic from the perspective of possible user confusion. Protocol designers are hereby warned that the FreeformClass contains codepoints they might not understand, and are encouraged to profile the IdentifierClass wherever feasible; however, if an application protocol requires more code points than are allowed by the IdentifierClass, protocol designers are encouraged to define a profile of the FreeformClass that restricts the allowable code points as tightly as possible. (The PRECIS Working Group considered the option of allowing "superclasses" as well as profiles of PRECIS string classes, but decided against allowing superclasses to reduce the likelihood of security and interoperability problems.)

12.4. Local Character Set Issues

When systems use local character sets other than ASCII and Unicode, this specification leaves the problem of converting between the local character set and Unicode up to the application or local system. If different applications (or different versions of one application) implement different rules for conversions among coded character sets, they could interpret the same name differently and contact different application servers or other network entities. This problem is not solved by security protocols, such as Transport Layer Security (TLS) [RFC5246] and the Simple Authentication and Security Layer (SASL) [RFC4422], that do not take local character sets into account.

12.5. Visually Similar Characters

Some characters are visually similar and thus can cause confusion among humans. Such characters are often called "confusable characters" or "confusables".

The problem of confusable characters is not necessarily caused by the use of Unicode code points outside the ASCII range. For example, in some presentations and to some individuals the string "ju1iet" (spelled with DIGIT ONE, U+0031, as the third character) might appear to be the same as "juliet" (spelled with LATIN SMALL LETTER L, U+006C), especially on casual visual inspection. This phenomenon is sometimes called "typejacking".

However, the problem is made more serious by introducing the full range of Unicode code points into protocol strings. For example, the characters U+13DA U+13A2 U+13B5 U+13AC U+13A2 U+13AC U+13D2 from the Cherokee block look similar to the ASCII characters "STPETER" as they might appear when presented using a "creative" font family.

In some examples of confusable characters, it is unlikely that the average human could tell the difference between the real string and the fake string. (Indeed, there is no programmatic way to distinguish with full certainty which is the fake string and which is the real string; in some contexts, the string formed of Cherokee characters might be the real string and the string formed of ASCII characters might be the fake string.) Because PRECIS-compliant strings can contain almost any properly-encoded Unicode code point, it can be relatively easy to fake or mimic some strings in systems that use the PRECIS framework. The fact that some strings are easily confused introduces security vulnerabilities of the kind that have also plagued the World Wide Web, specifically the phenomenon known as phishing.

Despite the fact that some specific suggestions about identification and handling of confusable characters appear in the Unicode Security Considerations [UTR36] and the Unicode Security Mechanisms [UTS39], it is also true (as noted in [RFC5890]) that "there are no comprehensive technical solutions to the problems of confusable characters". Because it is impossible to map visually similar characters without a great deal of context (such as knowing the font families used), the PRECIS framework does nothing to map similar-looking characters together, nor does it prohibit some characters because they look like others.

Nevertheless, specifications for application protocols that use this framework are strongly encouraged to describe how confusable characters can be abused to compromise the security of systems that use the protocol in question, along with any protocol-specific suggestions for overcoming those threats. In particular, software implementations and service deployments that use PRECIS-based technologies are strongly encouraged to define and implement consistent policies regarding the registration, storage, and

presentation of visually similar characters. The following recommendations are appropriate:

1. An application service SHOULD define a policy that specifies the scripts or blocks of characters that the service will allow to be registered (e.g., in an account name) or stored (e.g., in a file name). Such a policy SHOULD be informed by the languages and scripts that are used to write registered account names; in particular, to reduce confusion, the service SHOULD forbid registration or storage of strings that contain characters from more than one script and SHOULD restrict registrations to characters drawn from a very small number of scripts (e.g., scripts that are well-understood by the administrators of the service, to improve manageability).

2. User-oriented application software SHOULD define a policy that specifies how internationalized strings will be presented to a human user. Because every human user of such software has a preferred language or a small set of preferred languages, the software SHOULD gather that information either explicitly from the user or implicitly via the operating system of the user's device. Furthermore, because most languages are typically represented by a single script or a small set of scripts, and because most scripts are typically contained in one or more blocks of characters, the software SHOULD warn the user when presenting a string that mixes characters from more than one script or block, or that uses characters outside the normal range of the user's preferred language(s). (Such a recommendation is not intended to discourage communication across different communities of language users; instead, it recognizes the existence of such communities and encourages due caution when presenting unfamiliar scripts or characters to human users.)

The challenges inherent in supporting the full range of Unicode code points have in the past led some to hope for a way to programmatically negotiate more restrictive ranges based on locale, script, or other relevant factors, to tag the locale associated with a particular string, etc. As a general-purpose internationalization technology, the PRECIS framework does not include such mechanisms.

12.6. Security of Passwords

Two goals of passwords are to maximize the amount of entropy and to minimize the potential for false positives. These goals can be achieved in part by allowing a wide range of code points and by ensuring that passwords are handled in such a way that code points are not compared aggressively. Therefore, it is NOT RECOMMENDED for application protocols to profile the FreeformClass for use in

passwords in a way that removes entire categories (e.g., by disallowing symbols or punctuation). Furthermore, it is NOT RECOMMENDED for application protocols to map uppercase and titlecase code points to their lowercase equivalents in such strings; instead, it is RECOMMENDED to preserve the case of all code points contained in such strings and to compare them in a case-sensitive manner.

That said, software implementers need to be aware that there exist tradeoffs between entropy and usability. For example, allowing a user to establish a password containing "uncommon" code points might make it difficult for the user to access a service when using an unfamiliar or constrained input device.

Some application protocols use passwords directly, whereas others reuse technologies that themselves process passwords (one example of such a technology is the Simple Authentication and Security Layer [RFC4422]). Moreover, passwords are often carried by a sequence of protocols with backend authentication systems or data storage systems such as RADIUS [RFC2865] and LDAP [RFC4510]. Developers of application protocols are encouraged to look into reusing these profiles instead of defining new ones, so that end-user expectations about passwords are consistent no matter which application protocol is used.

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.

Further discussion of password handling can be found in [I-D.ietf-precis-saslprepbis].

13. Interoperability Considerations

13.1. Encoding

Although strings that are consumed in PRECIS-based application protocols are often encoded using UTF-8 [RFC3629], the exact encoding is a matter for the application protocol that uses PRECIS, not for the PRECIS framework.

13.2. Character Sets

It is known that some existing systems are unable to support the full Unicode character set, or even any characters outside the ASCII range. If two (or more) applications need to interoperate when exchanging data (e.g., for the purpose of authenticating a username or password), they will naturally need to have in common at least one

coded character set (as defined by [RFC6365]). Establishing such a baseline is a matter for the application protocol that uses PRECIS, not for the PRECIS framework.

13.3. Unicode Versions

Changes to the properties of Unicode code points can occur as the Unicode Standard is modified from time to time. For example, three code points underwent changes in their GeneralCategory between Unicode 5.2 (current at the time IDNA2008 was originally published) and Unicode 6.0, as described in [RFC6452]. Implementers might need to be aware that the treatment of these characters differs depending on which version of Unicode is available on the system that is using IDNA2008 or PRECIS. Other such differences might arise between the version of Unicode current at the time of this writing (7.0) and future versions.

13.4. Potential Changes to Handling of Certain Unicode Code Points

As part of the review of Unicode 7.0 for IDNA, a question was raised about a newly-added code point that led to a re-analysis of the Normalization Rules used by IDNA and inherited by this document (Section 5.2.4). Some of the general issues are described in [IAB-Statement] and pursued in more detail in [I-D.klensin-idna-5892upd-unicode70].

At the time of writing, these issues have yet to be settled. However, implementers need to be aware that this specification is likely to be updated in the future to address these issues. The potential changes include:

- The range of characters in the LetterDigits category (Section 4.2.1 and Section 9.1) might be narrowed.
- Some characters with special properties that are now allowed might be excluded.
- More "Additional Mapping Rules" (Section 5.2.2) might be defined.
- Alternative normalization methods might be added.

Nevertheless, implementations and deployments that are sensitive to the advice given in this specification are unlikely to run into significant problems as a consequence of these issues or potential changes - specifically the advice to use the more restrictive IdentifierClass whenever possible, or if using the FreeformClass to allow only a restricted set of characters, particularly avoiding characters whose implications they do not actually understand.

14. References

14.1. Normative References

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

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

[RFC5198] Klensin, J. and M. Padlipsky, "Unicode Format for Network Interchange", RFC 5198, March 2008.

[Unicode7.0] The Unicode Consortium, "The Unicode Standard, Version 7.0.0", 2014, <http://www.unicode.org/versions/Unicode7.0.0/>.

14.2. Informative References

[IAB-Statement] Internet Architecture Board, "IAB Statement on Identifiers and Unicode 7.0.0", January 2015, <https://www.iab.org/documents/correspondence-reports-documents/2015-2/iab-statement-on-identifiers-and-unicode-7-0-0/>.

[I-D.ietf-precis-mappings] Yoneya, Y. and T. NEMOTO, "Mapping characters for PRECIS classes", draft-ietf-precis-mappings-08 (work in progress), June 2014.

[I-D.ietf-precis-nickname] Saint-Andre, P., "Preparation and Comparison of Nicknames", draft-ietf-precis-nickname-14 (work in progress), December 2014.

[I-D.ietf-precis-saslprepbis] Saint-Andre, P. and A. Melnikov, "Username and Password Preparation Algorithms", draft-ietf-precis-saslprepbis-13 (work in progress), December 2014.

[I-D.ietf-xmpp-6122bis] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Address Format", draft-ietf-xmpp-6122bis-18 (work in progress), December 2014.

[I-D.klensin-idna-5892upd-unicode70]
Klensin, J. and P. Faeltstroem, "IDNA Update for Unicode 7.0.0", draft-klensin-idna-5892upd-unicode70-03 (work in progress), January 2015.

[RFC2865] Rigney, C., Willens, S., Rubens, A., and W. Simpson, "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[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.

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

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

[RFC4510] Zeilenga, K., "Lightweight Directory Access Protocol (LDAP): Technical Specification Road Map", RFC 4510, June 2006.

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

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

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

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[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.

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

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

[RFC6885] Blanchet, M. and A. Sullivan, "Stringprep Revision and Problem Statement for the Preparation and Comparison of Internationalized Strings (PRECIS)", RFC 6885, March 2013.

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

[UAX9] The Unicode Consortium, "Unicode Standard Annex #9: Unicode Bidirectional Algorithm", September 2012, <http://unicode.org/reports/tr9/>.

[UAX11] The Unicode Consortium, "Unicode Standard Annex #11: East Asian Width", September 2012, <http://unicode.org/reports/tr11/>.

[UAX15] The Unicode Consortium, "Unicode Standard Annex #15: Unicode Normalization Forms", August 2012, <http://unicode.org/reports/tr15/>.

[UnicodeCurrent]
The Unicode Consortium, "The Unicode Standard", 2014-present, <http://www.unicode.org/versions/latest/>.

[UTR36] The Unicode Consortium, "Unicode Technical Report #36: Unicode Security Considerations", July 2012, <http://unicode.org/reports/tr36/>.

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

14.3. URIs

[1] http://unicode.org/Public/UNIDATA/PropertyAliases.txt

[2] http://unicode.org/Public/UNIDATA/DerivedCoreProperties.txt

Appendix A. Acknowledgements

The authors would like to acknowledge the comments and contributions of the following individuals during working group discussion: David Black, Edward Burns, Dan Chiba, Mark Davis, Alan DeKok, Martin Duerst, Patrik Faltstrom, Ted Hardie, Joe Hildebrand, Bjoern Hoehrmann, Paul Hoffman, Jeffrey Hutzelman, Simon Josefsson, John Klensin, Alexey Melnikov, Takahiro Nemoto, Yoav Nir, Mike Parker, Pete Resnick, Andrew Sullivan, Dave Thaler, Yoshiro Yoneya, and Florian Zeitz.

Special thanks are due to John Klensin and Patrik Faltstrom for their challenging feedback and detailed reviews.

Charlie Kaufman, Tom Taylor, and Tim Wicinski reviewed the document on behalf of the Security Directorate, the General Area Review Team, and the Operations and Management Directorate, respectively.

During IESG review, Alissa Cooper, Stephen Farrell, and Barry Leiba provided comments that led to further improvements.

Some algorithms and textual descriptions have been borrowed from [RFC5892]. Some text regarding security has been borrowed from [RFC5890], [I-D.ietf-precis-saslprepbis], and [I-D.ietf-xmpp-6122bis].

Peter Saint-Andre wishes to acknowledge Cisco Systems, Inc., for employing him during his work on earlier versions of this document.

Authors' Addresses

Peter Saint-Andre
&yet

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

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

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

Network Working Group
Internet-Draft
Intended status: Informational
Expires: May 4, 2016

Y. YONEYA
JPRS
T. Nemoto
Keio University
November 1, 2015

Mapping characters for PRECIS classes
draft-ietf-precis-mappings-12

Abstract

The framework for preparation, enforcement, and comparison of internationalized strings ("PRECIS") defines several classes of strings for use in application protocols. Because many protocols perform case-sensitive or case-insensitive string comparison, it necessary to define methods for case mapping. In addition, both the Internationalized Domain Names in Applications (IDNA) and the PRECIS problem statement describe mappings for internationalized strings that are not limited to case, but include width mapping and mapping of delimiters and other special characters that can be taken into consideration. This document provides guidelines for designers of PRECIS profiles and describes several mappings that can be applied between receiving user input and passing permitted code points to internationalized protocols. In particular, this document describes both locale-dependent and context-depending case mappings as well as additional mappings for delimiters and special characters.

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 May 4, 2016.

Table of Contents

1. Introduction

In many cases, user input of internationalized strings is generated through the use of an input method editor ("IME") or through copy-and-paste from free text. Users generally do not care about the case and/or width of input characters because they consider those characters to be functionally equivalent or visually identical. Furthermore, users rarely switch the IME state to input special characters such as protocol elements. For Internationalized Domain Names ("IDNs"), the IDNA Mapping specification [RFC5895] describes methods for handling these issues. For PRECIS strings, case mapping and width mapping are defined in the PRECIS framework specification [RFC7564]. The case and width mappings defined in the PRECIS framework do not handle other mappings such as delimiter characters, special characters, and locale-dependent or context-dependent case; these mappings are also important in order to increase the probability that the resulting strings compare as users expect. This document provides guidelines for authors of protocol profiles of the PRECIS framework and describes several mappings that can be applied between receiving user input and passing permitted code points to internationalized protocols. The delimiter mapping and special mapping rules described here are applied as "additional mappings" beyond those defined in the PRECIS framework, whereas the "local case mapping" rule provides locale-dependent and context-dependent alternative case mappings for specific target characters.

2. Protocol dependent mappings

The PRECIS framework defines several protocol-independent mappings. The additional mappings and local case mapping defined in this document are protocol-dependent, i.e., they depend on the rules for a particular application protocol.

2.1. Delimiter mapping

Some application protocols define delimiters for their own use, resulting in the fact that the delimiters are different for each protocol. The delimiter mapping table should therefore be based on a well-defined mapping table for each protocol.

Delimiter mapping is used to map characters that are similar to protocol delimiters into the canonical delimiter characters. For example, there are width-compatible characters that correspond to the '@' in email addresses and the ':' and '/' in URIs. The '+', '-', '<' and '>' characters are other common delimiters that might require such mapping. For the FULL STOP character (U+002E), a delimiter in the visual presentation of domain names, some IMEs produce a character such as IDEOGRAPHIC FULL STOP (U+3002) when a user types

FULL STOP on the keyboard. In all these cases, the visually similar characters that can come from user input need to be mapped to the correct protocol delimiter characters before the string is passed to the protocol.

2.2. Special mapping

Aside from delimiter characters, certain protocols have characters which need to be mapped in ways that are different from the rules specified in the PRECIS framework (e.g., mapping non-ASCII space characters to ASCII space). In this document, these mappings are called "special mappings". They are different for each protocol. Therefore, the special mapping table should be based on a well-defined mapping table for each protocol. Examples of special mapping are the following;

- o White spaces such as CHARACTER TABULATION(U+0009) or IDEOGRAPHIC SPACE(U+3000) are mapped to SPACE (U+0020)

- o Some characters such as control characters are mapped to nothing (Deletion)

As examples, EAP [RFC3748], SASLprep [RFC4013], IMAP4 ACL [RFC4314] and LDAPprep [RFC4518] define the rule that some codepoints for the non-ASCII space are mapped to SPACE (U+0020).

2.3. Local case mapping

The purpose of local case mapping is to increase the probability of results that users expect when character case is changed (e.g., map uppercase to lowercase) between input and use in a protocol. Local case mapping selectively affects characters whose case mapping depends on locale and/or context.
(Note: The term "locale" in this document practically means "language" or "language and region" because the locale based on that language configuration of applications on POSIX is selected by "locale" information and referred "Note" in section 2.1.1 of BCP 47 [RFC5646].)

As an example of locale and context-dependent mapping, LATIN CAPITAL LETTER I ("I", U+0049) is normally mapped to LATIN SMALL LETTER I ("i", U+0069); however, if the language is Turkish (or one of several other languages), unless an I is before a dot_above, the character should be mapped to LATIN SMALL LETTER DOTLESS I (U+0131).

Case mapping using Unicode Default Case Folding in the PRECIS framework does not consider such locale or context because it is a common framework for internationalization. Local case mapping

defined in this document corresponds to demands from applications which supports users' locale and/or context. The complete set of possible target characters for local case mapping are the characters specified in the SpecialCasing.txt [Specialcasing] file in the Section 3.13 of the Unicode Standard [Unicode], but the specific set of target characters selected for local case mapping depends on locale and/or context, as further explained in the SpecialCasing.txt file.

The case folding method for a selected target character is to map into lower case as defined in SpecialCasing.txt. The case folding method for all other, non-target characters is as specified in the Section 5.2.3 of the PRECIS framework . When an application supports users' locale and/or context, use of local case mapping can increase the probability that string comparisons yield the results that users expect.

If a PRECIS profile selects Unicode Default Case Folding as the preferred method of case mapping, the profile designers may consider whether local case mapping can be applied. And if it can be applied, it is better to add "alternatively, local case mapping might be applicable" after "Unicode Default Case Folding" so that application developers are aware of the alternative. See the Appendix B for a description of why local case mapping can be an alternative.

3. Order of operations

Delimiter mapping and special mapping as described in this document are expected to be applied as the "Additional Mapping Rule" mentioned in the Section 5.2.2 of the PRECIS framework. Although the delimited mapping and special mapping could be applied in either order, this document recommends the following order to minimize the effect of code point changes introduced by the mappings and to be acceptable to the widest user community:

1. Delimiter mapping

2. Special mapping

4. Security Considerations

Detailed security considerations for PRECIS strings are discussed in the PRECIS framework specification [RFC7564]. This document inherits the considerations as well.

As with Mapping Characters for IDNA2008 [RFC5895], this document suggests creating mappings that might cause confusion for some users

while alleviating confusion in other users. Such confusion is not covered in any depth in this document.

5. IANA Considerations

This document has no actions for the IANA.

6. Acknowledgment

Martin Duerst suggested a need for the case folding about the mapping (map final sigma to sigma, German sz to ss,.).

Alexey Melnikov, Andrew Sullivan, Barry Leiba, David Black, Heather Flanagan, Joe Hildebrand, John Klensin, Marc Blanchet, Pete Resnick and Peter Saint-Andre, et al. gave important suggestion for this document during working group discussions.

7. References

7.1. Normative References

[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>.

[Unicode] The Unicode Consortium, "The Unicode Standard, Version 7.0.0", <http://www.unicode.org/versions/Unicode7.0.0/>, 2012.

[Casefolding] The Unicode Consortium, "CaseFolding-7.0.0.txt", Unicode Character Database, July 2011, <http://www.unicode.org/Public/7.0.0/ucd/CaseFolding.txt>.

[Specialcasing] The Unicode Consortium, "SpecialCasing-7.0.0.txt", Unicode Character Database, July 2011, <http://www.unicode.org/Public/7.0.0/ucd/SpecialCasing.txt>.

7.2. Informative References

[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>.

[RFC3490] Faltstrom, P., Hoffman, P., and A. Costello, "Internationalizing Domain Names in Applications (IDNA)", RFC 3490, DOI 10.17487/RFC3490, March 2003, <http://www.rfc-editor.org/info/rfc3490>.

[RFC3491] Hoffman, P. and M. Blanchet, "Nameprep: A Stringprep Profile for Internationalized Domain Names (IDN)", RFC 3491, DOI 10.17487/RFC3491, March 2003, <http://www.rfc-editor.org/info/rfc3491>.

[RFC3722] Bakke, M., "String Profile for Internet Small Computer Systems Interface (iSCSI) Names", RFC 3722, DOI 10.17487/RFC3722, April 2004, <http://www.rfc-editor.org/info/rfc3722>.

[RFC3748] Aboba, B., Blunk, L., Vollbrecht, J., Carlson, J., and H. Levkowetz, Ed., "Extensible Authentication Protocol (EAP)", RFC 3748, DOI 10.17487/RFC3748, June 2004, <http://www.rfc-editor.org/info/rfc3748>.

[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>.

[RFC4314] Melnikov, A., "IMAP4 Access Control List (ACL) Extension", RFC 4314, DOI 10.17487/RFC4314, December 2005, <http://www.rfc-editor.org/info/rfc4314>.

[RFC4518] Zeilenga, K., "Lightweight Directory Access Protocol (LDAP): Internationalized String Preparation", RFC 4518, DOI 10.17487/RFC4518, June 2006, <http://www.rfc-editor.org/info/rfc4518>.

[RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <http://www.rfc-editor.org/info/rfc5646>.

[RFC5895] Resnick, P. and P. Hoffman, "Mapping Characters for Internationalized Domain Names in Applications (IDNA) 2008", RFC 5895, DOI 10.17487/RFC5895, September 2010, <http://www.rfc-editor.org/info/rfc5895>.

[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>.

[RFC6885] Blanchet, M. and A. Sullivan, "Stringprep Revision and Problem Statement for the Preparation and Comparison of Internationalized Strings (PRECIS)", RFC 6885, DOI 10.17487/RFC6885, March 2013, <http://www.rfc-editor.org/info/rfc6885>.

[ISO.3166-1] International Organization for Standardization, "Codes for the representation of names of countries and their subdivisions - Part 1: Country codes", ISO Standard 3166- 1:1997, 1997.

Appendix A. Mapping type list each protocol

A.1. Mapping type list for each protocol

This table is the mapping type list for each protocol. Values marked "o" indicate that the protocol use the type of mapping. Values marked "-" indicate that the protocol doesn't use the type of mapping.

Protocol and mapping RFC	Width (NFKC)	Delimiter	Case	Special
IDNA (RFC 3490)	-	o	-	-
IDNA (RFC 3491)	o	-	o	-
iSCSI (RFC 3722)	o	-	o	-
EAP (RFC 3748)	o	-	-	o
IMAP (RFC 4314)	o	-	-	o
LDAP (RFC 4518)	o	-	o	o

Appendix B. The reason why local case mapping is alternative to case mapping in PRECIS framework

Local case mapping is alternative to Unicode Default Case Folding instead of being applied sequentially. Because, one outstanding issue regarding full case folding for characters is, some lowercase characters like "LATIN SMALL LETTER SHARP S" (U+00DF) (hereinafter referred to as "eszett") and ligatures like "LATIN SMALL LIGATURE FF" (U+FB00) that described in section Unconditional mappings of SpecialCasing.txt become a different codepoint by performing the case mapping using Unicode Default Case Folding in the PRECIS framework. In particular, German's eszett can not keep the locale because eszett becomes two "LATIN SMALL LETTER S"s (U+0073 U+0073) by performing the case mapping using Unicode Default Case Folding. On the other hand, eszett doesn't become a different codepoint by performing the case

mapping in SpecialCasing.txt. Therefore, if it is necessary to keep locale of characters, PRECIS profile designers should select local case mapping as alternative to Unicode Default Case Folding.

Appendix C. Limitation to local case mapping

As described in the section Section 2.3, the possible target characters of local case mapping are specified in SpecialCasing.txt. The Unicode Standard (at least, up to version 7.0.0) does not define any context-dependent mappings between "GREEK SMALL LETTER SIGMA" (U+03C3) (hereinafter referred to as "small sigma") and "GREEK SMALL LETTER FINAL SIGMA" (U+03C2) (hereinafter referred to as "final sigma"). Thus, local case mapping is not applicable to small sigma or final sigma, so case mapping in the PRECIS framework always maps final sigma to small sigma, independent of context, as also specified by Unicode Default Case Folding. (Note: Following comments are from SpecialCasing.txt.)

```
# Note: the following cases are not included, since they would
case-fold in lowercasing
# 03C3; 03C2; 03A3; 03A3; Final_Sigma; # GREEK SMALL LETTER SIGMA
# 03C2; 03C3; 03A3; 03A3; Not_Final_Sigma; # GREEK SMALL LETTER
```

Appendix D. Change Log

D.1. Changes since -00

- Modify the Section 4.3 "Local case mapping" to specify the method to calculate codepoints that local case mapping targets.
- Add the Section 6 "Open issues".
- Modify the Section 7 "IANA Considerations".
- Modify the Section 8 "Security Considerations".
- Remove the "The initial PRECIS local case mapping registrations".
- Add the Appendix C "Code points list for local case mapping".
- Add the Appendix D "Change Log".

D.2. Changes since -01

- Unified PRECIS notation in all capital letters as well as other documents.

- Removed the Section 1 "Types of mapping" and the Section 2 "Protocol independent mapping" because width mapping is now in framework document.

- Added relationship between the framework document and this document in the Section 3 "Order of operations".

- Updated the Section 4 "Open issues" to address new issue raised on mailing list.

- Move the Section 6 "IANA Considerations" after the Section 5 "Security Considerations".

- Remove the Appendix B "Codepoints which need special mapping" and mentioned related documents in the Section 2.2 .

D.3. Changes since -02

- Removed the "Open issues".

D.4. Changes since -03

- Modify the Section 1 "Introduction" in more clear text.

- Modify the Section 2.3 "Local case mapping" to clarify the purpose of the local case mapping and an example, and add restriction to use with PRECIS framework.

- Change the format in the Appendix B "Code points list for local case mapping".

- Split the Section 7 "References" into "Normative References" and "Informative References"

- Update the Unicode version 6.2 to 6.3 in this document.

D.5. Changes since -04

- Correct a sentence in the Section 2.3 "Local case mapping".

D.6. Changes since -05

- Correct some sentences in this document.

- Modify the local case mapping’s rule and target characters in the Section 2.3 "Local case mapping". This is to avoid user’s confusion towards Greek’s final sigma and German’s eszett.

- Add the Section 4 "Open issues".
- Modify the Section 8 "Security Considerations".
- Modify the table format in the Appendix A. "Mapping type list each protocol".
- Removed the Appendix B "Code points list for local case mapping".
- Add the Appendix B "Local case mapping vs Case mapping".

D.7. Changes since -06

- Removed the Section 4 "Open issues".
- Change the title of the Appendix B "Local case mapping vs Case mapping" to "The reason why local case mapping is alternative to case mapping in PRECIS framework".
- Add the Appendix C "Limitation to local case mapping".

D.8. Changes since -07

- Modify the Section 1 "Introduction".
- Modify the local case mapping’s rule and target characters in the Section 2.3 "Local case mapping".
- Modify the Section 3 "Order of operations".

D.9. Changes since -08

- Updated the Unicode version 6.3 to 7.0 in this document.

D.10. Changes since -09

- Modify the Section 1 "Introduction" to clarify to the discussion of string matching and the use of mappings from the SpecialCasing.txt.
- Modify the Section 2.3 "Local case mapping" to clarify to the discussion of string matching and the use of mappings from the SpecialCasing.txt.
- Modify the Appendix B "The reason why local case mapping is alternative to case mapping in PRECIS framework" to state the result of the case mapping in SpecialCasing.txt of eszett.

- Clarify the Appendix C "Limitation to local case mapping".

D.11. Changes since -10

- Modify the "Abstract" to clarify to sentences.
- Modify the Section 1 "Introduction" to clarify to a sentence.
- Modify the Section 2.2 "Special mapping" to add examples.
- Modify the Section 2.3 "Local case mapping" to clarify to sentences. And add a note to explain the term "locale" in this document.
- Modify the Section 3 "Order of operations" to clarify to sentences.
- Correct a sentence in the Section 4 "Security Considerations".
- Modify a sentence in the Section 6 "Acknowledgment".
- Change the references from [I-D.ietf-precis-framework] to [RFC7564] in the Section 7 "Normative References".
- Removed SASL and XMPP in the table of the Appendix A. "Mapping type list each protocol".
- Modify the Appendix B "The reason why local case mapping is alternative to case mapping in PRECIS framework" to clarify to sentences.
- Modify the Appendix C "Limitation to local case mapping" to clarify to sentences.

D.12. Changes since -11

- Correct a few sentence in the Section 2.3 "Local case mapping" to address comments by the IESG review.
- Removed citation part which includes "RECOMMENDED" (RFC 2119 word) in the Section 2.3 "Local case mapping" to avoid readers' confusion.
- Modify the Section 4 "Security Considerations" to add a reference to RFC7564.

Authors' Addresses

Yoshiro YONEYA
JPRS
Chiyoda First Bldg. East 13F
3-8-1 Nishi-Kanda
Chiyoda-ku, Tokyo 101-0065
Japan

Phone: +81 3 5215 8451
Email: yoshiro.yoneya@jprs.co.jp

Takahiro Nemoto
Keio University
Graduate School of Media Design
4-1-1 Hiyoshi, Kohoku-ku
Yokohama, Kanagawa 223-8526
Japan

Phone: +81 45 564 2517
Email: t.nemo10@kmd.keio.ac.jp

PRECIS
Internet-Draft
Obsoletes: 4013 (if approved)
Intended status: Standards Track
Expires: November 29, 2015

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

May 28, 2015

Preparation, Enforcement, and Comparison of Internationalized Strings Representing Usernames and Passwords
draft-ietf-precis-saslprepbis-18

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 PRECIS framework, RFC 7564, obsoletes RFC 3454, and this document obsoletes RFC 4013.

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 November 29, 2015.

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 [I-D.ietf-httpauth-basicauth-update]) 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 [I-D.ietf-httpauth-digest]).

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 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+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 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., 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 as specified in [I-D.ietf-httpauth-basicauth-update] and [I-D.ietf-httpauth-digest].

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 [I-D.ietf-xmpp-6122bis], which obsoletes [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 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 [I-D.ietf-httpauth-basicauth-update] and [I-D.ietf-httpauth-digest]), 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" (where the complete 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 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.

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 (on the distinction between these actions, refer to [RFC7564]).

3.2.1. 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 "IdentifierClass" base string class defined in [RFC7564]. In addition, the string MUST be encoded as UTF-8 [RFC3629].

3.2.2. Enforcement

An entity that performs enforcement according to this profile MUST prepare a string as described in the previous section 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: Fullwidth and halfwidth characters MUST be mapped to their decomposition mappings (see Unicode Standard Annex #11 [UAX11]).

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 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").

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 (on the distinction between these actions, refer to [RFC7564]).

3.3.1. 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 "IdentifierClass" base string class defined in [RFC7564]. In addition, the string MUST be encoded as UTF-8 [RFC3629].

3.3.2. Enforcement

An entity that performs enforcement according to this profile MUST prepare a string as described in the previous section 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: Fullwidth and halfwidth characters MUST be mapped to their decomposition mappings (see Unicode Standard Annex #11 [UAX11]).

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 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").

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, 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 apply the UsernameCasePreserved profile and thus not perform case mapping, such as backward compatibility with deployed infrastructure.

In particular:

- 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, 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.

- 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.

- Application protocols that do not use SASL (such as HTTP authentication with the Basic and Digest schemes as specified in [I-D.ietf-httpauth-basicauth-update] and [I-D.ietf-httpauth-digest]) but that directly re-use this profile 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 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 (all 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 space-separated set of names like "Firstname Middlename Lastname". Although such a construct is not a PRECIS profile (since 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 (or a profile thereof).

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).

Table 1: A sample of legal userparts

#	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)

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.

Table 2: A sample of strings that violate the userpart rule

#	Non-Userpart string	Notes
8	<foo bar>	Space (U+0020) is disallowed in the userpart
9	<>	Zero-length userpart
10	<henryⅣ>	The sixth character is ROMAN NUMERAL FOUR (U+2163)
11	<♚>	A localpart of BLACK CHESS KING (U+265A)

Here again, several points are worth noting. 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 IdentiferClass. 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 OpaqueString profile of the PRECIS FreeformClass specified below.

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.

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 on zero-length passwords is not a recommendation regarding password strength (since a password of only one byte is highly insecure), but is meant to prevent applications from mistakenly omitting a password entirely, since when internationalized characters are accepted 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 proper preparation of the password before applying the algorithm, since 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 (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 string MUST be encoded as UTF-8 [RFC3629].

4.2.2. Enforcement

An entity that performs enforcement according to this profile MUST prepare a string as described in the previous section and MUST also apply the rules specified below (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 general category of "Zs", naturally 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, 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. 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 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").

4.3. 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).

Table 3: A sample of legal passwords

#	Password	Notes
12	<correct horse battery staple>	ASCII space is allowed
13	<Correct Horse Battery Staple>	Different 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>

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

Table 4: A string that violates the password rules

#	Password	Notes
17	<my cat is a by>	Controls are disallowed

5. Use in Application Protocols

This specification defines only the PRECIS-based rules for 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 when in protocol processing or interface handling the rules need to be enforced. See Section 6 of [RFC7564] for guidelines about 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 since it can lead to false positives. An in-depth discussion can be found in "Issues in Identifier Comparison for Security Purposes" [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 migrating to use of the rules defined in this specification. In particular:

- 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. 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.

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 U+0049 LATIN CAPITAL LETTER I and U+0056 LATIN CAPITAL LETTER V vs. "HENRYIV" with the last character being U+2163 ROMAN NUMERAL FOUR, 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.

- 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 9.13 of [RFC7564] (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 7.0). For migration purposes, the operator might want to remove from usernames any code points contained in the PRECIS "M" category (e.g., U+00AD SOFT HYPHEN). 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.

- 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 either use the UsernameCaseMapped profile (thus losing the case information) or use 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 (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 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. 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 9.13 of [RFC7564] (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 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

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

7.1. UsernameCaseMapped Profile

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 XXXX, Section 3.2. [Note to RFC Editor: please change XXXX to the number issued for this specification.]

7.2. UsernameCasePreserved Profile

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 XXXX, Section 3.3. [Note to RFC Editor: please change XXXX to the number issued for this specification.]

7.3. OpaqueString Profile

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 XXXX, Section 4.2. [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 [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, 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.

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

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

[RFC7564] Saint-Andre, P. and M. Blanchet, "PRECIS Framework: Preparation, Enforcement, and Comparison of Internationalized Strings in Application Protocols", RFC 7564, May 2015.

[UAX11] The Unicode Consortium, "Unicode Standard Annex #11: East Asian Width", September 2012, <http://unicode.org/reports/tr11/>.

[Unicode7.0] The Unicode Consortium, "The Unicode Standard, Version 7.0.0", 2014, <http://www.unicode.org/versions/Unicode7.0.0/>.

[Unicode] The Unicode Consortium, "The Unicode Standard", 2015-present, <http://www.unicode.org/versions/latest/>.

9.2. Informative References

[I-D.ietf-httpauth-basicauth-update]
Reschke, J., "The 'Basic' HTTP Authentication Scheme", draft-ietf-httpauth-basicauth-update-07 (work in progress), February 2015.

[I-D.ietf-httpauth-digest]
Shekh-Yusef, R., Ahrens, D., and S. Bremer, "HTTP Digest Access Authentication", draft-ietf-httpauth-digest-19 (work in progress), April 2015.

[I-D.ietf-xmpp-6122bis]
Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Address Format", draft-ietf-xmpp-6122bis-22 (work in progress), May 2015.

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

[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.

[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.

[RFC6122] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Address Format", RFC 6122, March 2011.

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

[RFC7542] DeKok, A., "The Network Access Identifier", RFC 7542, May 2015.

[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 [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 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 three separate algorithms: one for usernames with case mapping, one for usernames with case preservation, and one 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 framework, 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

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

The following individuals provided helpful feedback on this document: Marc Blanchet, Ben Campbell, Alan DeKok, Joe Hildebrand, Jeffrey Hutzelman, Simon Josefsson, Jonathan Lennox, James Manger, Matt Miller, Chris Newman, Yutaka OIWA, Pete Resnick, Andrew Sullivan, Nico Williams, and Yoshiro YONEYA. Nico Williams in particular deserves special recognition for providing text that was used in Section 3.4. Thanks also to Takahiro NEMOTO and Yoshiro YONEYA for implementation feedback.

Robert Sparks and Derek Atkins reviewed the document on behalf of the General Area Review Team and the Security Directorate, respectively.

Stephen Farrell provided helpful input during IESG review.

Peter Saint-Andre wishes to acknowledge Cisco Systems, Inc., for employing him during his work on earlier draft versions of this document.

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

PRECIS
Internet-Draft
Intended status: Standards Track
Expires: January 9, 2014

Y. Oiwa
RISEC, AIST
T. Nemoto
Keio University
B. Kihara
Lepidum

July 8, 2013

HTTPAuthPrep: PRECIS profile for HTTP Authentication
draft-oiwa-precis-httpauthprep-00

Abstract

This document describes how to handle Unicode strings representing user names and passwords for HTTP authentication.

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 January 9, 2014.

Copyright Notice

Table of Contents

1. Introduction

1.1. Overview

This document describes how to handle Unicode strings representing user names and passwords for HTTP authentication.

For a long time starting HTTP/1.0 [RFC1945], character encodings of HTTP authentication related parameters are defined and handled quite loosely. [RFC1945] defined user-names of Basic authentication to be a subset of ASCII strings (token), and passwords to be assumed as ISO-8859-1 by recipients. Initial version of HTTP/1.1 [RFC2068] and later revisions define grammar rules which indirectly (through the definition of "TEXT" element) insist that both user-names and passwords are in ISO-8859-1. In any way, these definitions are quite often disregarded, and implementations tend to use their local character sets and encodings, which has caused several interoperability problems.

At the time of being (writing this document), the most promising way of solving this problem is to use Unicode [UNICODE] character set along with UTF-8 encoding [RFC3629] as a common vehicle. However, just using UTF-8 does not completely solve the problem, or even makes it worse, because of the non-unique encoding nature of Unicode character sets.

Recently, a PRECIS [I-D.ietf-precis-framework] framework is being standardized to cover this problem set. It defines a framework to resolve non-uniqueness problem of Unicode character sets for information-comparison purposes, especially useful for user identifications. This document describes how to apply the PRECIS framework for general HTTP user authentications, who to implement such framework, and how to use it.

1.2. Applicability

The rules defined in this document can be used in two ways: one way is to use them as MUST- (or SHOULD-) obey rules, by referring it from another standard or non-standard document. In such case, the rules defined in this document will have a normative property.

Another way is to use them as "best current practices", when some specific HTTP authentication scheme does not define any specific method of string preparations. In such case, any implementations are not required to implement (or not to implement) the string preparation rule in this document, but using it may sometimes improve interoperability between implementations.

Any specific authentication scheme MAY define its own string preparation method, especially when an underlying software layer supporting the authentication scheme (such as SASL) defines (or recommends) its own string preparation method. In such cases, implementations SHOULD NOT use the preparation rules described in this document, and these SHOULD obey the scheme-specific requirement.

It is not feasible to implement the string preparation within all HTTP implementations. For interoperability of authentication process, only a small portion of involved softwares are required to actually implement the string preparation algorithms. To this purpose, general application notes are provided in the latter part of this document.

1.3. Terminology

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].

2. Rules

This section defines two PRECIS string preparation rules for user names and passwords.

Note: the RFC 2119 requirements keywords such as "MUST" within this section are effective as the RFC keywords only when the application of these rules are either REQUIRED or RECOMMENDED by any authentication scheme definitions.

2.1. User Names

2.1.1. Definition

User names are strings to identify the user in HTTP authentication.

```
username       = 1*(idpoint)
                 ;
                 ; an "idpoint" is a UTF-8 encoded
                 ; Unicode code point that conforms to
                 ; the PRECIS "IdentifierClass"
                 ;
```

Note that some authentication schemes like Basic MAY restrict several characters to be used in username.

Note also that some authentication schemes like Digest modifies users' inputs to other forms like quoted-string. This document specifies only string preparation.

2.1.2. Preparation

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

Each username MUST conform to the definition of the PRECIS IdentifierClass provided in [I-D.ietf-precis-framework], where the width mapping, additional mapping, case mapping, normalization, and directionality rules are as described below.

1. Fullwidth and halfwidth characters MUST be mapped to their decomposition equivalents.
2. Additional mappings SHOULD NOT be applied, such as those defined in [I-D.ietf-precis-mappings], unless there are implementation-dependent reasons to do so, or these are exceptionally required by specific authentication schemes.
3. Case mapping is not applied.
4. Unicode Normalization Form C (NFC) MUST be applied to all characters.

With regard to directionality, the "Bidi Rule" provided in [RFC5893] applies.

2.2. Passwords

2.2.1. Definition

Passwords are strings to authenticate the user in HTTP authentication.

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

Note that some authentication schemes MAY restrict several characters to be used in passwords.

2.2.2. Preparation

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

A password MUST be treated as follows, where the operations specified MUST be completed in the order shown:

1. Width mapping is not applied.
2. Map any instances of non-ASCII space to ASCII space (U+0020).
3. Case mapping is not applied.
4. Apply Unicode Normalization Form C (NFC) to all characters.
5. Ensure that the resulting string conforms to the definition of the PRECIS FreeformClass.

With regard to directionality, the "Bidi Rule" (defined in [RFC5893]) and similar rules do not apply.

3. Application Notes

Implementation of the above rules are sometimes resource-consuming and not realistic, especially when the implementation is not aware of any Unicode string and is handling the authentication credentials as opaque byte strings. This section provides a general application notes for how to realize the above string preparation in the real software.

Note: the note for RFC keywords in the previous section does apply also for this section.

The general principle for the application is: "to send the string correctly, by some means." In particular, if there is "some" provision (either manually or automatically) to ensure the correct encoding and preparation of string at the time of sending, it is considered enough. As a definitive rule, the following provisions are to be taken:

- Recipient side (i.e. HTTP servers) MAY omit any part of string preparation, including Unicode normalization. It MAY process any received strings as is.
- Senders which forward already-prepared strings (i.e. HTTP proxies etc.) MAY omit any part of string preparation.
- Interactive clients which receive human users' input, as form of "characters", have an obligation to prepare the input string into a correct UTF-8 string with regards to the scheme-specific preparation rules. When the authentication scheme specifies that the preparation is a MUST, they MUST do it.
- Clients which receive credentials in a form of "list of octets" (such as those within configuration files) MAY require its users to prepare the string correctly within configuration phases, and MAY omit any part of string preparation at runtime.

As a reverse to these rules, any recipients MUST be prepared to

receive any unprepared byte lists or character lists as inputs. Such recipients MAY prepare the string by its own, MAY reject such inputs explicitly, or MAY process these inputs silently when it will lead to failed authentication attempts. However, Such recipients MUST NOT process such inputs in any way which leads to false authentication successes (modulo cryptographically negligible level of probabilities).

4. Design principles

Note: the content of this section is not normative.

The design of the rules provided in previous sections are made under the following concerns and observations.

- ASCII transparency:
  Every code-point within U+0020 to U+007E MUST be preserved, distinguished, and mapped to its one-byte equivalent respectively. This is a strong requirement for compatibility with existing HTTP authentication.
- Latin-1 preservation:
  Code-points U+00A1 to U+00FE SHOULD be preserved and distinguished in the output string. This enables non-crashing mapping from existing ISO-8859-1 user databases, and, if applicable, enables backward-compatible server-side implementation for Basic plain-text authentication.
- Case (non-)mapping:
  As a subset of the above two rules, no case mapping shall be applied (as a basic rule). Without this, some existing user database will become non-useful, especially when it has already used a non-mapped credentials, and its entries are hashed or one-way encrypted. The opposite case (case-mapped existing databases) can be at least worked around by users, server implementations, or both.
  Of course, some authentication schemes designed for specific use-cases can always define a case-folded mappings whenever needed.
- Strong normalizations:
  All representations (decomposed and composed) of every single "character" within Unicode MUST be normalized to exactly one UTF-8 byte sequence. Without this, virtually all "non-Basic" authentication may become broken with regard to internationalized username and passwords (e.g. Digest).

5. Security Considerations

As mentioned previously, any recipients MUST NOT assume that senders

will always send a correctly-prepared strings. Care must be taken that incorrectly-prepared strings MUST lead to either a correct result or an authentication failure.

6. IANA Considerations

[[TBD: more precise IANA Considerations here.]]

The IANA shall add the following two entries to the PRECIS Usage Registry:

```
----
Applicability: User Names in HTTP Authentication.
Base Class: IdentifierClass.
Subclass: No.
Replaces: No.
Width Mapping: Map fullwidth and halfwidth characters to their
decomposition equivalents.
Additional Mappings: None.
Case Mapping: None.
Normalization: NFC.
Directionality: The "Bidi Rule" defined in RFC 5893 applies.
Specification: RFC XXXX. [Note to RFC Editor: please change XXXX to
the number issued for this specification.]
----
```

```
----
Applicability: Passwords of HTTP Authentication.
Base Class: FreeformClass
Subclass: No.
Replaces: No.
Width Mapping: None.
Additional Mappings: Map non-ASCII space to ASCII space (U+0020).
Case Mapping: None.
Normalization: NFC.
Directionality: The "Bidi Rule" defined in RFC 5893 does not apply.
Specification: RFC XXXX. [Note to RFC Editor: please change XXXX to
the number issued for this specification.]
----
```

7. References

7.1. Normative References

[I-D.ietf-precis-framework]
Saint-Andre, P. and M. Blanchet, "PRECIS Framework:

Preparation and Comparison of Internationalized Strings in Application Protocols", draft-ietf-precis-framework-08 (work in progress), April 2013.

[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.

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

[UNICODE] The Unicode Consortium, "The Unicode Standard, Version 6.1", 2012, <http://www.unicode.org/versions/Unicode6.1.0/>.

7.2. Informative References

[RFC1945] Berners-Lee, T., Fielding, R., and H. Nielsen, "Hypertext Transfer Protocol -- HTTP/1.0", RFC 1945, May 1996.

[RFC2068] Fielding, R., Gettys, J., Mogul, J., Nielsen, H., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2068, January 1997.

[I-D.ietf-precis-mappings] Yoneya, Y. and T. NEMOTO, "Mapping characters for PRECIS classes", draft-ietf-precis-mappings-02 (work in progress), May 2013.

Appendix A. Document History (to be removed)

Initial submit.

Authors' Addresses

Yutaka Oiwa
National Institute of Advanced Industrial Science and Technology
Research Institute for Secure Systems
3-11-46 Nakouji
Amagasaki, Hyogo
JP

Email: mutual-auth-contact-ml@aist.go.jp

Takahiro Nemoto
Keio University
Graduate School of Media Design
4-1-1 Hiyoshi, Kohoku-ku
Yokohama, Kanagawa 223-8526
Japan

Email: t.nemo10@kmd.keio.ac.jp

Boku Kihara
Lepidum Co. Ltd.
#602, Village Sasazuka 3
1-30-3 Sasazuka
Shibuya-ku, Tokyo
Japan