Internet Engineering Task Force (IETF)
Request for Comments: 8265
Obsoletes: 7613
Category: Standards Track
ISSN: 2070-1721

P. Saint-Andre
Jabber.org
A. Melnikov
Isode Ltd

October 2017

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

Abstract

This document describes updated methods for handling Unicode strings representing usernames and passwords. The previous approach was known as SASLprep (RFC 4013) and was based on Stringprep (RFC 3454). The methods specified in this document provide a more sustainable approach to the handling of internationalized usernames and passwords. This document obsoletes RFC 7613.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 7841.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at https://www.rfc-editor.org/info/rfc8265.

Table of Contents

1. Introduction

Usernames and passwords are widely used for authentication and authorization on the Internet, either directly when provided in plaintext (as in the PLAIN Simple Authentication and Security Layer (SASL) mechanism [RFC4616] and the HTTP Basic scheme [RFC7617]) or indirectly when provided as the input to a cryptographic algorithm such as a hash function (as in the Salted Challenge Response Authentication Mechanism (SCRAM) SASL mechanism [RFC5802] and the HTTP Digest scheme [RFC7616]).

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 handling internationalized strings that represent usernames and passwords. Such strings consist of code points from the Unicode coded character set [Unicode], with special attention to code points outside the ASCII range [RFC20]. The rules for handling such strings are specified through profiles of the string classes defined in the preparation, enforcement, and comparison of internationalized strings (PRECIS) framework specification [RFC8264].

Profiles of the PRECIS framework enable software to handle Unicode code points outside the ASCII range in an automated way, so that such code points 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 code points. For instance, in almost all application protocols it would be dangerous to treat the Unicode code point "Â¹" (SUPERSCRIPT ONE, U+00B9) as equivalent to "1" (DIGIT ONE, U+0031), because that would result in false accepts during comparison, authentication, and authorization (e.g., an attacker could easily spoof an account "user1@example.com").

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

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

Although the historical predecessor of this document was the SASLprep profile of Stringprep [RFC3454]), the approach defined here can be used by technologies other than SASL [RFC4422], such as HTTP authentication as specified in [RFC7617] and [RFC7616].

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 [RFC7622]). Non-coordinated updates to protocol implementations are discouraged because they can have a negative impact on interoperability and security.

2. Terminology

A "username" or "user identifier" is a string of characters designating an account on a computing device or system, often but not necessarily for use by a person. Although some devices and systems might allow a username to be part or all of a person's name and a person might want their account designator to be part or all of their name, because of the complexities involved, that outcome is not guaranteed for all human names on all computing devices or systems that follow the rules defined in this specification. Protocol designers and application developers who wish to allow a wider range of characters are encouraged to consider a separation between more restrictive account identifiers and more expressive display names or nicknames (see [RFC8266]).

A "password" is a string of characters that allows access to a computing device or system, often associated with a particular username. A password is not literally limited to a word, because 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 username" (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 [RFC7617] and [RFC7616]), 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; note also that a username does not necessarily map to any particular application identifier.

Many important terms used in this document are defined in [RFC5890], [RFC6365], [RFC8264], and [Unicode]. The term "non-ASCII space" refers to any Unicode code point having a Unicode general category of "Zs", naturally with the exception of SPACE (U+0020).

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 BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Usernames

3.1. Definition

This document specifies that a username is a string of Unicode code points [Unicode] that is structured as an ordered sequence of "userparts" and expressed in a standard Unicode Encoding Form (such as UTF-8 [RFC3629]). A userpart is allowed to contain only code points that are allowed by the PRECIS IdentifierClass defined in Section 4.2 of [RFC8264] and thus consists almost exclusively of letters and digits. A username can consist of a single userpart or a space-separated sequence of userparts.

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

```
username   = userpart *(1*SP userpart)
userpart   = 1*(idpoint)
             ;
             ; an "idpoint" is a Unicode code point that
             ; can be contained in a string conforming to
             ; the PRECIS IdentifierClass
             ;
```

All code points and blocks not explicitly allowed in the PRECIS IdentifierClass are disallowed; this includes private-use code points, surrogate code points, and the other code points and blocks that were defined as "Prohibited Output" in Section 2.3 of [RFC4013] (when corrected per [Err1812]). 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 code points that can be used in

existing SASL mechanisms and in application protocols that use SASL, and even in most application protocols that do not currently use SASL.

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

This specification defines two profiles for usernames: the UsernameCaseMapped profile performs case mapping, and the UsernameCasePreserved performs case preservation (see further discussion under Section 3.2).

In protocols that provide usernames as input to a cryptographic algorithm such as a hash function, the client will need to perform enforcement of the rules for the UsernameCaseMapped or UsernameCasePreserved profile before applying the algorithm.

3.2. 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 their use (or not) of the Case Mapping Rule and are otherwise identical.

Case mapping is a matter for the application protocol, protocol implementation, or end deployment. In general, this document suggests that it is preferable to apply the UsernameCaseMapped profile and therefore perform case mapping, because not doing so can lead to false accepts 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. Because case mapping results in information loss, in order to retain that information for as long as possible during processing, implementations SHOULD delay any case mapping to the last possible moment, such as when doing a lookup by username, performing username comparisons, or generating a cryptographic salt from a username (if the last possible moment happens on a server, then decisions about case mapping can be a matter of service deployment policy). In keeping with [RFC4422],

SASL mechanisms are not to apply this or any other profile to authorization identifiers, only to authentication identifiers.

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

- Application protocols that do not use SASL (such as HTTP authentication with the HTTP Basic and Digest schemes as specified in [RFC7617] and [RFC7616]) but that directly reuse this profile MUST specify whether and when case mapping is to be applied to authentication identifiers or authorization identifiers, or both. Such "non-SASL application protocols" SHOULD delay any case mapping to the last possible moment, such as when doing a lookup by username, performing username comparisons, or generating a cryptographic salt from a username (if the last possible moment happens on the server, then decisions about case mapping can be a matter of service deployment policy).

If the specification for a SASL mechanism, SASL application protocol, or non-SASL application protocol uses the UsernameCaseMapped profile, it MUST clearly describe whether case mapping is to be applied at the level of the protocol itself, implementations thereof, or service deployments (each of these approaches can be legitimate, depending on the application in question).

3.3. UsernameCaseMapped Profile

3.3.1. Rules

The following rules are defined for use within the UsernameCaseMapped profile of the PRECIS IdentifierClass.

1. Width Mapping Rule: Map fullwidth and halfwidth code points to their decomposition mappings (see Unicode Standard Annex #11 [UAX11]).

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

3. Case Mapping Rule: Map uppercase and titlecase code points to their lowercase equivalents, preferably using the Unicode toLowerCase() operation as defined in the Unicode Standard [Unicode]; see further discussion in Section 3.2.

4. Normalization Rule: Apply Unicode Normalization Form C (NFC) to all strings.

5. Directionality Rule: Apply the "Bidi Rule" defined in [RFC5893] to strings that contain right-to-left code points (i.e., each of the six conditions of the Bidi Rule must be satisfied); for strings that do not contain right-to-left code points, there is no special processing for directionality.

3.3.2. Preparation

An entity that prepares an input string for subsequent enforcement according to this profile MUST proceed as follows (applying the steps in the order shown).

1. Apply the width mapping rule specified in Section 3.3.1. It is necessary to apply the rule at this point because otherwise the PRECIS "HasCompat" category specified in Section 9.17 of [RFC8264] would forbid fullwidth and halfwidth code points.

2. Ensure that the string consists only of Unicode code points that are explicitly allowed by the PRECIS IdentifierClass defined in Section 4.2 of [RFC8264].

3.3.3. Enforcement

An entity that performs enforcement according to this profile MUST prepare an input string as described in Section 3.3.2 and MUST also apply the following rules specified in Section 3.3.1 in the order shown:

1. Case Mapping Rule

2. Normalization Rule

3. Directionality Rule

After all of the foregoing rules have been enforced, the entity MUST ensure that the username is not zero bytes in length (this is done after enforcing the rules to prevent applications from mistakenly omitting a username entirely, because when internationalized strings are accepted, a non-empty sequence of characters can result in a zero-length username after canonicalization).

The result of the foregoing operations is an output string that conforms to the UsernameCaseMapped profile. Until an implementation produces such an output string, it MUST NOT treat the string as conforming (in particular, it MUST NOT assume that an input string is conforming before the enforcement operation has been completed).

3.3.4. Comparison

An entity that performs comparison of two strings according to this profile MUST prepare each string as specified in Section 3.3.2 and then MUST enforce the rules specified in Section 3.3.3. The two strings are to be considered equivalent if and only if they are an exact octet-for-octet match (sometimes called "bit-string identity").

Until an implementation determines whether two strings are to be considered equivalent, it MUST NOT treat them as equivalent (in particular, it MUST NOT assume that two input strings are equivalent before the comparison operation has been completed).

3.4. UsernameCasePreserved Profile

3.4.1. Rules

The following rules are defined for use within the UsernameCasePreserved profile of the PRECIS IdentifierClass.

1. Width Mapping Rule: Map fullwidth and halfwidth code points to their decomposition mappings (see Unicode Standard Annex #11 [UAX11]).

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

3. Case Mapping Rule: There is no case mapping rule.

4. Normalization Rule: Apply Unicode Normalization Form C (NFC) to all strings.

5. Directionality Rule: Apply the "Bidi Rule" defined in [RFC5893] to strings that contain right-to-left code points (i.e., each of the six conditions of the Bidi Rule must be satisfied); for strings that do not contain right-to-left code points, there is no special processing for directionality.

3.4.2. Preparation

An entity that prepares a string for subsequent enforcement according to this profile MUST proceed as follows (applying the steps in the order shown).

1. Apply the width mapping rule specified in Section 3.4.1. It is necessary to apply the rule at this point because otherwise the PRECIS "HasCompat" category specified in Section 9.17 of [RFC8264] would forbid fullwidth and halfwidth code points.

2. Ensure that the string consists only of Unicode code points that are explicitly allowed by the PRECIS IdentifierClass defined in Section 4.2 of [RFC8264].

3.4.3. Enforcement

An entity that performs enforcement according to this profile MUST prepare a string as described in Section 3.4.2 and MUST also apply the following rules specified in Section 3.4.1 in the order shown:

1. Normalization Rule

2. Directionality Rule

After all of the foregoing rules have been enforced, the entity MUST ensure that the username is not zero bytes in length (this is done after enforcing the rules to prevent applications from mistakenly omitting a username entirely, because when internationalized strings are accepted, a non-empty sequence of characters can result in a zero-length username after canonicalization).

The result of the foregoing operations is an output string that conforms to the UsernameCasePreserved profile. Until an implementation produces such an output string, it MUST NOT treat the string as conforming (in particular, it MUST NOT assume that an input string is conforming before the enforcement operation has been completed).

3.4.4. Comparison

An entity that performs comparison of two strings according to this profile MUST prepare each string as specified in Section 3.4.2 and then MUST enforce the rules specified in Section 3.4.3. The two strings are to be considered equivalent if and only if they are an exact octet-for-octet match (sometimes called "bit-string identity").

Until an implementation determines whether two strings are to be considered equivalent, it MUST NOT treat them as equivalent (in particular, it MUST NOT assume that two input strings are equivalent before the comparison operation has been completed).

3.5. Application-Layer Constructs

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

3.6. Examples

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

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

Table 1: A Sample of Legal Userparts

Regarding examples 2 and 3: although in German writing the character eszett "Ã" (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 "Î£" (GREEK CAPITAL LETTER SIGMA, U+03A3) to the lowercase character "Ï" (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 the character "Ï‚" (GREEK SMALL LETTER FINAL SIGMA, U+03C2), the userparts in examples 5 and 7 or examples 6 and 7 would not be matched during comparison.

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

#	Non-Userpart String	Notes
8	<foo bar>	SPACE (U+0020) is disallowed in the userpart
9	<>	Zero-length userpart
10	<henryâ£>	The sixth character is ROMAN NUMERAL FOUR (U+2163)
11	<â>	A userpart of INFINITY (U+221E)

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

Regarding example 8: although this is not a valid userpart, it is a valid username because it is a space-separated sequence of userparts.

Regarding example 10: the character "â£" (ROMAN NUMERAL FOUR, U+2163) has a compatibility equivalent of the characters "I" (LATIN CAPITAL LETTER I, U+0049) and "V" (LATIN CAPITAL LETTER V, U+0056), but code points with compatibility equivalents are not allowed in the PRECIS IdentifierClass.

Regarding example 11: symbol characters such as "â" (INFINITY, U+221E) 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] that is conformant to the OpaqueString profile (specified below) of the PRECIS FreeformClass defined in Section 4.3 of [RFC8264] and expressed in a standard Unicode Encoding Form (such as UTF-8 [RFC3629]).

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

```
password   = 1*(freepoint)
             ;
             ; a "freepoint" is a Unicode code point that
             ; can be contained in a string conforming to
             ; the PRECIS FreeformClass
             ;
```

All code points and blocks not explicitly allowed in the PRECIS FreeformClass are disallowed; this includes private-use code points, surrogate code points, and the other code points and blocks defined as "Prohibited Output" in Section 2.3 of [RFC4013] (when corrected per [Err1812]).

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

> Note: Some existing systems allow an empty string in places where a password would be expected (e.g., command-line tools that might be called from an automated script, or servers that might need to be restarted without human intervention). From the perspective of this document (and RFC 4013 before it), these empty strings are not passwords but are workarounds for the practical difficulty of using passwords in certain scenarios.

> Note: The prohibition of zero-length passwords is not a recommendation regarding password strength (because a password of only one byte is highly insecure) but is meant to prevent applications from mistakenly omitting a password entirely; such an outcome is possible when internationalized strings are accepted, because a non-empty sequence of characters can result in a zero-length password after canonicalization.

In protocols that provide passwords as input to a cryptographic algorithm such as a hash function, the client will need to perform enforcement of the rules for the OpaqueString profile before applying

the algorithm, because the password is not available to the server in plaintext form.

4.2. OpaqueString Profile

The definition of the OpaqueString profile is provided in the following sections, including detailed information about preparation, enforcement, and comparison (for details on the distinction between these actions, refer to [RFC8264]).

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 are explicitly allowed by the FreeformClass string class defined in [RFC8264].

4.2.2. Enforcement

An entity that performs enforcement according to this profile MUST prepare a string as described in Section 4.2.1 and MUST also apply the rules specified below for the OpaqueString profile (these rules MUST be applied in the order shown):

1. Width Mapping Rule: Fullwidth and halfwidth code points 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 SPACE (U+0020); a non-ASCII space is any Unicode code point having a Unicode general category of "Zs", with the exception of SPACE (U+0020). As was the case in RFC 4013, the inclusion of only SPACE (U+0020) prevents confusion with various non-ASCII space code points, many of which are difficult to reproduce across different input methods.

3. Case Mapping Rule: There is no case mapping rule (because mapping uppercase and titlecase code points to their lowercase equivalents would lead to false accepts and thus to reduced security).

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

5. Directionality Rule: There is no directionality rule. The "Bidi Rule" (defined in [RFC5893]) and similar rules are unnecessary and inapplicable to passwords, because they can reduce the repertoire 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.

The result of the foregoing operations is an output string that conforms to the OpaqueString profile. Until an implementation produces such an output string, it MUST NOT treat the string as conforming (in particular, it MUST NOT assume that an input string is conforming before the enforcement operation has been completed).

4.2.3. Comparison

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

Until an implementation determines whether two strings are to be considered equivalent, it MUST NOT treat them as equivalent (in particular, it MUST NOT assume that two input strings are equivalent before the comparison operation has been completed).

See Section 8.2 regarding comparison of passwords and passphrases.

4.3. Examples

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

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

Table 3: A Sample of Legal Passwords

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

#	Password	Notes
17	<>	Zero-length passwords are disallowed
18	<my cat is a by>	Control characters like TAB (U+0009) are disallowed

Table 4: A Sample of Strings That Violate the Password Rules

Note: Following the "XML Notation" used in [RFC3987], the character TAB (U+0009) in example 18 is represented as &#x9 because otherwise it could not be shown in running text.

5. Use in Application Protocols

This specification defines only the PRECIS-based rules for the handling of strings conforming to the UsernameCaseMapped and UsernameCasePreserved profiles of the PRECIS IdentifierClass, and strings conforming to the OpaqueString profile of the PRECIS

FreeformClass. It is the responsibility of an application protocol to specify the protocol slots in which such strings can appear, the entities that are expected to enforce the rules governing such strings, and at what points during protocol processing or interface handling the rules need to be enforced. See Section 6 of [RFC8264] for guidelines on using PRECIS profiles in applications.

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

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

Applying the rules for any given PRECIS profile is not necessarily an idempotent procedure for all code points. Therefore, an implementation SHOULD apply the rules repeatedly until the output string is stable; if the output string does not stabilize after reapplying the rules three (3) additional times after the first application, the implementation SHOULD terminate application of the rules and reject the input string as invalid.

6. Migration

The rules defined in this specification differ slightly from those defined by the SASLprep specification [RFC4013] (but not from [RFC7613]). In order to smooth the process of migrating from SASLprep to the approach defined herein, the following sections describe these differences, along with their implications for migration, in more detail.

6.1. Usernames

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

- 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. "Å¿" (LATIN SMALL LETTER LONG S, U+017F) is compatibility equivalent to "s" (LATIN SMALL LETTER S, U+0073).

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

3. "ï¬" (LATIN SMALL LIGATURE FI, U+FB01) is compatibility equivalent to "f" (LATIN SMALL LETTER F, U+0066) and "i" (LATIN SMALL LETTER I, U+0069).

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

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

- 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 code points, which correspond to the code points from the PRECIS "M" category defined under Section 9.13 of [RFC8264]. For migration purposes, the operator might want to remove from usernames any code points contained in the PRECIS "M" category (e.g., SOFT HYPHEN (U+00AD)). Because these code points would have been "mapped to nothing" in Stringprep, in practice a user would not notice the difference if, upon migration to PRECIS, the code points are removed.

- SASLprep allowed uppercase and titlecase code points, whereas the UsernameCaseMapped profile maps uppercase and titlecase code

points to their lowercase equivalents (by contrast, the UsernameCasePreserved profile matches SASLprep in this regard). For migration purposes, the operator can use either the UsernameCaseMapped profile (thus losing the case information) or the UsernameCasePreserved profile (thus ignoring case difference when comparing usernames).

6.2. Passwords

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

- 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 accepts when comparing passwords. A few examples might suffice to indicate the nature of the problem:

  1. "Å¿" (LATIN SMALL LETTER LONG S, U+017F) is compatibility equivalent to "s" (LATIN SMALL LETTER S, U+0073).

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

  3. "ï¬" (LATIN SMALL LIGATURE FI, U+FB01) is compatibility equivalent to "f" (LATIN SMALL LETTER F, U+0066) and "i" (LATIN SMALL LETTER I, U+0069).

  Under SASLprep, the use of NFKC also handled the mapping of fullwidth and halfwidth code points to their decomposition mappings. Although it is expected that code points with compatibility equivalents are rare in existing passwords, some passwords that matched when SASLprep was used might no longer work when the rules in this specification are applied.

- SASLprep mapped the "characters commonly mapped to nothing" (from Appendix B.1 of [RFC3454]) to nothing, whereas the PRECIS FreeformClass entirely disallows such code points, which correspond to the code points from the PRECIS "M" category defined under Section 9.13 of [RFC8264]. In practice, this change will probably have no effect on comparison, but user-oriented software

might reject such code points instead of ignoring them during password preparation.

7. IANA Considerations

IANA has made the updates described below.

7.1. UsernameCaseMapped Profile

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

Name: UsernameCaseMapped.

Base Class: IdentifierClass.

Applicability: Usernames in security and application protocols.

Replaces: The SASLprep profile of Stringprep.

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

Additional Mapping Rule: None.

Case Mapping Rule: Map uppercase and titlecase code points 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: Section 3.3 of RFC 8265.

7.2. UsernameCasePreserved Profile

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

Name: UsernameCasePreserved.

Base Class: IdentifierClass.

Applicability: Usernames in security and application protocols.

Replaces: The SASLprep profile of Stringprep.

Width Mapping Rule: Map fullwidth and halfwidth code points 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: Section 3.4 of RFC 8265.

7.3. OpaqueString Profile

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

Name: OpaqueString.

Base Class: FreeformClass.

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

Replaces: The SASLprep profile of Stringprep.

Width Mapping Rule: None.

Additional Mapping Rule: Map non-ASCII space code points to SPACE (U+0020).

Case Mapping Rule: None.

Normalization Rule: NFC.

Directionality Rule: None.

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

Specification: Section 4.2 of RFC 8265.

7.4. Stringprep Profile

The Stringprep specification [RFC3454] did not provide for entries in the "Stringprep Profiles" registry to have any state except "Current" or "Not Current". Because RFC 7613 obsoleted RFC 4013, which registered the SASLprep profile of Stringprep, IANA previously marked that profile as "Not Current" and cited RFC 7613 as an additional reference. IANA has modified the profile so that the current document is now cited as the additional reference.

8. Security Considerations

8.1. Password/Passphrase Strength

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

8.2. Password/Passphrase Comparison

In systems that conform to modern best practices for security, verification of passwords during authentication will not use the comparison defined in Section 4.2.3. Instead, because the system performs cryptographic calculations to verify the password, it will prepare the password as defined in Section 4.2.1 and enforce the rules as defined in Section 4.2.2 before performing the relevant calculations.

8.3. Identifier Comparison

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

8.4. Reuse of PRECIS

The security considerations described in [RFC8264] apply to the IdentifierClass and FreeformClass string classes used in this document for usernames and passwords, respectively.

8.5. Reuse of Unicode

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

9. References

9.1. Normative References

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

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

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

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

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8264] Saint-Andre, P. and M. Blanchet, "PRECIS Framework: Preparation, Enforcement, and Comparison of Internationalized Strings in Application Protocols", RFC 8264, DOI 10.17487/RFC8264, October 2017, <https://www.rfc-editor.org/info/rfc8264>.

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

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

9.2. Informative References

[Err1812] RFC Errata, Erratum ID 1812, RFC 4013, <https://www.rfc-editor.org/errata/eid1812>.

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

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

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

[RFC3987] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, DOI 10.17487/RFC3987, January 2005, <https://www.rfc-editor.org/info/rfc3987>.

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

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

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

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

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

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

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

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

[RFC7613] Saint-Andre, P. and A. Melnikov, "Preparation, Enforcement, and Comparison of Internationalized Strings Representing Usernames and Passwords", RFC 7613, DOI 10.17487/RFC7613, August 2015, <https://www.rfc-editor.org/info/rfc7613>.

[RFC7616] Shekh-Yusef, R., Ed., Ahrens, D., and S. Bremer, "HTTP Digest Access Authentication", RFC 7616, DOI 10.17487/RFC7616, September 2015, <https://www.rfc-editor.org/info/rfc7616>.

[RFC7617] Reschke, J., "The 'Basic' HTTP Authentication Scheme", RFC 7617, DOI 10.17487/RFC7617, September 2015, <https://www.rfc-editor.org/info/rfc7617>.

[RFC7622] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Address Format", RFC 7622, DOI 10.17487/RFC7622, September 2015, <https://www.rfc-editor.org/info/rfc7622>.

[RFC8266] Saint-Andre, P., "Preparation, Enforcement, and Comparison of Internationalized Strings Representing Nicknames", RFC 8266, DOI 10.17487/RFC8266, October 2017, <https://www.rfc-editor.org/info/rfc8266>.

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

Appendix A. Changes from RFC 7613

The following changes were made from [RFC7613].

- Corrected the order of operations for the UsernameCaseMapped profile to ensure consistency with [RFC8264].

- In accordance with working group discussions and updates to [RFC8264], removed the use of the Unicode toCaseFold() operation in favor of the Unicode toLowerCase() operation.
- Modified the presentation (but not the content) of the rules.
- Removed UTF-8 as a mandatory encoding, because that is a matter for the application.
- Clarified several editorial matters.
- Updated references.

See [RFC7613] for a description of the differences from [RFC4013].

Acknowledgements

Thanks to Christian Schudt and Sam Whited for their bug reports and feedback.

See [RFC7613] for acknowledgements related to the specification that this document supersedes.

Authors' Addresses

Peter Saint-Andre
Jabber.org
P.O. Box 787
Parker, CO 80134
United States of America

Phone: +1 720 256 6756
Email: stpeter@jabber.org
URI: https://www.jabber.org/

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

Email: Alexey.Melnikov@isode.com