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.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 4
2. Terminology 6
3. Preparation, Enforcement, and Comparison 7
4. String Classes 7
4.1. Overview 7
4.2. IdentifierClass 9
4.2.1. Valid 9
4.2.2. Contextual Rule Required 9
4.2.3. Disallowed 10
4.2.4. Unassigned 10
4.2.5. Examples 10
4.3. FreeformClass 11
4.3.1. Valid 11
4.3.2. Contextual Rule Required 11
4.3.3. Disallowed 12
4.3.4. Unassigned 12
4.3.5. Examples 12
5. Profiles 12
5.1. Profiles Must Not Be Multiplied Beyond Necessity 13
5.2. Rules 13
5.2.1. Width Mapping Rule 13
5.2.2. Additional Mapping Rule 14
5.2.3. Case Mapping Rule 14
5.2.4. Normalization Rule 15
5.2.5. Directionality Rule 15
5.3. A Note about Spaces 16
6. Applications 17
6.1. How to Use PRECIS in Applications 17
6.2. Further Excluded Characters 17
6.3. Building Application-Layer Constructs 18
7. Order of Operations 19

8. Code Point Properties 19
9. Category Definitions Used to Calculate Derived Property . . . 22
9.1. LetterDigits (A) 22
9.2. Unstable (B) 22
9.3. IgnorableProperties (C) 23
9.4. IgnorableBlocks (D) 23
9.5. LDH (E) 23
9.6. Exceptions (F) 23
9.7. BackwardCompatible (G) 23
9.8. JoinControl (H) 23
9.9. OldHangulJamo (I) 23
9.10. Unassigned (J) 24
9.11. ASCII7 (K) 24
9.12. Controls (L) 24
9.13. PrecisIgnorableProperties (M) 24
9.14. Spaces (N) 24
9.15. Symbols (O) 24
9.16. Punctuation (P) 25
9.17. HasCompat (Q) 25
9.18. OtherLetterDigits (R) 25
10. Guidelines for Designated Experts 25
11. IANA Considerations 26
11.1. PRECIS Derived Property Value Registry 26
11.2. PRECIS Base Classes Registry 26
11.3. PRECIS Profiles Registry 27
12. Security Considerations 29
12.1. General Issues 29
12.2. Use of the IdentifierClass 30
12.3. Use of the FreeformClass 30
12.4. Local Character Set Issues 30
12.5. Visually Similar Characters 30
12.6. Security of Passwords 32
13. Interoperability Considerations 33
13.1. Encoding 33
13.2. Character Sets 33
13.3. Unicode Versions 34
13.4. Potential Changes to Handling of Certain Unicode Code Points 34
14. References 35
14.1. Normative References 35
14.2. Informative References 35
14.3. URIs 38
Appendix A. Acknowledgements 38
Authors' Addresses 39

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:

- o 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.

- o 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.

- o 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

- o 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.

- o 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

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

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

4.2.3. Disallowed

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

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

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

- o Space characters, i.e., the Spaces ("N") category defined under Section 9.14.

- o Symbol characters, i.e., the Symbols ("O") category defined under Section 9.15.

- o Punctuation characters, i.e., the Punctuation ("P") category defined under Section 9.16.

- o 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.

- o 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).

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

4.3.3. Disallowed

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

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

- o 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).

- o 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).

- o 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.

- o 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).

- o 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.

- o 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:

- o Would an existing PRECIS string class or profile solve the problem? If not, why not? (See Section 5.1 for related considerations.)

- o Is the problem being addressed by this profile well-defined?

- o Does the specification define what kinds of applications are involved and the protocol elements to which this profile applies?

- o Is the profile clearly defined?

- o 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?

- o Are the width mapping, case mapping, additional mappings, normalization, and directionality rules appropriate for the intended use?

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

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

- o 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/

PRECIS
Internet-Draft
Intended status: Standards Track
Expires: March 6, 2016

P. Saint-Andre
&yet

September 3, 2015

Preparation, Enforcement, and Comparison of Internationalized Strings Representing Nicknames
draft-ietf-precis-nickname-19

Abstract

This document describes methods for handling Unicode strings representing memorable, human-friendly names (variously called "nicknames", "display names", or "petnames") for people, devices, accounts, websites, and other entities.

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 March 6, 2016.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 2
1.1. Overview 2
1.2. Terminology 3
2. Nickname Profile 3
2.1. Rules 3
2.2. Preparation 4
2.3. Enforcement 5
2.4. Comparison 5
3. Examples 5
4. Use in Application Protocols 6
5. IANA Considerations 7
6. Security Considerations 8
6.1. Reuse of PRECIS 8
6.2. Reuse of Unicode 8
6.3. Visually Similar Characters 8
7. References 8
7.1. Normative References 8
7.2. Informative References 9
7.3. URIs 10
Appendix A. Acknowledgements 10
Author’s Address 10

1. Introduction

1.1. Overview

A number of technologies and applications provide the ability for a person to choose a memorable, human-friendly name in a communications context, or to set such a name for another entity entity such as a device, account, contact, or website. Such names are variously called "nicknames" (e.g., in chatroom applications), "display names" (e.g., in Internet mail), or "petnames" (see [1]); for consistency, these are all called "nicknames" in this document.

Nicknames are commonly supported in technologies for textual chatrooms, e.g., Internet Relay Chat [RFC2811] and multi-party chat technologies based on the Extensible Messaging and Presence Protocol (XMPP) [RFC6120] [XEP-0045], the Message Session Relay Protocol (MSRP) [RFC4975] [I-D.ietf-simple-chat], and Centralized Conferencing (XCON) [RFC5239] [I-D.boulton-xcon-session-chat]. Recent chatroom technologies also allow internationalized nicknames because they support characters from outside the ASCII range [RFC20], typically by means of the Unicode character set [Unicode]. Although such nicknames tend to be used primarily for display purposes, they are sometimes used for programmatic purposes as well (e.g., kicking users or avoiding nickname conflicts).

A similar usage enables a person to set their own preferred display name or to set a preferred display name for another user (e.g., the "display-name" construct in the Internet message format [RFC5322] and [XEP-0172] in XMPP).

Memorable, human-friendly names are also used in contexts other than personal messaging, such as names for devices (e.g., in a network visualization application), websites (e.g., for bookmarks in a web browser), accounts (e.g., in a web interface for a list of payees in a bank account), people (e.g., in a contact list application), and the like.

The rules specified in this document can be applied in all of the foregoing contexts.

To increase the likelihood that memorable, human-friendly names will work in ways that make sense for typical users throughout the world, this document defines rules for preparing, enforcing, and comparing internationalized nicknames.

1.2. Terminology

Many important terms used in this document are defined in [RFC7564], [RFC6365], and [Unicode].

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 RFC 2119 [RFC2119].

2. Nickname Profile

2.1. Rules

The following rules apply within the Nickname profile of the PRECIS FreeformClass.

1. Width Mapping Rule: There is no width-mapping rule (such a rule is not necessary because width mapping is performed as part of normalization using NFKC as specified below).

2. Additional Mapping Rule: The additional mapping rule consists of the following sub-rules.

    1. 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.

2. Any instances of the ASCII space character at the beginning or end of a nickname MUST be removed (e.g., "stpeter " is mapped to "stpeter").

3. Interior sequences of more than one ASCII space character MUST be mapped to a single ASCII space character (e.g., "St Peter" is mapped to "St Peter").

3. Case Mapping Rule: Uppercase and titlecase characters MUST be mapped to their lowercase equivalents 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]). In applications that prohibit conflicting nicknames, this rule helps to reduce the possibility of confusion by ensuring that nicknames differing only by case (e.g., "stpeter" vs. "StPeter") would not be presented to a human user at the same time.

4. Normalization Rule: The string MUST be normalized using Unicode Normalization Form KC (NFKC). Because NFKC is more "aggressive" in finding matches than other normalization forms (in the terminology of Unicode, it performs both canonical and compatibility decomposition before recomposing code points), this rule helps to reduce the possibility of confusion by increasing the number of characters that would match (e.g., U+2163 ROMAN NUMERAL FOUR would match the combination of U+0049 LATIN CAPITAL LETTER I and U+0056 LATIN CAPITAL LETTER V).

5. Directionality Rule: There is no directionality rule. The "Bidi Rule" (defined in [RFC5893]) and similar rules are unnecessary and inapplicable to nicknames, because it is perfectly acceptable for a given nickname to be presented differently in different layout systems (e.g., a user interface that is configured to handle primarily a right-to-left script vs. an interface that is configured to handle primarily a left-to-right script), as long as the presentation is consistent in any given layout system.

2.2. Preparation

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

2.3. Enforcement

An entity that performs enforcement according to this profile MUST prepare a string as described in Section 2.2 section and MUST also apply the rules specified in Section 2.1. The rules MUST be applied in the order shown.

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

2.4. Comparison

An entity that performs comparison of two strings according to this profile MUST prepare each string and enforce the rules as specified in Section 2.2 and Section 2.3. The two strings are to be considered equivalent if they are an exact octet-for-octet match (sometimes called "bit-string identity").

3. Examples

The following examples illustrate a small number of nicknames that are consistent with the format defined above, along with the output string resulting from application of the PRECIS rules (note that the characters < and > are used to delineate the actual nickname and are not part of the nickname strings).

Table 1: A sample of legal nicknames

#	Nickname	Output for Comparison
1	<Foo>	<foo>
2	<foo>	<foo>
3	<Foo Bar>	<foo bar>
4	<foo bar>	<foo bar>
5	<Σ>	GREEK SMALL LETTER SIGMA (U+03C3)
6	<σ>	GREEK SMALL LETTER SIGMA (U+03C3)
7	<ς>	GREEK SMALL LETTER FINAL SIGMA (U+03C2)
8	<♚>	BLACK CHESS KING (U+265A)
9	<Richard Ⅳ>	<richard iv>

Regarding examples 5, 6, and 7: applying Unicode Default Case Folding to GREEK CAPITAL LETTER SIGMA (U+03A3) results in GREEK SMALL LETTER SIGMA (U+03C3), and doing so during comparison would result in matching the nicknames 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 nicknames in examples 5 and 7 or examples 6 and 7 would not be matched. Regarding example 8: symbol characters such as BLACK CHESS KING (U+265A) are allowed by the PRECIS FreeformClass and thus can be used in nicknames. Regarding example 9: applying Unicode Default Case Folding to ROMAN NUMERAL FOUR (U+2163) results in SMALL ROMAN NUMERAL FOUR (U+2173), and applying NFKC to SMALL ROMAN NUMERAL FOUR (U+2173) results in LATIN SMALL LETTER I (U+0069) LATIN SMALL LETTER V (U+0086).

4. Use in Application Protocols

This specification defines only the PRECIS-based rules for handling of nickname strings. It is the responsibility of an application protocol (e.g., MSRP, XCON, or XMPP) or application definition to specify the protocol slots in which nickname strings can appear, the entities that are expected to enforce the rules governing nickname 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 nickname strings (rules regarding the minimum or maximum length of nicknames, further restrictions on allowable characters or character ranges, safeguards to mitigate the effects of visually similar characters, etc.).

Naturally, application protocols can also specify rules governing the actual use of nicknames in applications (reserved nicknames, authorization requirements for using nicknames, whether certain nicknames can be prohibited, handling of duplicates, the relationship between nicknames and underlying identifiers such as SIP URIs or Jabber IDs, etc.).

Entities that enforce the rules specified in this document are encouraged to be liberal in what they accept by following this procedure:

1. Where possible, map characters (e.g, through width mapping, additional mapping, case mapping, or normalization) and accept the mapped string.

2. If mapping is not possible (e.g., because a character is disallowed in the FreeformClass), reject the string.

5. IANA Considerations

The IANA shall add the following entry to the PRECIS Profiles Registry:

Name: Nickname.

Base Class: FreeformClass.

Applicability: Nicknames in messaging and text conferencing technologies; petnames for devices, accounts, and people; and other uses of nicknames or petnames.

Replaces: None.

Width Mapping Rule: None (handled via NFKC).

Additional Mapping Rule: Map non-ASCII space characters to ASCII space, strip leading and trailing space characters, map interior sequences of multiple space characters to a single ASCII space.

Case Mapping Rule: Map uppercase and titlecase characters to lowercase using Unicode Default Case Folding.

Normalization Rule: NFKC.

Directionality Rule: None.

Enforcement: To be specified by applications.

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

6. Security Considerations

6.1. Reuse of PRECIS

The security considerations described in [RFC7564] apply to the "FreeformClass" string class used in this document for nicknames.

6.2. Reuse of Unicode

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

6.3. Visually Similar Characters

[RFC7564] describes some of the security considerations related to visually similar characters, also called "confusable characters" or "confusables".

Although the mapping rules defined under Section 2 of this document are designed in part to reduce the possibility of confusion about nicknames, this document does not provide more detailed recommendations regarding the handling of visually similar characters, such as those provided in [UTS39].

7. References

7.1. 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.

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

[UTS39] The Unicode Consortium, "Unicode Technical Standard #39: Unicode Security Mechanisms", November 2013, <http://unicode.org/reports/tr39/>.

[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/>.

7.2. Informative References

[I-D.boulton-xcon-session-chat] Barnes, M., Boulton, C., and S. Loreto, "Chatrooms within a Centralized Conferencing (XCON) System", draft-boulton-xcon-session-chat-08 (work in progress), July 2011.

[I-D.ietf-simple-chat] Niemi, A., Garcia, M., and G. Sandbakken, "Multi-party Chat Using the Message Session Relay Protocol (MSRP)", draft-ietf-simple-chat-18 (work in progress), January 2013.

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

[RFC2811] Kalt, C., "Internet Relay Chat: Channel Management", RFC 2811, April 2000.

[RFC4975] Campbell, B., Mahy, R., and C. Jennings, "The Message Session Relay Protocol (MSRP)", RFC 4975, September 2007.

[RFC5239] Barnes, M., Boulton, C., and O. Levin, "A Framework for Centralized Conferencing", RFC 5239, June 2008.

[RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, October 2008.

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

[XEP-0045] Saint-Andre, P., "Multi-User Chat", XSF XEP 0045, February 2012.

[XEP-0172] Saint-Andre, P. and V. Mercier, "User Nickname", XSF XEP 0172, March 2012.

7.3. URIs

[1] http://www.skyhunter.com/marcs/petnames/IntroPetNames.html

Appendix A. Acknowledgements

Thanks to Kim Alvefur, Mary Barnes, Ben Campbell, Dave Cridland, Miguel Garcia, Salvatore Loreto, Enrico Marocco, Matt Miller, and Yoshiro YONEYA for their reviews and comments.

Paul Kyzivat and Melinda Shore reviewed the document for the General Area Review Team and Operations Directorate, respectively.

During IESG review, Ben Campbell and Kathleen Moriarty provided comments that led to further improvements.

Thanks to Matt Miller as document shepherd, Pete Resnick and Andrew Sullivan as IANA designated experts, Marc Blanchet and Alexey Melnikov as working group chairs, and Barry Leiba as area director.

The author wishes to acknowledge Cisco Systems, Inc., for employing him during his work on earlier draft versions of this document.

Author’s Address

Peter Saint-Andre
&yet

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

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

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 3
2. Terminology 4
3. Usernames 5
3.1. Definition 5
3.2. UsernameCaseMapped Profile 6
3.2.1. Preparation 6
3.2.2. Enforcement 6
3.2.3. Comparison 7
3.3. UsernameCasePreserved Profile 7
3.3.1. Preparation 7
3.3.2. Enforcement 7
3.3.3. Comparison 8
3.4. Case Mapping vs. Case Preservation 8
3.5. Application-Layer Constructs 9
3.6. Examples 9
4. Passwords 11
4.1. Definition 11
4.2. OpaqueString Profile 12
4.2.1. Preparation 12
4.2.2. Enforcement 12
4.2.3. Comparison 13
4.3. Examples 13
5. Use in Application Protocols 14
6. Migration 15
6.1. Usernames 15
6.2. Passwords 16
7. IANA Considerations 17
7.1. UsernameCaseMapped Profile 17
7.2. UsernameCasePreserved Profile 18
7.3. OpaqueString Profile 19
8. Security Considerations 19
8.1. Password/Passphrase Strength 19
8.2. Identifier Comparison 19
8.3. Reuse of PRECIS 20
8.4. Reuse of Unicode 20
9. References 20
9.1. Normative References 20
9.2. Informative References 21
Appendix A. Differences from RFC 4013 22
Appendix B. Acknowledgements 23
Authors' Addresses 23

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.

- o 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.

- o SASLprep mapped the "characters commonly mapped to nothing" from Appendix B.1 of [RFC3454]) to nothing, whereas the PRECIS IdentifierClass entirely disallows most of these characters, which correspond to the code points from the "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.

- o SASLprep allowed uppercase and titlecase characters, whereas the UsernameCaseMapped profile maps uppercase and titlecase characters to their lowercase equivalents (by contrast, the UsernameCasePreserved profile matches SASLprep in this regard). For migration purposes, the operator can 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:

- o SASLprep specified the use of Unicode Normalization Form KC (NFKC), whereas the OpaqueString profile employs Unicode Normalization Form C (NFC). Because NFKC is more aggressive about finding matches than NFC, in practice this change is unlikely to cause significant problems and indeed has the security benefit of probably resulting in fewer false positives when comparing passwords. A few examples might suffice to indicate the nature of the problem:

  1. 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.

- o SASLprep mapped the "characters commonly mapped to nothing" from Appendix B.1 of [RFC3454]) to nothing, whereas the PRECIS FreeformClass entirely disallows such characters, which correspond to the code points from the "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.

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

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

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

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

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

XMPP
Internet-Draft
Obsoletes: 6122 (if approved)
Intended status: Standards Track
Expires: December 13, 2015

P. Saint-Andre
&yet

June 11, 2015

Extensible Messaging and Presence Protocol (XMPP): Address Format
draft-ietf-xmpp-6122bis-24

Abstract

This document defines the address format for the Extensible Messaging and Presence Protocol (XMPP), including support for code points outside the ASCII range. This document obsoletes RFC 6122.

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 December 13, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 2
2. Terminology 3
3. Addresses 3
3.1. Fundamentals 3
3.2. Domainpart 5
3.3. Localpart 7
3.4. Resourcepart 8
3.5. Examples 9
4. Enforcement in JIDs and JID Parts 13
5. Internationalization Considerations 15
6. IANA Considerations 15
6.1. Stringprep Profiles Registry 15
7. Security Considerations 16
7.1. Reuse of PRECIS 16
7.2. Reuse of Unicode 16
7.3. Address Spoofing 16
8. Conformance Requirements 18
9. References 20
9.1. Normative References 20
9.2. Informative References 21
Appendix A. Differences from RFC 6122 24
Appendix B. Acknowledgements 25
Author’s Address 25

1. Introduction

The Extensible Messaging and Presence Protocol (XMPP) [RFC6120] is an application profile of the Extensible Markup Language [XML] for streaming XML data in close to real time between any two or more network-aware entities. The address format for XMPP entities was originally developed in the Jabber open-source community in 1999, first described by [XEP-0029] in 2002, and then defined canonically by [RFC3920] in 2004 and [RFC6122] in 2011.

As specified in RFC 3920 and RFC 6122, the XMPP address format used the "stringprep" technology for preparation and comparison of non-ASCII characters [RFC3454]. Following the movement of internationalized domain names away from stringprep, this document defines the XMPP address format in a way that no longer depends on stringprep (see the PRECIS problem statement [RFC6885]). Instead, this document builds upon the internationalization framework defined by the IETF’s PRECIS Working Group [RFC7564].

Although every attempt has been made to ensure that the characters allowed in Jabber Identifiers (JIDs) under Stringprep are still allowed and handled in the same way under PRECIS, there is no

guarantee of strict backward compatibility because of changes in Unicode and the fact that PRECIS handling is based on Unicode properties, not a hardcoded table of characters. Because it is possible that previously-valid JIDs might no longer be valid (or previously-invalid JIDs might now be valid), operators of XMPP services are advised to perform careful testing before migrating accounts and other data (see Section 6.1 of [I-D.ietf-precis-saslprepbis] for guidance).

This document obsoletes RFC 6122.

2. Terminology

Many important terms used in this document are defined in [RFC7564], [RFC5890], [RFC6120], [RFC6365], and [Unicode].

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. Addresses

3.1. Fundamentals

An XMPP entity is anything that can communicate using XMPP. For historical reasons, the network address of an XMPP entity is called a Jabber Identifier ("JID"). A valid JID is a string of Unicode code points [Unicode], encoded using UTF-8 [RFC3629], and structured as an ordered sequence of localpart, domainpart, and resourcepart, where the first two parts are demarcated by the '@' character used as a separator and the last two parts are similarly demarcated by the '/' character (e.g., <juliet@example.com/balcony>).

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

```
jid          = [ localpart "@" ] domainpart [ "/" resourcepart ]
localpart    = 1*1023(userbyte)
               ;
               ; a "userbyte" is a byte used to represent a
               ; UTF-8 encoded Unicode code point that can be
               ; contained in a string that conforms to the
               ; UsernameCaseMapped profile of the PRECIS
               ; IdentifierClass defined in
               ; draft-ietf-precis-saslprepbis
               ;
domainpart   = IP-literal / IPv4address / ifqdn
               ;
               ; the "IPv4address" and "IP-literal" rules are
               ; defined in RFC 3986 and RFC 6874 respectively,
               ; and the first-match-wins (a.k.a. "greedy")
               ; algorithm described in Appendix B of RFC 3986
               ; applies to the matching process
               ;
ifqdn        = 1*1023(domainbyte)
               ;
               ; a "domainbyte" is a byte used to represent a
               ; UTF-8 encoded Unicode code point that can be
               ; contained in a string that conforms to RFC 5890
               ;
resourcepart = 1*1023(opaquebyte)
               ;
               ; an "opaquebyte" is a byte used to represent a
               ; UTF-8 encoded Unicode code point that can be
               ; contained in a string that conforms to the
               ; OpaqueString profile of the PRECIS
               ; FreeformClass defined in
               ; draft-ietf-precis-saslprepbis
               ;
```

All JIDs are based on the foregoing structure. However, note that the formal syntax provided above does not capture all of the rules and restrictions that apply to JIDs, which are described below.

Each allowable portion of a JID (localpart, domainpart, and resourcepart) is 1 to 1023 octets in length, resulting in a maximum total size (including the '@' and '/' separators) of 3071 octets.

Implementation Note: The length limits on JIDs and parts of JIDs are based on octets (bytes), not characters. UTF-8 encoding can result in more than one octet per character.

Implementation Note: When dividing a JID into its component parts, an implementation needs to match the separator characters '@' and

'/' before applying any transformation algorithms, which might decompose certain Unicode code points to the separator characters.

Implementation Note: Reuse of the IP-literal rule from [RFC6874] implies that IPv6 addresses are enclosed within square brackets (i.e., beginning with '[' and ; ending with ']'), which was not the case with the definition of the XMPP address format in [RFC3920] but which was changed in [RFC6122]. Also note that the IP-literal rule was updated between [RFC3986] and [RFC6874] to optionally add a zone identifier to any literal address.

This document defines the native format for JIDs; see [RFC5122] for information about the representation of a JID as a Uniform Resource Identifier (URI) [RFC3986] or Internationalized Resource Identifier (IRI) [RFC3987] and the extraction of a JID from an XMPP URI or IRI.

3.2. Domainpart

The domainpart of a JID is that portion which remains once the following parsing steps are taken:

1. Remove any portion from the first '/' character to the end of the string (if there is a '/' character present).

2. Remove any portion from the beginning of the string to the first '@' character (if there is a '@' character present).

This parsing order is important, as illustrated by example 15 in Section 3.5.

The domainpart is the primary identifier and is the only REQUIRED element of a JID (a mere domainpart is a valid JID). Typically a domainpart identifies the "home" server to which clients connect for XML routing and data management functionality. However, it is not necessary for an XMPP domainpart to identify an entity that provides core XMPP server functionality (e.g., a domainpart can identify an entity such as a multi-user chat service [XEP-0045], a publish-subscribe service [XEP-0060], or a user directory).

The domainpart for every XMPP service MUST be a fully-qualified domain name (FQDN), an IPv4 address, an IPv6 address, or an unqualified hostname (i.e., a text label that is resolvable on a local network).

Informational Note: The term "fully-qualified domain name" is not well defined. In [RFC1034] it is also called an absolute domain name, and the two terms are associated in [RFC1535]. The earliest use of the term can be found in [RFC1123]. References to those

older specifications ought not to be construed as limiting the characters of a fully-qualified domain name to the ASCII range; for example, [RFC5890] mentions that a fully-qualified domain name can contain one or more U-labels.

Interoperability Note: Domainparts that are IP addresses might not be accepted by other services for the purpose of server-to-server communication, and domainparts that are unqualified hostnames cannot be used on public networks because they are resolvable only on a local network.

If the domainpart includes a final character considered to be a label separator (dot) by [RFC1034], this character MUST be stripped from the domainpart before the JID of which it is a part is used for the purpose of routing an XML stanza, comparing against another JID, or constructing an XMPP URI or IRI [RFC5122]. In particular, such a character MUST be stripped before any other canonicalization steps are taken.

In general, the content of a domainpart is an Internationalized Domain Name ("IDN") as described in the specifications for Internationalized Domain Names in Applications (commonly called "IDNA2008"), and a domainpart is an "IDNA-aware domain name slot" as defined in [RFC5890].

After any and all normalization, conversion, and mapping of code points as well as encoding of the string as UTF-8, a domainpart MUST NOT be zero octets in length and MUST NOT be more than 1023 octets in length. (Naturally, the length limits of [RFC1034] apply, and nothing in this document is to be interpreted as overriding those more fundamental limits.)

Detailed rules and considerations for preparation, enforcement, and comparison are provided in the following sections.

3.2.1. Preparation

An entity that prepares a string for inclusion in an XMPP domainpart slot MUST ensure that the string consists only of Unicode code points that are allowed in NR-LDH labels or U-labels as defined in [RFC5890]. This implies that the string MUST NOT include A-labels as defined in [RFC5890]; each A-label MUST be converted to a U-label during preparation of a string for inclusion in a domainpart slot. In addition, the string MUST be encoded as UTF-8 [RFC3629].

3.2.2. Enforcement

An entity that performs enforcement in XMPP domainpart slots MUST prepare a string as described in the previous section and MUST also apply the normalization, case-mapping, and width-mapping rules defined in [RFC5892].

> The order in which the rules are applied for IDNA2008 (see [RFC5892] and [RFC5895]) is different from the order for localparts and resourceparts as described under Section 3.3 and Section 3.4.

3.2.3. Comparison

An entity that performs comparison of two strings before or after their inclusion in XMPP domainpart slots MUST prepare each string and enforce the normalization, case-mapping, and width-mapping 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. Localpart

The localpart of a JID is an optional identifier placed before the domainpart and separated from the latter by the '@' character. Typically a localpart uniquely identifies the entity requesting and using network access provided by a server (i.e., a local account), although it can also represent other kinds of entities (e.g., a chat room associated with a multi-user chat service [XEP-0045]). The entity represented by an XMPP localpart is addressed within the context of a specific domain (i.e., <localpart@domainpart>).

The localpart of a JID MUST NOT be zero octets in length and MUST NOT be more than 1023 octets in length. This rule is to be enforced after any normalization and mapping of code points as well as encoding of the string as UTF-8.

The localpart of a JID is an instance of the UsernameCaseMapped profile of the PRECIS IdentifierClass, which is specified in [I-D.ietf-precis-saslprepbis]. The rules and considerations provided in that specification MUST be applied to XMPP localparts.

> Implementation Note: XMPP uses the Simple Authentication and Security Layer (SASL) [RFC4422] for authentication. At the time of this writing, some SASL mechanisms use SASLprep [RFC4013] for handling of usernames and passwords; in the future these SASL mechanisms will likely transition to the use of PRECIS-based handling rules as specified in [I-D.ietf-precis-saslprepbis]. For

a detailed discussion about the implications of that transition (including the potential need to modify or remove certain characters in the underlying account database), see both Section 6.1 and Appendix A of [I-D.ietf-precis-saslprepbis].

3.3.1. Further Excluded Characters

In XMPP, the following characters are explicitly disallowed in XMPP localparts even though they are allowed by the IdentifierClass base class and the UsernameCaseMapped profile:

" U+0022 (QUOTATION MARK)

& U+0026 (AMPERSAND)

' U+0027 (APOSTROPHE)

/ U+002F (SOLIDUS)

: U+003A (COLON)

< U+003C (LESS-THAN SIGN)

> U+003E (GREATER-THAN SIGN)

@ U+0040 (COMMERCIAL AT)

Implementation Note: An XMPP-specific method for escaping the foregoing characters (along with U+0020, i.e., ASCII SPACE) has been defined in the JID Escaping specification [XEP-0106].

3.4. Resourcepart

The resourcepart of a JID is an optional identifier placed after the domainpart and separated from the latter by the '/' character. A resourcepart can modify either a <localpart@domainpart> address or a mere <domainpart> address. Typically a resourcepart uniquely identifies a specific connection (e.g., a device or location) or object (e.g., an occupant in a multi-user chat room [XEP-0045]) belonging to the entity associated with an XMPP localpart at a domain (i.e., <localpart@domainpart/resourcepart>).

XMPP entities SHOULD consider resourceparts to be opaque strings and SHOULD NOT impute meaning to any given resourcepart. In particular:

- Use of the '/' character as a separator between the domainpart and the resourcepart does not imply that XMPP addresses are hierarchical in the way that, say, HTTP URIs are hierarchical (see

[RFC3986] for general discussion); thus for example an XMPP address of the form <localpart@domainpart/foo/bar> does not identify a resource "bar" that exists below a resource "foo" in a hierarchy of resources associated with the entity "localpart@domainpart".

- o The ’@’ character is allowed in the resourcepart and is often used in the "handle" shown in XMPP chatrooms [XEP-0045]. For example, the JID <room@chat.example.com/user@host> describes an entity who is an occupant of the room <room@chat.example.com> with a handle of <user@host>. However, chatroom services do not necessarily check such an asserted handle against the occupant’s real JID.

The resourcepart of a JID MUST NOT be zero octets in length and MUST NOT be more than 1023 octets in length. This rule is to be enforced after any normalization and mapping of code points as well as encoding of the string as UTF-8.

The resourcepart of a JID is an instance of the OpaqueString profile of the PRECIS FreeformClass, which is specified in [I-D.ietf-precis-saslprepbis]. The rules and considerations provided in that specification MUST be applied to XMPP resourceparts.

3.4.1. Applicability to XMPP Extensions

In some contexts, it might be appropriate to apply more restrictive rules to the preparation, enforcement, and comparison of XMPP resourceparts. For example, in XMPP Multi-User Chat [XEP-0045] it might be appropriate to apply the rules specified in [I-D.ietf-precis-nickname]. However, the application of more restrictive rules is out of scope for resourceparts in general and is properly defined in specifications for the relevant XMPP extensions.

3.5. Examples

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

Table 1: A sample of legal JIDs

#	JID	Notes
1	<juliet@example.com>	A "bare JID"
2	<juliet@example.com/foo>	A "full JID"
3	<juliet@example.com/foo bar>	Single space in resourcepart
4	<juliet@example.com/foo@bar>	At sign in resourcepart
5	<foo\20bar@example.com>	Single space in localpart, as optionally escaped using the XMPP "JID Escaping" extension
6	<fussball@example.com>	Another bare JID
7	<fußball@example.com>	The third character is LATIN SMALL LETTER SHARP S (U+00DF)
8	<π@example.com>	A localpart of GREEK SMALL LETTER PI (U+03C0)
9	<Σ@example.com/foo>	A localpart of GREEK CAPITAL LETTER SIGMA (U+03A3)
10	<σ@example.com/foo>	A localpart of GREEK SMALL LETTER SIGMA (U+03C3)
11	<ς@example.com/foo>	A localpart of GREEK SMALL LETTER FINAL SIGMA (U+03C2)
12	<king@example.com/&#x265A>;	A resourcepart of the Unicode character BLACK CHESS KING (U+265A)
13	<example.com>	A domainpart
14	<example.com/foobar>	A domainpart and resourcepart
15	<a.example.com/b@example.net>	A domainpart followed by a resourcepart that contains an at sign

Several points are worth noting. Regarding examples 6 and 7: although in German the character esszett (LATIN SMALL LETTER SHARP S, U+00DF) can mostly be used interchangeably with the two characters "ss", the localparts 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 9, 10, and 11: 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 JIDs in examples 9 and 10; however, because the PRECIS mapping rules do not account for the special status of GREEK SMALL LETTER FINAL SIGMA (U+03C2), the JIDs in examples 9 and 11 or examples 10 and 11 would not be matched. Regarding example 12: symbol characters such as BLACK CHESS KING (U+265A) are allowed by the PRECIS FreeformClass and thus can be used in resourceparts. Regarding examples 14 and 15: JIDs consisting of a domainpart and resourcepart are rarely seen in the wild, but are allowed according to the XMPP address format. Example 15 illustrates the need for careful extraction of the domainpart as described in the first paragraph of Section 3.2.

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

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

#	Non-JID string	Notes
16	<"juliet"@example.com>	Quotation marks (U+0022) in localpart
17	<foo bar@example.com>	Space (U+0020) in localpart
18	<juliet@example.com/ foo>	Leading space in resourcepart
19	<@example.com/>	Zero-length localpart and resourcepart
20	<henryⅣ@example.com>	The sixth character is ROMAN NUMERAL FOUR (U+2163)
21	<♚@example.com>	A localpart of BLACK CHESS KING (U+265A)
22	<juliet@>	A localpart without a domainpart
23	</foobar>	A resourcepart without a domainpart

Here again, several points are worth noting. Regarding example 17, even though ASCII SPACE (U+0020) is disallowed in the PRECIS IdentifierClass, it can be escaped to "\20" in XMPP localparts by using the JID Escaping rules defined in [XEP-0106], as illustrated by example 4 in Table 1. Regarding example 20, 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 21: symbol characters such as BLACK CHESS KING (U+265A) are not allowed in the PRECIS IdentifierClass; however, both of the non-ASCII characters in examples 20 and 21 are allowed in the PRECIS Freeform class and therefore in the XMPP resourcepart (as illustrated for U+265A by example 12 in Table 1). Regarding examples 22 and 23: the domainpart is required in a JID.

4. Enforcement in JIDs and JID Parts

Enforcement entails applying all of the rules specified in this document. Enforcement of the XMPP address format rules is the responsibility of XMPP servers. Although XMPP clients SHOULD prepare complete JIDs and parts of JIDs in accordance with this document before including them in protocol slots within XML streams, XMPP servers MUST enforce the rules wherever possible and reject stanzas and other XML elements that violate the rules (for stanzas, by returning a <jid-malformed/> error to the sender as described in Section 8.3.3.8 of [RFC6120]).

Entities that enforce the rules specified in this document are encouraged to be liberal in what they accept by following this procedure:

1. Where possible, map characters (e.g, through width mapping, additional mapping, special mapping, case mapping, or normalization) and accept the mapped string.

2. If mapping is not possible (e.g., because a character is disallowed in the FreeformClass), reject the string and return a <jid-malformed/> error.

Enforcement applies to complete JIDs and to parts of JIDs. To facilitate implementation, this document defines the concepts of "JID slot", "localpart slot", and "resourcepart slot" (similar to the concept of a "domain name slot" for IDNA2008 defined in Section 2.3.2.6 of [RFC5890]):

JID Slot: An XML element or attribute explicitly designated in XMPP or in XMPP extensions for carrying a complete JID.

Localpart Slot: An XML element or attribute explicitly designated in XMPP or in XMPP extensions for carrying the localpart of a JID.

Resourcepart Slot: An XML element or attribute explicitly designated in XMPP or in XMPP extensions for carrying the resourcepart of a JID.

A server is responsible for enforcing the address format rules when receiving protocol elements from clients where the server is expected to handle such elements directly or to use them for purposes of routing a stanza to another domain or delivering a stanza to a local entity; two examples from [RFC6120] are the ’to’ attribute on XML stanzas (which is a JID slot used by XMPP servers for routing of outbound stanzas) and the <resource/> child of the <bind/> element (which is a resourcepart slot used by XMPP servers for binding of a

resource to an account for routing of stanzas between the server and a particular client). An example from [RFC6121] is the 'jid' attribute of the roster <item/> element.

A server is not responsible for enforcing the rules when the protocol elements are intended for communication among other entities, typically within the payload of a stanza that the server is merely routing to another domain or delivering to a local entity. Two examples are the 'initiator' attribute in the Jingle extension [XEP-0166] (which is a JID slot used for client-to-client coordination of multimedia sessions) and the 'nick' attribute in the Multi-User Chat extension [XEP-0045] (which is a resourcepart slot used for administrative purposes in the context of XMPP chatrooms). In such cases, the entities involved SHOULD enforce the rules themselves and not depend on the server to do so, and client implementers need to understand that not enforcing the rules can lead to a degraded user experience or to security vulnerabilities. However, when an add-on service (e.g., a multi-user chat service) handles a stanza directly, it ought to enforce the rules as well, as defined in the relevant specification for that type of service.

This document does not provide an exhaustive list of JID slots, localpart slots, or resourcepart slots. However, implementers of core XMPP servers are advised to consider as JID slots at least the following elements and attributes when they are handled directly or used for purposes of routing to another domain or delivering to a local entity:

- The 'from' and 'to' stream attributes and the 'from' and 'to' stanza attributes [RFC6120].

- The 'jid' attribute of the roster <item/> element for contact list management [RFC6121].

- The 'value' attribute of the <item/> element for Privacy Lists [RFC3921] [XEP-0016] when the value of the 'type' attribute is "jid".

- The 'jid' attribute of the <item/> element for Service Discovery defined in [XEP-0030].

- The <value/> element for Data Forms [XEP-0004], when the 'type' attribute is "jid-single" or "jid-multi".

- The 'jid' attribute of the <conference/> element for Bookmark Storage [XEP-0048].

- o The <JABBERID/> of the <vCard/> element for vCard 3.0 [XEP-0054] and the <uri/> child of the <impp/> element for vCard 4.0 [XEP-0292] when the XML character data identifies an XMPP URI [RFC5122].

- o The 'from' attribute of the <delay/> element for Delayed Delivery [XEP-0203].

- o The 'jid' attribute of the <item/> element for the Blocking Command [XEP-0191].

- o The 'from' and 'to' attributes of the <result/> and <verify/> elements for Server Dialback [RFC3921], [XEP-0220].

- o The 'from' and 'to' attributes of the <iq/>, <message/>, and <presence/> elements for the Jabber Component Protocol [XEP-0114].

Developers of XMPP clients and specialized XMPP add-on services are advised to check the appropriate specifications for JID slots, localpart slots, and resourcepart slots in XMPP protocol extensions such as Service Discovery [XEP-0030], Multi-User Chat [XEP-0045], Publish-Subscribe [XEP-0060], SOCKS5 Bytestreams [XEP-0065], In-Band Registration [XEP-0077], Roster Item Exchange [XEP-0144], and Jingle [XEP-0166].

5. Internationalization Considerations

XMPP applications MUST support IDNA2008 for domainparts as described under Section 3.2, the "UsernameCaseMapped" profile for localparts as described under Section 3.3, and the "OpaqueString" profile for resourceparts as described under Section 3.4. This enables XMPP addresses to include a wide variety of characters outside the ASCII range. Rules for enforcement of the XMPP address format are provided in [RFC6120] and specifications for various XMPP extensions.

> Interoperability Note: For backward compatibility, many existing XMPP implementations and deployments support IDNA2003 [RFC3490] for domainparts, and the stringprep [RFC3454] profiles Nodeprep and Resourceprep [RFC3920] for localparts and resourceparts.

6. IANA Considerations

6.1. Stringprep Profiles Registry

The Stringprep specification [RFC3454] did not provide for entries in the Stringprep Profiles registry to have any state anything except current or not current. Because this document obsoletes RFC 6122, which registered the "Nodeprep" and "Resourceprep" profiles of

Stringprep, IANA is requested to mark those profiles as not current and to cite this document as an additional reference.

7. Security Considerations

7.1. Reuse of PRECIS

The security considerations described in [RFC7564] apply to the "IdentifierClass" and "FreeformClass" base string classes used in this document for XMPP localparts and resourceparts, respectively. The security considerations described in [RFC5890] apply to internationalized domain names, which are used here for XMPP domainparts.

7.2. Reuse of Unicode

The security considerations described in [UTS39] apply to the use of Unicode characters in XMPP addresses.

7.3. Address Spoofing

There are two forms of address spoofing: forging and mimicking.

7.3.1. Address Forging

In the context of XMPP technologies, address forging occurs when an entity is able to generate an XML stanza whose ’from’ address does not correspond to the account credentials with which the entity authenticated onto the network (or an authorization identity provided during negotiation of SASL authentication [RFC4422] as described in [RFC6120]). For example, address forging occurs if an entity that authenticated as "juliet@im.example.com" is able to send XML stanzas from "nurse@im.example.com" or "romeo@example.net".

Address forging is difficult in XMPP systems, given the requirement for sending servers to stamp ’from’ addresses and for receiving servers to verify sending domains via server-to-server authentication (see [RFC6120]). However, address forging is possible if:

- o A poorly implemented server ignores the requirement for stamping the ’from’ address. This would enable any entity that authenticated with the server to send stanzas from any localpart@domainpart as long as the domainpart matches the sending domain of the server.

- o An actively malicious server generates stanzas on behalf of any registered account at the domain or domains hosted at that server.

Therefore, an entity outside the security perimeter of a particular server cannot reliably distinguish between JIDs of the form <localpart@domainpart> at that server and thus can authenticate only the domainpart of such JIDs with any level of assurance. This specification does not define methods for discovering or counteracting the kind of poorly implemented or rogue servers just described. However, the end-to-end authentication or signing of XMPP stanzas could help to mitigate this risk, since it would require the rogue server to generate false credentials for signing or encryption of each stanza, in addition to modifying 'from' addresses.

7.3.2. Address Mimicking

Address mimicking occurs when an entity provides legitimate authentication credentials for and sends XML stanzas from an account whose JID appears to a human user to be the same as another JID. Because many characters are visually similar, it is relatively easy to mimic JIDs in XMPP systems. As one simple example, the localpart "ju1iet" (using the Arabic numeral one as the third character) might appear the same as the localpart "juliet" (using lowercase "L" as the third character).

As explained in [RFC5890], [RFC7564], [UTR36], and [UTS39], there is no straightforward solution to the problem of visually similar characters. Furthermore, IDNA and PRECIS technologies do not attempt to define such a solution. As a result, XMPP domainparts, localparts, and resourceparts could contain such characters, leading to security vulnerabilities such as the following:

- A domainpart is always employed as one part of an entity's address in XMPP. One common usage is as the address of a server or server-side service, such as a multi-user chat service [XEP-0045]. The security of such services could be compromised based on different interpretations of the internationalized domainpart; for example, a user might authorize a malicious entity at a fake server to view the user's presence information, or a user could join chatrooms at a fake multi-user chat service.

- A localpart can be employed as one part of an entity's address in XMPP. One common usage is as the username of an instant messaging user; another is as the name of a multi-user chat room; and many other kinds of entities could use localparts as part of their addresses. The security of such services could be compromised based on different interpretations of the internationalized localpart; for example, a user entering a single internationalized localpart could access another user's account information, or a user could gain access to a hidden or otherwise restricted chat room or service.

- o A resourcepart can be employed as one part of an entity's address in XMPP. One common usage is as the name for an instant messaging user's connected resource; another is as the nickname of a user in a multi-user chat room; and many other kinds of entities could use resourceparts as part of their addresses. The security of such services could be compromised based on different interpretations of the internationalized resourcepart; for example, two or more confusable resources could be bound at the same time to the same account (resulting in inconsistent authorization decisions in an XMPP application that uses full JIDs), or a user could send a private message to someone other than the intended recipient in a multi-user chat room.

XMPP services and clients are strongly encouraged to define and implement consistent policies regarding the registration, storage, and presentation of visually similar characters in XMPP systems. In particular, service providers and software implementers are strongly encouraged to apply the policies recommended in [RFC7564].

8. Conformance Requirements

This section describes a protocol feature set that summarizes the conformance requirements of this specification (similar feature sets are provided for XMPP in [RFC6120] and [RFC6121]). The summary is purely informational and the conformance keywords of [RFC2119] as used here are intended only to briefly describe the referenced normative text from the body of this specification. This feature set is appropriate for use in software certification, interoperability testing, and implementation reports. For each feature, this section provides the following information:

- o A human-readable name
- o An informational description
- o A reference to the particular section of this document that normatively defines the feature
- o Whether the feature applies to the Client role, the Server role, or both (where "N/A" signifies that the feature is not applicable to the specified role)
- o Whether the feature MUST or SHOULD be implemented, where the capitalized terms are to be understood as described in [RFC2119]

The feature set specified here provides a basis for interoperability testing and follows the spirit of a proposal made by Larry Masinter within the IETF's NEWTRK Working Group in 2005 [INTEROP].

Feature: address-domain-length

Description: Ensure that the domainpart of an XMPP address is at least one octet in length and at most 1023 octets in length, and that it conforms to the underlying length limits of the DNS.

Section: Section 3.2

Roles: Server MUST, client SHOULD.

Feature: address-domain-prep

Description: Ensure that the domainpart of an XMPP address conforms to IDNA2008, that it contains only NR-LDH labels and U-labels (not A-labels), and that all uppercase and titlecase code points are mapped to their lowercase equivalents.

Section: Section 3.2

Roles: Server MUST, client SHOULD.

Feature: address-localpart-length

Description: Ensure that the localpart of an XMPP address is at least one octet in length and at most 1023 octets in length.

Section: Section 3.3

Roles: Server MUST, client SHOULD.

Feature: address-localpart-prep

Description: Ensure that the localpart of an XMPP address conforms to the "UsernameCaseMapped" profile of the PRECIS IdentifierClass.

Section: Section 3.3

Roles: Server MUST, client SHOULD.

Feature: address-resource-length

Description: Ensure that the resourcepart of an XMPP address is at least one octet in length and at most 1023 octets in length.

Section: Section 3.4

Roles: Server MUST, client SHOULD.

Feature: address-resource-prep

Description: Ensure that the resourcepart of an XMPP address conforms to the "OpaqueString" profile of the PRECIS FreeformClass.

Section: Section 3.4

Roles: Server MUST, client SHOULD.

9. References

9.1. Normative References

[I-D.ietf-precis-saslprepbis] Saint-Andre, P. and A. Melnikov, "Username and Password Preparation Algorithms", draft-ietf-precis-saslprepbis-18 (work in progress), May 2015.

[RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, November 1987.

[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.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC5234] Crocker, D., Ed. 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.

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

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

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

[RFC6874] Carpenter, B., Cheshire, S., and R. Hinden, "Representing IPv6 Zone Identifiers in Address Literals and Uniform Resource Identifiers", RFC 6874, February 2013.

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

[UTR36] The Unicode Consortium, "Unicode Technical Report #36: Unicode Security Considerations", November 2013, <http://www.unicode.org/reports/tr36/>.

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

9.2. Informative References

[I-D.ietf-precis-nickname] Saint-Andre, P., "Preparation and Comparison of Nicknames", draft-ietf-precis-nickname-17 (work in progress), April 2015.

[INTEROP] Masinter, L., "Formalizing IETF Interoperability Reporting", Work in Progress, October 2005.

[RFC1123] Braden, R., "Requirements for Internet Hosts - Application and Support", STD 3, RFC 1123, October 1989.

[RFC1535] Gavron, E., "A Security Problem and Proposed Correction With Widely Deployed DNS Software", RFC 1535, October 1993.

[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.

See Section 1 for an explanation of why the normative reference to an obsoleted specification is needed.

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

[RFC3921] Saint-Andre, P., Ed., "Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence", RFC 3921, October 2004.

[RFC3987] Duerst, M. and M. Suignard, "Internationalized Resource Identifiers (IRIs)", RFC 3987, January 2005.

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

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

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

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

[RFC6121] Saint-Andre, P., "Extensible Messaging and Presence Protocol (XMPP): Instant Messaging and Presence", RFC 6121, March 2011.

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

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

[XEP-0004] Eatmon, R., Hildebrand, J., Miller, J., Muldowney, T., and P. Saint-Andre, "Data Forms", XSF XEP 0004, August 2007.

[XEP-0016] Millard, P. and P. Saint-Andre, "Privacy Lists", XSF XEP 0016, February 2007.

[XEP-0029]
Kaes, C., "Definition of Jabber Identifiers (JIDs)", XSF XEP 0029, October 2003.

[XEP-0030]
Hildebrand, J., Millard, P., Eatmon, R., and P. Saint-Andre, "Service Discovery", XSF XEP 0030, June 2008.

[XEP-0045]
Saint-Andre, P., "Multi-User Chat", XSF XEP 0045, February 2012.

[XEP-0048]
Blackman, R., Millard, P., and P. Saint-Andre, "Bookmarks", XSF XEP 0048, November 2007.

[XEP-0054]
Saint-Andre, P., "vcard-temp", XSF XEP 0054, July 2008.

[XEP-0060]
Millard, P., Saint-Andre, P., and R. Meijer, "Publish-Subscribe", XSF XEP 0060, July 2010.

[XEP-0065]
Smith, D., Miller, M., Saint-Andre, P., and J. Karneges, "SOCKS5 Bytestreams", XSF XEP 0065, April 2011.

[XEP-0077]
Saint-Andre, P., "In-Band Registration", XSF XEP 0077, January 2012.

[XEP-0106]
Hildebrand, J. and P. Saint-Andre, "JID Escaping", XSF XEP 0106, June 2007.

[XEP-0114]
Saint-Andre, P., "Jabber Component Protocol", XSF XEP 0114, March 2005.

[XEP-0144]
Saint-Andre, P., "Roster Item Exchange", XSF XEP 0144, August 2005.

[XEP-0166]
Ludwig, S., Beda, J., Saint-Andre, P., McQueen, R., Egan, S., and J. Hildebrand, "Jingle", XSF XEP 0166, December 2009.

[XEP-0191]
Saint-Andre, P., "Blocking Command", XSF XEP 0191, July 2012.

[XEP-0203]
Saint-Andre, P., "Delayed Delivery", XSF XEP 0203, September 2009.

[XEP-0220]
Miller, J., Saint-Andre, P., and P. Hancke, "Server Dialback", XSF XEP 0220, August 2012.

[XEP-0292]
Saint-Andre, P. and S. Mizzi, "vCard4 Over XMPP", XSF XEP 0292, October 2011.

[XML] Maler, E., Yergeau, F., Sperberg-McQueen, C., Paoli, J., and T. Bray, "Extensible Markup Language (XML) 1.0 (Fifth Edition)", World Wide Web Consortium Recommendation REC-xml-20081126, November 2008, <http://www.w3.org/TR/2008/REC-xml-20081126>.

Appendix A. Differences from RFC 6122

Based on consensus derived from working group discussion, implementation and deployment experience, and formal interoperability testing, the following substantive modifications were made from RFC 6122.

- Changed domainpart preparation to use IDNA2008 (instead of IDNA2003).
- Changed localpart preparation to use the UsernameCaseMapped profile of the PRECIS IdentifierClass (instead of the Nodeprep profile of Stringprep).
- Changed resourcepart preparation to use the OpaqueString profile of the PRECIS FreeformClass (instead of the Resourceprep profile of Stringprep).
- Specified that internationalized labels within domainparts must be U-labels (instead of "should be" U-labels).
- Specified that fullwidth and halfwidth characters must be mapped to their decomposition mappings (previously handled through the use of NFKC).

- o Specified the use of Unicode Normalization Form C (instead of Unicode Normalization Form KC as specified in the Nodeprep and Resourceprep profiles of Stringprep).

- o Specified that servers must enforce the address formatting rules.

Appendix B. Acknowledgements

Thanks to Ben Campbell, Dave Cridland, Miguel Garcia, Joe Hildebrand, Jonathan Lennox, Matt Miller, Florian Schmaus, Sam Whited, and Florian Zeitz for their input during working group discussion.

Dan Romascanu completed a helpful review on behalf of the General Area Review Team.

During IESG review, Alissa Cooper, Brian Haberman, and Barry Leiba provided comments that led to improvements in the document.

Thanks also to Matt Miller in his role as document shepherd, Joe Hildebrand in his role as working group chair, and Ben Campbell in his role as sponsoring Area Director.

The author wishes to acknowledge Cisco Systems, Inc., for employing him during his work on earlier draft versions of this document.

Author’s Address

Peter Saint-Andre
&yet

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

PRECIS
Internet-Draft
Intended status: Informational
Expires: March 6, 2015

P. Saint-Andre
&yet

September 2, 2014

PRECIS Codepoint Table
draft-saintandre-precis-codepoints-00

Abstract

This document provides a codepoint table, current at the time of publication, that results from applying the rules defined in the PRECIS framework specification to the code points of version 7.0 of the Unicode specification.

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 March 6, 2015.

Copyright Notice

Copyright (c) 2014 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction 2
2. Table 2
3. IANA Considerations 32
4. Security Considerations 33
5. Normative References 33
Appendix A. Acknowledgements 33
Author's Address 33

1. Introduction

This document provides a codepoint table, current at the time of publication, that results from applying the rules defined in the PRECIS framework specification [I-D.ietf-precis-framework] to the code points of version 7.0 of the Unicode specification [Unicode7.0.0].

2. Table

By applying the property calculation rules from [I-D.ietf-precis-framework] to the code points U+0000 to U+10FFFF in Unicode 7.0.0, the following table is produced. The table is in in Unicode Character Database (UCD) format and the columns of the table are as follows:

1. The code point or codepoint range.

2. The assignment for the code point or range, where the value is one of PVALID, DISALLOWED, UNASSIGNED, CONTEXTO, CONTEXTJ, or FREE_PVAL.

3. The name or names for the code point or range.

This table is non-normative, is provided only for illustrative purposes, and applies only to Unicode 7.0.0, not to past or future versions of Unicode. Please note that the strings displayed in the third column are not necessarily the formal name of the code point (as defined in [Unicode7.0.0]) because the fixed width of the RFC format necessitated truncation of many names.

```
0000..001F  ; DISALLOWED  # <control>
0020        ; FREE_PVAL   # SPACE
0021..007E  ; PVALID      # EXCLAM MARK..TILDE
007F..009F  ; DISALLOWED  # <control>
00A0..00AC  ; FREE_PVAL   # NO-BREAK SPACE..NOT SIGN
00AD        ; DISALLOWED  # SOFT HYPH
00AE..00B6  ; FREE_PVAL   # REGISTERED SIGN..PILCROW SIGN
```

```
00B7        ; CONTEXTO    # MIDDLE DOT
00B8..00BF  ; FREE_PVAL   # CEDILLA..INV QUEST IND
00C0..00D6  ; PVALID      # LAT CAP LET A W GRAV..LAT CAP O
00D7        ; FREE_PVAL   # MULTIPLICATION SIGN
00D8..00F6  ; PVALID      # LAT CAP LET O W STROKE..LAT SM
00F7        ; FREE_PVAL   # DIVISION SIGN
00F8..0131  ; PVALID      # LAT SM LET O W STROKE..LAT SM LET
0132..0133  ; FREE_PVAL   # LAT CAP LIG IJ..LAT SM LIB IJ
0134..013E  ; PVALID      # LAT CAP LET J W CIRCUM..LAT SM LET
013F..0140  ; FREE_PVAL   # LAT CAP LET L W MID DOT..LAT SM LET
0141..0148  ; PVALID      # LAT CAP LET L W STROKE..LAT SM LET
0149        ; FREE_PVAL   # LAT SM LET N PRECEDED BY APOS
014A..017E  ; PVALID      # LAT CAP LET ENG..LAT SM LET Z W CA
017F        ; FREE_PVAL   # LAT SM LET LONG S
0180..01C3  ; PVALID      # LAT SM LET B W STROKE..LAT LET RETR
01C4..01CC  ; FREE_PVAL   # LAT CAP LET DZ W CARON..LAT SM
01CD..01F0  ; PVALID      # LAT CAP LET A W CARON..LAT SM LET J
01F1..01F3  ; FREE_PVAL   # LAT CAP LET DZ..LAT SM LET DZ
01F4..02AF  ; PVALID      # LAT CAP LET G W ACUTE..LAT SM
02B0..02B8  ; FREE_PVAL   # MOD LET SM H..MOD LET SM Y
02B9..02C1  ; PVALID      # MOD LET PRIME..MOD LET REV GLOT ST
02C2..02C5  ; FREE_PVAL   # MOD LET L ARROW..MOD LET D ARROW
02C6..02D1  ; PVALID      # MOD LET CIRCUM ACC..MOD LET HALF TR
02D2..02EB  ; FREE_PVAL   # MOD LET CENT R HALF RING..MOD LET Y
02EC        ; PVALID      # MOD LET VOICING
02ED        ; FREE_PVAL   # MOD LET UNASPIRATED
02EE        ; PVALID      # MOD LET DOUBLE APOS
02EF..02FF  ; FREE_PVAL   # MOD LET LOW D ARR..MOD LET LOW L AR
0300..034E  ; PVALID      # COMB GRAVE ACCENT..COMB UP ARROW BE
034F        ; DISALLOWED  # COMB GRAPHEME JOINER
0350..0374  ; PVALID      # COMB RIGHT ARROWHEAD..GREEK NUM SIG
0375        ; CONTEXTO    # GREEK LOW NUM SIGN
0376..0377  ; PVALID      # GR CAP LET PAMPHYLIAN DIGAMMA..GR S
0378..0379  ; UNASSIGNED  # <reserved>..<reserved>
037A        ; FREE_PVAL   # GR YPOGEGRAMMENI..GR SM REV DOT LUN
037B..037D  ; PVALID      # GR SM REV LUN SIG..GR SM REV DOT LU
037E        ; FREE_PVAL   # GREEK QUEST MARK
037F..0383  ; UNASSIGNED  # <reserved>..<reserved>
0384..0385  ; FREE_PVAL   # GREEK TONOS..GREEK DIALYTIKA TONOS
0386        ; PVALID      # GR CAP LET ALPHA W TONOS
0387        ; FREE_PVAL   # GREEK ANO TELEIA
0388..038A  ; PVALID      # GR CAP LET EPSILON W TONOS..GR CAP
038B        ; UNASSIGNED  # <reserved>
038C        ; PVALID      # GREEK CAP LET OMICRON W TONOS
038D        ; UNASSIGNED  # <reserved>
038E..03A1  ; PVALID      # GR CAP LET EPSILON W TONOS..GR CAP
03A2        ; UNASSIGNED  # <reserved>
03A3..03CF  ; PVALID      # GREEK CAP LET SIGMA..GR CAP
```

```
03D0..03D2  ; FREE_PVAL   # GR BETA SYM..GR UPSILON W HOOK
03D3..03D4  ; FREE_PVAL   # GR UPSILON W ACUTE AND HOOK..GR UP
03D5..03D6  ; FREE_PVAL   # GR PHI SYM..GR PI SYM
03D7..03EF  ; PVALID      # GR KAI SYM..COPT SM LET DEI
03F0..03F2  ; FREE_PVAL   # GR KAPPA SYM..GR LUNATE SIGMA
03F3        ; PVALID      # GREEK LET YOT
03F4..03F6  ; FREE_PVAL   # GR CAP THETA..GR REV LUNATE EPSILON
03F7..03F8  ; PVALID      # GR CAP LET SHO..GR SM LET SHO
03F9        ; FREE_PVAL   # GREEK CAP LUNATE SIGMA SYM
03FA..0481  ; PVALID      # GR CAP LET SAN..CYR SML LET KOPPA
0482        ; FREE_PVAL   # CYR THOUSANDS SIGN
0483..0487  ; PVALID      # COMB CYR TITLO..COMB CYR POK
0488..0489  ; FREE_PVAL   # COMB CYR HUNDRED THOUSANDS SIGN..C
048A..0527  ; PVALID      # CYR CAP LET SH I W TAIL..CYR S
0528..0530  ; UNASSIGNED  # <reserved>..<reserved>
0531..0556  ; PVALID      # ARM CAP LET AYB..ARM CAP LET FEH
0557..0558  ; UNASSIGNED  # <reserved>..<reserved>
0559        ; PVALID      # ARM MOD LET LEFT HALF RING
055A..055F  ; FREE_PVAL   # ARM APOS..ARM ABBREV
0560        ; UNASSIGNED  # <reserved>
0561..0586  ; PVALID      # ARM SM LET AYB..ARMENIAN SM LE
0587        ; FREE_PVAL   # ARM SM LIG ECH YIWN
0588        ; UNASSIGNED  # <reserved>
0589..058A  ; FREE_PVAL   # ARMENIAN FULL STOP..ARMENIAN HYPH
058B..058E  ; UNASSIGNED  # <reserved>..<reserved>
058F        ; FREE_PVAL   # ARMENIAN DRAM SIGN
0590        ; UNASSIGNED  # <reserved>
0591..05BD  ; PVALID      # HEBR ACC ETNAHTA..HEBR PNT ME
05BE        ; FREE_PVAL   # HEBR PUNCT MAQAF
05BF        ; PVALID      # HEBR PNT RAFE
05C0        ; FREE_PVAL   # HEBR PUNCT PASEQ
05C1..05C2  ; PVALID      # HEBR PNT SHIN DOT..HEBR PNT SIN DOT
05C3        ; FREE_PVAL   # HEBR PUNCT SOF PASUQ
05C4..05C5  ; PVALID      # HEBR MARK UP DOT..HEBR MARK LOW DOT
05C6        ; FREE_PVAL   # HEBR PUNCT NUN HAFUKHA
05C7        ; PVALID      # HEBR PNT QAMATS QATAN
05C8..05CF  ; UNASSIGNED  # <reserved>..<reserved>
05D0..05EA  ; PVALID      # HEBR LET ALEF..HEBR LET TAV
05EB..05EF  ; UNASSIGNED  # <reserved>..<reserved>
05F0..05F2  ; PVALID      # HEBR LIG YIDDISH DOUBLE VAV..HEBR L
05F3..05F4  ; CONTEXTO    # HEBR PUNCT GERESH..HEBR PUNCTUATIO
05F5..05FF  ; UNASSIGNED  # <reserved>..<reserved>
0600..0604  ; DISALLOWED  # ARAB NUM SIGN..ARAB SIGN SAM
0605        ; UNASSIGNED  # <reserved>..<reserved>
0606..060F  ; FREE_PVAL   # AR-IND CUBE ROOT..ARAB SIGN MISRA
0610..061A  ; PVALID      # ARAB SIGN SALLALLAHOU ALAYHE ..AR
061B        ; FREE_PVAL   # ARAB SEMICOLON
061C        ; DISALLOWED  # ARAB LET MARK
```

```
061D..061D  ; UNASSIGNED  # <reserved>..<reserved>
061E..061F  ; FREE_PVAL   # ARAB TRIPLE DOT PUNCT MARK..ARAB Q
0620..063F  ; PVALID      # ARAB LET KASH..ARAB LET FARSI YEH
0640        ; DISALLOWED  # ARAB TATWEEL
0641..065F  ; PVALID      # ARAB LET FEH..ARAB WAVY HAMZA BEL
0660..0669  ; CONTEXTO    # AR-IND DIG ZERO..AR-IND DIG
066A..066D  ; FREE_PVAL   # ARAB PCT SIGN..ARAB FIVE PNTED STA
066E..0674  ; PVALID      # ARAB LET DOTLESS BEH..ARAB LET HIG
0675..0678  ; FREE_PVAL   # ARAB LET HIGH HAMZA ALEF..ARAB LET
0679..06D3  ; PVALID      # ARAB LET TTEH..ARAB LET YEH BARREE
06D4        ; FREE_PVAL   # ARAB FULL STOP
06D5..06DC  ; PVALID      # ARAB LET AE..ARAB SM HIGH SEEN
06DD        ; DISALLOWED  # ARAB END OF AYAH
06DE        ; FREE_PVAL   # ARAB START OF RUB EL HIZB
06DF..06E8  ; PVALID      # ARAB SM HIGH ROUNDED ZERO..ARAB SM
06E9        ; FREE_PVAL   # ARAB PLACE OF SAJDAH
06EA..06EF  ; PVALID      # ARAB EMPTY CENTRE LOW STOP..ARAB LET
06F0..06F9  ; CONTEXTO    # EXT AR-IND DIG ZERO..EXT A
06FA..06FF  ; PVALID      # ARAB LET SHEEN W DOT BEL..ARAB
0700..070D  ; FREE_PVAL   # SYR END OF PARA..SYR HARKLEAN AST
070E        ; UNASSIGNED  # <reserved>
070F        ; DISALLOWED  # SYR ABBR MARK
0710..074A  ; PVALID      # SYR LET ALAPH..SYR BARREKH
074B..074C  ; UNASSIGNED  # <reserved>..<reserved>
074D..07B1  ; PVALID      # SYR LET SOGDIAN ZHAIN..THAANA LET N
07B2..07BF  ; UNASSIGNED  # <reserved>..<reserved>
07C0..07F5  ; PVALID      # NKO DIG ZERO..NKO LOW TONE APOS
07F6..07F9  ; FREE_PVAL   # NKO SYM OO DENNEN..NKO EXCLAMATI
07FA        ; DISALLOWED  # NKO LAJANYALAN
07FB..07FF  ; UNASSIGNED  # <reserved>..<reserved>
0800..082D  ; PVALID      # SAMAR LET ALAF..SAMAR MARK NEQUDA
082E..082F  ; UNASSIGNED  # <reserved>..<reserved>
0830..083E  ; FREE_PVAL   # SAMAR PUNCT NEQUDAA..SAMAR PUN
083F        ; UNASSIGNED  # <reserved>
0840..085B  ; PVALID      # MANDAIC LET HALQA..MANDAIC GEM
085C..085D  ; UNASSIGNED  # <reserved>..<reserved>
085E        ; FREE_PVAL   # MANDAIC PUNCTUATION
085F..089F  ; UNASSIGNED  # <reserved>..<reserved>
08A0        ; PVALID      # ARAB LET BEH W SM V BEL
08A1        ; UNASSIGNED  # <reserved>
08A2..08AC  ; PVALID      # ARAB LET JEEM W 2 DOTS AB..ARAB
08AD..08E3  ; UNASSIGNED  # <reserved>..<reserved>
08E4..08FE  ; PVALID      # ARAB CURLY FATHA..ARAB DAMMA W
08FF        ; UNASSIGNED  # <reserved>
0900..0963  ; PVALID      # DEVAN SIGN INV CANDRABINDU..DEVAN V
0964..0965  ; FREE_PVAL   # DEVAN DANDA..DEVAN DOUBLE DANDA
0966..096F  ; PVALID      # DEVAN DIG ZERO..DEVAN DIG NINE
0970        ; FREE_PVAL   # DEVAN ABBR SIGN
```

```
0971..0977  ; PVALID      # DEVAN SIGN HIGH SPACING DOT..DEVAN
0978        ; UNASSIGNED  # <reserved>
0979..097F  ; PVALID      # DEVAN SIGN HIGH SPACING DOT..DEVAN
0980        ; UNASSIGNED  # <reserved>
0981..0983  ; PVALID      # BENG SIGN CANDRABINDU..BENG SIGN VIS
0984        ; UNASSIGNED  # <reserved>
0985..098C  ; PVALID      # BENG LET A..BENG LET VOC L
098D..098E  ; UNASSIGNED  # <reserved>..<reserved>
098F..0990  ; PVALID      # BENG LET E..BENG LET AI
0991..0992  ; UNASSIGNED  # <reserved>..<reserved>
0993..09A8  ; PVALID      # BENG LET O..BENG LET NA
09A9        ; UNASSIGNED  # <reserved>
09AA..09B0  ; PVALID      # BENG LET PA..BENG LET RA
09B1        ; UNASSIGNED  # <reserved>
09B2        ; PVALID      # BENG LET LA
09B3..09B5  ; UNASSIGNED  # <reserved>..<reserved>
09B6..09B9  ; PVALID      # BENG LET SHA..BENG LET HA
09BA..09BB  ; UNASSIGNED  # <reserved>..<reserved>
09BC..09C4  ; PVALID      # BENG SIGN NUKTA..BENG VOW SIGN VOCAL
09C5..09C6  ; UNASSIGNED  # <reserved>..<reserved>
09C7..09C8  ; PVALID      # BENG VOW SIGN E..BENG VOW SIGN AI
09C9..09CA  ; UNASSIGNED  # <reserved>..<reserved>
09CB..09CE  ; PVALID      # BENG VOW SIGN O..BENG LET KHANDA
09CF..09D6  ; UNASSIGNED  # <reserved>..<reserved>
09D7        ; PVALID      # BENG AU LEN MARK
09D8..09DB  ; UNASSIGNED  # <reserved>..<reserved>
09DC..09DD  ; PVALID      # BENG LET RRA..BENG LET RHA
09DE        ; UNASSIGNED  # <reserved>
09DF..09E3  ; PVALID      # BENG LET YYA..BENG VOW SIG
09E4..09E5  ; UNASSIGNED  # <reserved>..<reserved>
09E6..09F1  ; PVALID      # BENG DIG ZERO..BENG LET RA W L
09F2..09FB  ; FREE_PVAL   # BENG RUPEE MARK..BENG GANDA MARK
09FC..0A00  ; UNASSIGNED  # <reserved>..<reserved>
0A01..0A03  ; PVALID      # GURMUKHI SIGN ADAK BINDI..GURMUKHI
0A04        ; UNASSIGNED  # <reserved>
0A05..0A0A  ; PVALID      # GURMUKHI LET A..GURMUKHI LET UU
0A0B..0A0E  ; UNASSIGNED  # <reserved>..<reserved>
0A0F..0A10  ; PVALID      # GURMUKHI LET EE..GURMUKHI LET AI
0A11..0A12  ; UNASSIGNED  # <reserved>..<reserved>
0A13..0A28  ; PVALID      # GURMUKHI LET OO..GURMUKHI LET NA
0A29        ; UNASSIGNED  # <reserved>
0A2A..0A30  ; PVALID      # GURMUKHI LET PA..GURMUKHI LET RA
0A31        ; UNASSIGNED  # <reserved>
0A32..0A33  ; PVALID      # GURMUKHI LET LA..GURMUKHI LET LLA
0A34        ; UNASSIGNED  # <reserved>
0A35.0A36   ; PVALID      # GURMUKHI LET VA..GURMUKHI LET SHA
0A37        ; UNASSIGNED  # <reserved>
0A38..0A39  ; PVALID      # GURMUKHI LET SA..GURMUKHI LET HA
```

```
0A3A..0A3B  ; UNASSIGNED  # <reserved>..<reserved>
0A3C        ; PVALID      # GURMUKHI SIGN NUKTA
0A3D        ; UNASSIGNED  # <reserved>
0A3E..0A42  ; PVALID      # GURMUKHI VOW SIGN AA..GURMUKHI V
0A43..0A46  ; UNASSIGNED  # <reserved>..<reserved>
0A47..0A48  ; PVALID      # GURMUKHI VOW SIGN EE..GURMUKHI V
0A49..0A4A  ; UNASSIGNED  # <reserved>..<reserved>
0A4B..0A4D  ; PVALID      # GURMUKHI VOW SIGN OO..GURMUKHI S
0A4E..0A50  ; UNASSIGNED  # <reserved>..<reserved>
0A51        ; PVALID      # GURMUKHI SIGN UDAAT
0A52..0A58  ; UNASSIGNED  # <reserved>..<reserved>
0A59..0A5C  ; PVALID      # GURMUKHI LET KHHA..GURMUKHI LET RRA
0A5D        ; UNASSIGNED  # <reserved>
0A5E        ; PVALID      # GURMUKHI LET FA
0A5F..0A65  ; UNASSIGNED  # <reserved>..<reserved>
0A66..0A75  ; PVALID      # GURMUKHI DIG ZERO..GURMUKHI SIGN YA
0A76..0A80  ; UNASSIGNED  # <reserved>..<reserved>
0A81..0A83  ; PVALID      # GUJARATI SIGN CANDRABINDU..GUJARATI
0A84        ; UNASSIGNED  # <reserved>
0A85..0A8D  ; PVALID      # GUJARATI LET A..GUJARATI VOW CAND
0A8E        ; UNASSIGNED  # <reserved>
0A8F..0A91  ; PVALID      # GUJARATI LET E..GUJARATI VOW CAND
0A92        ; UNASSIGNED  # <reserved>
0A93..0AA8  ; PVALID      # GUJARATI LET O..GUJARATI LET NA
0AA9        ; UNASSIGNED  # <reserved>
0AAA..0AB0  ; PVALID      # GUJARATI LET PA..GUJARATI LET RA
0AB1        ; UNASSIGNED  # <reserved>
0AB2..0AB3  ; PVALID      # GUJARATI LET LA..GUJARATI LET LLA
0AB4        ; UNASSIGNED  # <reserved>
0AB5..0AB9  ; PVALID      # GUJARATI LET VA..GUJARATI LET HA
0ABA..0ABB  ; UNASSIGNED  # <reserved>..<reserved>
0ABC..0AC5  ; PVALID      # GUJARATI SIGN NUKTA..GUJARATI VOW
0AC6        ; UNASSIGNED  # <reserved>
0AC7..0AC9  ; PVALID      # GUJARATI VOW SIGN E..GUJARATI VOW
0ACA        ; UNASSIGNED  # <reserved>
0ACB..0ACD  ; PVALID      # GUJARATI VOW SIGN O..GUJARATI SIG
0ACE..0ACF  ; UNASSIGNED  # <reserved>..<reserved>
0AD0        ; PVALID      # GUJARATI OM
0AD1..0ADF  ; UNASSIGNED  # <reserved>..<reserved>
0AE0..0AE3  ; PVALID      # GUJARATI LET VOC RR..GUJARATI V
0AE4..0AE5  ; UNASSIGNED  # <reserved>..<reserved>
0AE6..0AEF  ; PVALID      # GUJARATI DIG ZERO..GUJARATI DIG NINE
0AF0..0AF1  ; FREE_PVAL   # GUJARATI ABBR SIGN..GUJARATI RUPEE S
0AF2..0B00  ; UNASSIGNED  # <reserved>..<reserved>
0B01..0B03  ; PVALID      # ORIYA SIGN CANDRABINDU..ORIYA SIGN V
0B04        ; UNASSIGNED  # <reserved>
0B05..0B0C  ; PVALID      # ORIYA LET A..ORIYA LET VOC L
0B0D..0B0E  ; UNASSIGNED  # <reserved>..<reserved>
```

```
0B0F..0B10  ; PVALID      # ORIYA LET E..ORIYA LET AI
0B11..0B12  ; UNASSIGNED  # <reserved>..<reserved>
0B13..0B28  ; PVALID      # ORIYA LET O..ORIYA LET NA
0B29        ; UNASSIGNED  # <reserved>
0B2A..0B30  ; PVALID      # ORIYA LET PA..ORIYA LET RA
0B31        ; UNASSIGNED  # <reserved>
0B32..0B33  ; PVALID      # ORIYA LET LA..ORIYA LET LLA
0B34        ; UNASSIGNED  # <reserved>
0B35..0B39  ; PVALID      # ORIYA LET VA..ORIYA LET HA
0B3A..0B3B  ; UNASSIGNED  # <reserved>..<reserved>
0B3C..0B44  ; PVALID      # ORIYA SIGN NUKTA..ORIYA VOW SIGN
0B45..0B46  ; UNASSIGNED  # <reserved>..<reserved>
0B47..0B48  ; PVALID      # ORIYA VOW SIGN E..ORIYA VOW SIG
0B49..0B4A  ; UNASSIGNED  # <reserved>..<reserved>
0B4B..0B4D  ; PVALID      # ORIYA VOW SIGN O..ORIYA SIGN VIRA
0B4E..0B55  ; UNASSIGNED  # <reserved>..<reserved>
0B56..0B57  ; PVALID      # ORIYA AI LEN MARK..ORIYA AU LENG
0B58..0B5B  ; UNASSIGNED  # <reserved>..<reserved>
0B5C..0B5D  ; PVALID      # ORIYA LET RRA..ORIYA LET RHA
0B5E        ; UNASSIGNED  # <reserved>
0B5F..0B63  ; PVALID      # ORIYA LET YYA..ORIYA VOW SIGN VOCA
0B64..0B65  ; UNASSIGNED  # <reserved>..<reserved>
0B66..0B6F  ; PVALID      # ORIYA DIG ZERO..ORIYA DIG NINE
0B70        ; FREE_PVAL   # ORIYA ISSHAR
0B71        ; PVALID      # ORIYA LET WA
0B72..0B77  ; FREE_PVAL   # ORIYA FRACT ONE QUART..ORIYA FRACT
0B78..0B81  ; UNASSIGNED  # <reserved>..<reserved>
0B82..0B83  ; PVALID      # TAMIL SIGN ANUSVARA..TAMIL SIGN VIS
0B84        ; UNASSIGNED  # <reserved>
0B85..0B8A  ; PVALID      # TAMIL LET A..TAMIL LET UU
0B8B..0B8D  ; UNASSIGNED  # <reserved>..<reserved>
0B8E..0B90  ; PVALID      # TAMIL LET E..TAMIL LET AI
0B91        ; UNASSIGNED  # <reserved>
0B92..0B95  ; PVALID      # TAMIL LET O..TAMIL LET KA
0B96..0B98  ; UNASSIGNED  # <reserved>..<reserved>
0B99..0B9A  ; PVALID      # TAMIL LET NGA..TAMIL LET CA
0B9B        ; UNASSIGNED  # <reserved>
0B9C        ; PVALID      # TAMIL LET JA
0B9D        ; UNASSIGNED  # <reserved>
0B9E..0B9F  ; PVALID      # TAMIL LET NYA..TAMIL LET TTA
0BA0..0BA2  ; UNASSIGNED  # <reserved>..<reserved>
0BA3..0BA4  ; PVALID      # TAMIL LET NNA..TAMIL LET TA
0BA5..0BA7  ; UNASSIGNED  # <reserved>..<reserved>
0BA8..0BAA  ; PVALID      # TAMIL LET NA..TAMIL LET PA
0BAB..0BAD  ; UNASSIGNED  # <reserved>..<reserved>
0BAE..0BB9  ; PVALID      # TAMIL LET MA..TAMIL LET HA
0BBA..0BBD  ; UNASSIGNED  # <reserved>..<reserved>
0BBE..0BC2  ; PVALID      # TAMIL VOW SIGN AA..TAMIL VOW SI
```

```
0BC3..0BC5  ; UNASSIGNED  # <reserved>..<reserved>
0BC6..0BC8  ; PVALID      # TAMIL VOW SIGN E..TAMIL VOW SIG
0BC9        ; UNASSIGNED  # <reserved>
0BCA..0BCD  ; PVALID      # TAMIL VOW SIGN O..TAMIL SIGN VIRA
0BCE..0BCF  ; UNASSIGNED  # <reserved>..<reserved>
0BD0        ; PVALID      # TAMIL OM
0BD1..0BD6  ; UNASSIGNED  # <reserved>..<reserved>
0BD7        ; PVALID      # TAMIL AU LEN MARK
0BD8..0BE5  ; UNASSIGNED  # <reserved>..<reserved>
0BE6..0BEF  ; PVALID      # TAMIL DIG ZERO..TAMIL DIG NINE
0BF0..0BFA  ; FREE_PVAL   # TAMIL NUM TEN..TAMIL NUM SIGN
0BFB..0C00  ; UNASSIGNED  # <reserved>..<reserved>
0C01..0C03  ; PVALID      # TELUGU SIGN CANDRABINDU..TELUGU SIG
0C04        ; UNASSIGNED  # <reserved>
0C05..0C0C  ; PVALID      # TELUGU LET A..TELUGU LET VOC L
0C0D        ; UNASSIGNED  # <reserved>
0C0E..0C10  ; PVALID      # TELUGU LET E..TELUGU LET AI
0C11        ; UNASSIGNED  # <reserved>
0C12..0C28  ; PVALID      # TELUGU LET O..TELUGU LET NA
0C29        ; UNASSIGNED  # <reserved>
0C2A..0C33  ; PVALID      # TELUGU LET PA..TELUGU LET LLA
0C34        ; UNASSIGNED  # <reserved>
0C35..0C39  ; PVALID      # TELUGU LET VA..TELUGU LET HA
0C3A..0C3C  ; UNASSIGNED  # <reserved>..<reserved>
0C3D..0C44  ; PVALID      # TELUGU SIGN AVAGRAHA..TELUGU VOW SI
0C45        ; UNASSIGNED  # <reserved>
0C46..0C48  ; PVALID      # TELUGU VOW SIGN E..TELUGU VOW SIGN
0C49        ; UNASSIGNED  # <reserved>
0C4A..0C4D  ; PVALID      # TELUGU VOW SIGN O..TELUGU SIGN VIRA
0C4E..0C54  ; UNASSIGNED  # <reserved>..<reserved>
0C55..0C56  ; PVALID      # TELUGU LEN MARK..TELUGU AI LEN MARK
0C57        ; UNASSIGNED  # <reserved>
0C58..0C59  ; PVALID      # TELUGU LET TSA..TELUGU LET DZA
0C5A..0C5F  ; UNASSIGNED  # <reserved>..<reserved>
0C60..0C63  ; PVALID      # TELUGU LET VOC RR..TELUGU VOW S
0C64..0C65  ; UNASSIGNED  # <reserved>..<reserved>
0C66..0C6F  ; PVALID      # TELUGU DIG ZERO..TELUGU DIG NINE
0C70..0C77  ; UNASSIGNED  # <reserved>..<reserved>
0C78..0C7F  ; FREE_PVAL   # TELUGU FRACTION DIG ZERO..TELUGU S
0C80..0C81  ; UNASSIGNED  # <reserved>..<reserved>
0C82..0C83  ; PVALID      # KANNADA SIGN ANUSVARA..KANNADA SIGN
0C84        ; UNASSIGNED  # <reserved>
0C85..0C8C  ; PVALID      # KANNADA LET A..KANNADA LET VOC L
0C8D        ; UNASSIGNED  # <reserved>
0C8E..0C90  ; PVALID      # KANNADA LET E..KANNADA LET AI
0C91        ; UNASSIGNED  # <reserved>
0C92..0CA8  ; PVALID      # KANNADA LET O..KANNADA LET NA
0CA9        ; UNASSIGNED  # <reserved>
```

```
0CAA..0CB3  ; PVALID      # KANNADA LET PA..KANNADA LET LLA
0CB4        ; UNASSIGNED  # <reserved>
0CB5..0CB9  ; PVALID      # KANNADA LET VA..KANNADA LET HA
0CBA..0CBB  ; UNASSIGNED  # <reserved>..<reserved>
0CBC..0CC4  ; PVALID      # KANNADA SIGN NUKTA..KANNADA VOW SIG
0CC5        ; UNASSIGNED  # <reserved>
0CC6..0CC8  ; PVALID      # KANNADA VOW SIGN E..KANNADA VOW SIG
0CC9        ; UNASSIGNED  # <reserved>
0CCA..0CCD  ; PVALID      # KANNADA VOW SIGN O..KANNADA SIGN VI
0CCE..0CD4  ; UNASSIGNED  # <reserved>..<reserved>
0CD5..0CD6  ; PVALID      # KANNADA LEN MARK..KANNADA AI LEN MA
0CD7..0CDD  ; UNASSIGNED  # <reserved>..<reserved>
0CDE        ; PVALID      # KANNADA LET FA
0CDF        ; UNASSIGNED  # <reserved>
0CE0..0CE3  ; PVALID      # KANNADA LET VOC RR..KANNADA VOW SIG
0CE4..0CE5  ; UNASSIGNED  # <reserved>..<reserved>
0CE6..0CEF  ; PVALID      # KANNADA DIG ZERO..KANNADA DIG NINE
0CF0        ; UNASSIGNED  # <reserved>
0CF1..0CF2  ; PVALID      # KANNADA SIGN JIHVAMULIYA..KANNADA S
0CF3..0D01  ; UNASSIGNED  # <reserved>..<reserved>
0D02..0D03  ; PVALID      # MALAY SIGN ANUSVARA..MALAY SIGN VIS
0D04        ; UNASSIGNED  # <reserved>
0D05..0D0C  ; PVALID      # MALAY LET A..MALAY LET VOC
0D0D        ; UNASSIGNED  # <reserved>
0D0E..0D10  ; PVALID      # MALAY LET E..MALAY LET AI
0D11        ; UNASSIGNED  # <reserved>
0D12..0D3A  ; PVALID      # MALAY LET O..MALAY LET TTTA
0D3B..0D3C  ; UNASSIGNED  # <reserved>..<reserved>
0D3D..0D44  ; PVALID      # MALAY SIGN AVAGRAHA..MALAY VOW SIG
0D45        ; UNASSIGNED  # <reserved>
0D46..0D48  ; PVALID      # MALAY VOW SIGN E..MALAY VOW SIGN
0D49        ; UNASSIGNED  # <reserved>
0D4A..0D4E  ; PVALID      # MALAY VOW SIGN O..MALAY LET DOT REP
0D4F..0D56  ; UNASSIGNED  # <reserved>..<reserved>
0D57        ; PVALID      # MALAY AU LEN MARK
0D58..0D5F  ; UNASSIGNED  # <reserved>..<reserved>
0D60..0D63  ; PVALID      # MALAY LET VOC RR..MALAY VOW
0D64..0D65  ; UNASSIGNED  # <reserved>..<reserved>
0D66..0D6F  ; PVALID      # MALAY DIG ZERO..MALAY DIG NINE
0D70..0D75  ; FREE_PVAL   # MALAY NUM TEN..MALAY FRACTION THR
0D76..0D78  ; UNASSIGNED  # <reserved>..<reserved>
0D79        ; FREE_PVAL   # MALAY DATE MARK
0D7A..0D7F  ; PVALID      # MALAY LET CHILLU NN..MALAY LET
0D80..0D81  ; UNASSIGNED  # <reserved>..<reserved>
0D82..0D83  ; PVALID      # SINH SIGN ANUSVARAYA..SINH SIGN VIS
0D84        ; UNASSIGNED  # <reserved>
0D85..0D96  ; PVALID      # SINH LET AYANNA..SINH LET AUYANN
0D97..0D99  ; UNASSIGNED  # <reserved>..<reserved>
```

```
0D9A..0DB1  ; PVALID      # SINH LET ALPAPRAANA KAYANNA..SINH L
0DB2        ; UNASSIGNED  # <reserved>
0DB3..0DBB  ; PVALID      # SINH LET SANYAKA DAYANNA..SINH LETT
0DBC        ; UNASSIGNED  # <reserved>
0DBD        ; PVALID      # SINH LET DANTAJA LAYANNA
0DBE..0DBF  ; UNASSIGNED  # <reserved>..<reserved>
0DC0..0DC6  ; PVALID      # SINH LET VAYANNA..SINH LET FAYAN
0DC7..0DC9  ; UNASSIGNED  # <reserved>..<reserved>
0DCA        ; PVALID      # SINH SIGN AL-LAKUNA
0DCB..0DCE  ; UNASSIGNED  # <reserved>..<reserved>
0DCF..0DD4  ; PVALID      # SINH VOW SIGN AELA-PILLA..SINH VOW
0DD5        ; UNASSIGNED  # <reserved>
0DD6        ; PVALID      # SINH VOW SIGN DIGA PAA-PILLA
0DD7        ; UNASSIGNED  # <reserved>
0DD8..0DDF  ; PVALID      # SINH VOW SIGN GAETTA-PILLA..SINH VO
0DE0..0DF1  ; UNASSIGNED  # <reserved>..<reserved>
0DF2..0DF3  ; PVALID      # SINH VOW SIGN DIGA GAETTA-PILLA..SI
0DF4        ; FREE_PVAL   # SINH PUNCT KUNDDALIYA
0DF5..0E00  ; UNASSIGNED  # <reserved>..<reserved>
0E01..0E32  ; PVALID      # THAI CHAR KO KAI..THAI CHAR SARA A
0E33        ; FREE_PVAL   # THAI CHAR SARA AM
0E34..0E3A  ; PVALID      # THAI CHAR SARA I..THAI CHAR PHINTH
0E3B..0E3E  ; UNASSIGNED  # <reserved>..<reserved>
0E3F        ; FREE_PVAL   # THAI CURRENCY SYM BAHT
0E40..0E4E  ; PVALID      # THAI CHAR SARA E..THAI CHAR YAMAKK
0E4F        ; FREE_PVAL   # THAI CHAR FONGMAN
0E50..0E59  ; PVALID      # THAI DIG ZERO..THAI DIG NINE
0E5A..0E5B  ; FREE_PVAL   # THAI CHAR ANGKHANKHU..THAI CHAR KH
0E5C..0E80  ; UNASSIGNED  # <reserved>..<reserved>
0E81..0E82  ; PVALID      # LAO LET KO..LAO LET KHO SUNG
0E83        ; UNASSIGNED  # <reserved>
0E84        ; PVALID      # LAO LET KHO TAM
0E85..0E86  ; UNASSIGNED  # <reserved>..<reserved>
0E87..0E88  ; PVALID      # LAO LET NGO..LAO LET CO
0E89        ; UNASSIGNED  # <reserved>
0E8A        ; PVALID      # LAO LET SO TAM
0E8B..0E8C  ; UNASSIGNED  # <reserved>..<reserved>
0E8D        ; PVALID      # LAO LET NYO
0E8E..0E93  ; UNASSIGNED  # <reserved>..<reserved>
0E94..0E97  ; PVALID      # LAO LET DO..LAO LET THO TAM
0E98        ; UNASSIGNED  # <reserved>
0E99..0E9F  ; PVALID      # LAO LET NO..LAO LET FO SUNG
0EA0        ; UNASSIGNED  # <reserved>
0EA1..0EA3  ; PVALID      # LAO LET MO..LAO LET LO LING
0EA4        ; UNASSIGNED  # <reserved>
0EA5        ; PVALID      # LAO LET LO LOOT
0EA6        ; UNASSIGNED  # <reserved>
0EA7        ; PVALID      # LAO LET WO
```

```
0EA8..0EA9  ; UNASSIGNED  # <reserved>..<reserved>
0EAA..0EAB  ; PVALID      # LAO LET SO SUNG..LAO LET HO SUNG
0EAC        ; UNASSIGNED  # <reserved>
0EAD..0EB2  ; PVALID      # LAO LET O..LAO VOW SIGN AA
0EB3        ; FREE_PVAL   # LAO VOW SIGN AM
0EB4..0EB9  ; PVALID      # LAO VOW SIGN I..LAO VOW SIGN UU
0EBA        ; UNASSIGNED  # <reserved>
0EBB..0EBD  ; PVALID      # LAO VOW SIGN MAI KON..LAO SEMIVOW SIG
0EBE..0EBF  ; UNASSIGNED  # <reserved>..<reserved>
0EC0..0EC4  ; PVALID      # LAO VOW SIGN E..LAO VOW SIGN AI
0EC5        ; UNASSIGNED  # <reserved>
0EC6        ; PVALID      # LAO KO LA
0EC7        ; UNASSIGNED  # <reserved>
0EC8..0ECD  ; PVALID      # LAO TONE MAI EK..LAO NIGGAHITA
0ECE..0ECF  ; UNASSIGNED  # <reserved>..<reserved>
0ED0..0ED9  ; PVALID      # LAO DIG ZERO..LAO DIG NINE
0EDA..0EDB  ; UNASSIGNED  # <reserved>..<reserved>
0EDC..0EDD  ; FREE_PVAL   # LAO HO NO..LAO HO MO
0EDE..0EDF  ; PVALID      # LAO LET KHMU GO..TIB SYL OM
0EE0..0EEF  ; UNASSIGNED  # <reserved>..<reserved>
0F00        ; PVALID      # TIB SYLL OM
0F01..0F0A  ; FREE_PVAL   # TIB MARK GTER YIG MGO TRUNC A..TIB
0F0B        ; PVALID      # TIB MARK INTERSYLLABIC TSHEG
0F0C..0F17  ; FREE_PVAL   # TIB MARK DELIMITER TSHEG BSTAR..TIB
0F18..0F19  ; PVALID      # TIB ASTROLOGICAL SIGN -KHYUD PA..TIB
0F1A..0F1F  ; FREE_PVAL   # TIB SIGN RDEL DKAR GCIG..TIB SIGN RD
0F20..0F29  ; PVALID      # TIB DIG ZERO..TIB DIG NINE
0F2A..0F34  ; FREE_PVAL   # TIB DIG HALF ONE..TIB MARK BSDUS R
0F35        ; PVALID      # TIB MARK NGAS BZUNG NYI ZLA
0F36        ; FREE_PVAL   # TIB MARK CARET DZUD RTAGS BZHI MIG C
0F37        ; PVALID      # TIB MARK NGAS BZUNG SGOR RTAGS
0F38        ; FREE_PVAL   # TIB MARK CHE MGO
0F39        ; PVALID      # TIB MARK TSA PHRU
0F3A..0F3D  ; FREE_PVAL   # TIB MARK GUG RTAGS GYON..TIB MARK AN
0F3E..0F47  ; PVALID      # TIB SIGN YAR TSHES..TIB LET JA
0F48        ; UNASSIGNED  # <reserved>
0F49..0F6C  ; PVALID      # TIB LET NYA..TIB LET RRA
0F6D..0F70  ; UNASSIGNED  # <reserved>..<reserved>
0F71..0F76  ; PVALID      # TIB VOW SIGN AA..TIB VOW SIGN VO
0F77        ; FREE_PVAL   # TIB VOW SIGN VO RR
0F78        ; PVALID      # TIB VOW SIGN VO L
0F79        ; FREE_PVAL   # TIB VOW SIGN VO LL
0F7A..0F84  ; PVALID      # TIB VOW SIGN E..TIB MARK H
0F85        ; FREE_PVAL   # TIB MARK PALUTA
0F86..0F8F  ; PVALID      # TIB SIGN LCI RTAGS..TIB SUBJOIN S
0F90..0F97  ; PVALID      # TIB SUBJOIN LET KA..TIB SUBJOIN
0F98        ; UNASSIGNED  # <reserved>
0F99..0FBC  ; PVALID      # TIB SUBJOIN LET NYA..TIB SUBJOI
```

```
0FBD        ; UNASSIGNED  # <reserved>
0FBE..0FC5  ; FREE_PVAL   # TIB KU RU KHA..TIB SYM RDO RJE
0FC6        ; PVALID      # TIB SYM PADMA GDAN
0FC7..0FCC  ; FREE_PVAL   # TIB SYM RDO RJE RGYA GRAM..TIB SY
0FCD        ; UNASSIGNED  # <reserved>
0FCE..0FDA  ; FREE_PVAL   # TIB SIGN RDEL NAG RDEL DKAR..TIB MA
0FDB..0FFF  ; UNASSIGNED  # <reserved>..<reserved>
1000..1049  ; PVALID      # MYAN LET KA..MYAN DIG NINE
104A..104F  ; FREE_PVAL   # MYAN SIGN LITTLE SECTION..MYAN SYM
1050..109D  ; PVALID      # MYAN LET SHA..MYAN VOW SIGN AITON
109E..109F  ; FREE_PVAL   # MYAN SYM SHAN ONE..MYAN SYM SHAN EX
10A0..10C5  ; PVALID      # GEORG CAP LET AN..GEORG CAP LET HOE
10C6        ; UNASSIGNED  # <reserved>
10C7        ; PVALID      # GEORG CAP LET YN
10C8..10CC  ; UNASSIGNED  # <reserved>..<reserved>
10CD        ; PVALID      # GEORG CAP LET AEN
10CE..10CF  ; UNASSIGNED  # <reserved>..<reserved>
10D0..10FA  ; PVALID      # GEORG LET AN..GEORG LET AIN
10FB..10FC  ; FREE_PVAL   # GEORG PARA SEP..MOD LET GEORG NAR
10FD..10FF  ; PVALID      # GEORG LET AEN..GEORG LET LABIAL
1100..11FF  ; DISALLOWED  # HANGUL CHO KIYEOK..HANGUL JONG SSA
1200..1248  ; PVALID      # ETHI SYL HA..ETHI SYL QWA
1249        ; UNASSIGNED  # <reserved>
124A..124D  ; PVALID      # ETHI SYL QWI..ETHI SYL QWE
124E..124F  ; UNASSIGNED  # <reserved>..<reserved>
1250..1256  ; PVALID      # ETHI SYL QHA..ETHI SYL QHO
1257        ; UNASSIGNED  # <reserved>
1258        ; PVALID      # ETHI SYL QHWA
1259        ; UNASSIGNED  # <reserved>
125A..125D  ; PVALID      # ETHI SYL QHWI..ETHI SYL QH
125E..125F  ; UNASSIGNED  # <reserved>..<reserved>
1260..1288  ; PVALID      # ETHI SYL BA..ETHI SYL XWA
1289        ; UNASSIGNED  # <reserved>
128A..128D  ; PVALID      # ETHI SYL XWI..ETHI SYL XWE
128E..128F  ; UNASSIGNED  # <reserved>..<reserved>
1290..12B0  ; PVALID      # ETHI SYL NA..ETHI SYL KWA
12B1        ; UNASSIGNED  # <reserved>
12B2..12B5  ; PVALID      # ETHI SYL KWI..ETHI SYL KWE
12B6..12B7  ; UNASSIGNED  # <reserved>..<reserved>
12B8..12BE  ; PVALID      # ETHI SYL KXA..ETHI SYL KXO
12BF        ; UNASSIGNED  # <reserved>
12C0        ; PVALID      # ETHI SYL KXWA
12C1        ; UNASSIGNED  # <reserved>
12C2..12C5  ; PVALID      # ETHI SYL KXWI..ETHI SYL KX
12C6..12C7  ; UNASSIGNED  # <reserved>..<reserved>
12C8..12D6  ; PVALID      # ETHI SYL WA..ETHI SYL PHAR
12D7        ; UNASSIGNED  # <reserved>
12D8..1310  ; PVALID      # ETHI SYL ZA..ETHI SYL GWA
```

```
1311        ; UNASSIGNED  # <reserved>
1312..1315  ; PVALID      # ETHI SYL GWI..ETHI SYL GWE
1316..1317  ; UNASSIGNED  # <reserved>..<reserved>
1318..135A  ; PVALID      # ETHI SYL GGA..ETHI SYL FYA
135B..135C  ; UNASSIGNED  # <reserved>..<reserved>
135D..135F  ; PVALID      # ETHI COMB GEM AND VOW..ETHI COMB GE
1360..137C  ; FREE_PVAL   # ETHI SECT MARK..ETHI NUM TEN THOUS
137D..137F  ; UNASSIGNED  # <reserved>..<reserved>
1380..138F  ; PVALID      # ETHI SYL SEBATBEIT MWA..ETHI SYL PW
1390..1399  ; FREE_PVAL   # ETHI TON MARK YIZET..ETHI TON MARK
139A..139F  ; UNASSIGNED  # <reserved>..<reserved>
13A0..13F4  ; PVALID      # CHEROKEE LET A..CHEROKEE LET YV
13F5..13FF  ; UNASSIGNED  # <reserved>..<reserved>
1400        ; FREE_PVAL   # CANAD SYL HYPHEN
1401..166C  ; PVALID      # CANAD SYL E..CANAD SYL CAR
166D..166E  ; FREE_PVAL   # CANAD SYL CHI SIGN..CANAD SYLLAB
166F..167F  ; PVALID      # CANAD SYL QAI..CANAD SYL B
1680        ; FREE_PVAL   # OGHAM SPACE MARK
1681..169A  ; PVALID      # OGHAM LET BEITH..OGHAM LET PEITH
169B..169C  ; FREE_PVAL   # OGHAM FEATHER MARK..OGHAM REV FEAT
169D..169F  ; UNASSIGNED  # <reserved>..<reserved>
16A0..16EA  ; PVALID      # RUNIC LET FEHU FEOH FE F..RUNIC LET
16EB..16F0  ; FREE_PVAL   # RUNIC SINGLE PUNCT..RUNIC BELGTHOR
16F1..16FF  ; UNASSIGNED  # <reserved>..<reserved>
1700..170C  ; PVALID      # TAGALOG LET A..TAGALOG LET YA
170D        ; UNASSIGNED  # <reserved>
170E..1714  ; PVALID      # TAGALOG LET LA..TAGALOG SIGN VIRAMA
1715..171F  ; UNASSIGNED  # <reserved>..<reserved>
1720..1734  ; PVALID      # HANUNOO LET A..HANUNOO SIGN PAMUDPO
1735..1736  ; FREE_PVAL   # PHILIP SINGLE PUNCT..PHILIP DOUBLE
1737..173F  ; UNASSIGNED  # <reserved>..<reserved>
1740..1753  ; PVALID      # BUHID LET A..BUHID VOW SIGN U
1754..175F  ; UNASSIGNED  # <reserved>..<reserved>
1760..176C  ; PVALID      # TAGBANWA LET A..TAGBANWA LET YA
176D        ; UNASSIGNED  # <reserved>
176E..1770  ; PVALID      # TAGBANWA LET LA..TAGBANWA LET SA
1771        ; UNASSIGNED  # <reserved>
1772..1773  ; PVALID      # TAGBANWA VOW SIGN I..TAGBANWA VOW S
1774..177F  ; UNASSIGNED  # <reserved>..<reserved>
1780..17B3  ; PVALID      # KHMER LET KA..KHMER IND VOW QAU
17B4..17B5  ; DISALLOWED  # KHMER VOW INH AQ..KHMER VOW INH AA
17B6..17D3  ; PVALID      # KHMER VOW SIGN AA..KHMER SIGN BATHA
17D4..17D6  ; FREE_PVAL   # KHMER SIGN KHAN..KHMER SIGN CAMNUC
17D7        ; PVALID      # KHMER SIGN LEK TOO
17D8..17DB  ; FREE_PVAL   # KHMER SIGN BEYYAL..KHMER CURR SYM R
17DC..17DD  ; PVALID      # KHMER SIGN AVAKRAHASANYA..KHMER SIG
17DE..17DF  ; UNASSIGNED  # <reserved>..<reserved>
17E0..17E9  ; PVALID      # KHMER DIG ZERO..KHMER DIG NINE
```

```
17EA..17EF  ; UNASSIGNED  # <reserved>..<reserved>
17F0..17F9  ; FREE_PVAL   # KHMER SYM LEK ATTAK SON..KHMER SYM
17FA..17FF  ; UNASSIGNED  # <reserved>..<reserved>
1800..180A  ; FREE_PVAL   # MONG BIRGA..MONG NIRUGU
180B..180E  ; DISALLOWED  # MONG FREE VAR SEL ONE..MONG VOW SEP
180F        ; UNASSIGNED  # <reserved>
1810..1819  ; PVALID      # MONG DIG ZERO..MONG DIG NINE
181A..181F  ; UNASSIGNED  # <reserved>..<reserved>
1820..1877  ; PVALID      # MONG LET A..MONG LET MANCHU
1878..187F  ; UNASSIGNED  # <reserved>..<reserved>
1880..18AA  ; PVALID      # MONG LET ALI GALI ANUSVARA ONE..MON
18AB..18AF  ; UNASSIGNED  # <reserved>..<reserved>
18B0..18F5  ; PVALID      # CAN SYL OY..CAN SYL CA
18F6..18FF  ; UNASSIGNED  # <reserved>..<reserved>
1900..191C  ; PVALID      # LIMBU VOW-CARRIER LET..LIMBU LET HA
191D..191F  ; UNASSIGNED  # <reserved>..<reserved>
1920..192B  ; PVALID      # LIMBU VOW SIGN A..LIMBU SUBJOIN LET
192C..192F  ; UNASSIGNED  # <reserved>..<reserved>
1930..193B  ; PVALID      # LIMBU SM LET KA..LIMBU SIGN SA-I
193C..193F  ; UNASSIGNED  # <reserved>..<reserved>
1940        ; FREE_PVAL   # LIMBU SIGN LOO
1941..1943  ; UNASSIGNED  # <reserved>..<reserved>
1944..1945  ; FREE_PVAL   # LIMBU EXCLAM MARK..LIMBU QUEST MARK
1946..196D  ; PVALID      # LIMBU DIG ZERO..TAI LE LET AI
196E..196F  ; UNASSIGNED  # <reserved>..<reserved>
1970..1974  ; PVALID      # TAI LE LET TONE-2..TAI LE LET TONE-
1975..197F  ; UNASSIGNED  # <reserved>..<reserved>
1980..19AB  ; PVALID      # NEW TAI LUE LET HIGH QA..NEW TAI LU
19AC..19AF  ; UNASSIGNED  # <reserved>..<reserved>
19B0..19C9  ; PVALID      # NEW TAI LUE VOW SIGN VOW SHORT..NEW
19CA..19CF  ; UNASSIGNED  # <reserved>..<reserved>
19D0..19D9  ; PVALID      # NEW TAI LUE DIG ZERO..NEW TAI DIG N
19DA        ; FREE_PVAL   # NEW TAI LUE THAM
19DB..19DD  ; UNASSIGNED  # <reserved>..<reserved>
19DE..19FF  ; FREE_PVAL   # NEW TAI LUE SIGN LAE..KHMER SYM DAP
1A00..1A1B  ; PVALID      # BUGIN LET KA..BUGIN VOW SIGN AE
1A1C..1A1D  ; UNASSIGNED  # <reserved>..<reserved>
1A1E..1A1F  ; FREE_PVAL   # BUGIN PALLAWA..BUGIN END OF SECTION
1A20..1A5E  ; PVALID      # TAI THAM LET HIGH KA..TAI THAM CONS
1A5F        ; UNASSIGNED  # <reserved>
1A60..1A7C  ; PVALID      # TAI THAM SIGN SAKOT..TAI THAM SIGN
1A7D..1A7E  ; UNASSIGNED  # <reserved>..<reserved>
1A7F..1A89  ; PVALID      # TAI THAM COMB CRYPT DOT..TAI THAM D
1A8A..1A8F  ; UNASSIGNED  # <reserved>..<reserved>
1A90..1A99  ; PVALID      # TAI THAM THAM DIG ZERO..TAI THAM TH
1A9A..1A9F  ; UNASSIGNED  # <reserved>..<reserved>
1AA0..1AA6  ; FREE_PVAL   # TAI THAM SIGN WIANG..TAI THAM SIGN
1AA7        ; PVALID      # TAI THAM SIGN MAI YAMOK
```

```
1AA8..1AAD  ; FREE_PVAL   # TAI THAM SIGN KAAN..TAI THAM SIGN C
1AAE..1AFF  ; UNASSIGNED  # <reserved>..<reserved>
1B00..1B4B  ; PVALID      # BAL SIGN ULU RICEM..BAL LET ASYURA
1B4C..1B4F  ; UNASSIGNED  # <reserved>..<reserved>
1B50..1B59  ; PVALID      # BAL DIG ZERO..BAL DIG NINE
1B5A..1B6A  ; FREE_PVAL   # BAL PANTI..BAL MUS SYM DANG
1B6B..1B73  ; PVALID      # BAL MUS SYM COMB TEGEH..BAL MUS
1B74..1B7C  ; FREE_PVAL   # BAL MUS SYM RIGHT-HAND OPEN DUG
1B7D..1B7F  ; UNASSIGNED  # <reserved>..<reserved>
1B80..1BF3  ; PVALID      # SUND SIGN PANYECEK..BATAK PANONGONAN
1BF4..1BFB  ; UNASSIGNED  # <reserved>..<reserved>
1BFC..1BFF  ; FREE_PVAL   # BATAK SYM BINDU NA METEK..BATAK SYM
1C00..1C37  ; PVALID      # LEPCHA LET KA..LEPCHA SIGN NUKTA
1C38..1C3A  ; UNASSIGNED  # <reserved>..<reserved>
1C3B..1C3F  ; FREE_PVAL   # LEPCHA PUNCT TA-ROL..LEPCHA PUNCT T
1C40..1C49  ; PVALID      # LEPCHA DIG ZERO..LEPCHA DIG NINE
1C4A..1C4C  ; UNASSIGNED  # <reserved>..<reserved>
1C4D..1C7D  ; PVALID      # LEPCHA LET TTA..OL CHIKI AHAD
1C7E..1C7F  ; FREE_PVAL   # OL CHIKI PUNCT MUCAAD..OL CHIKI PUN
1C80..1CBF  ; UNASSIGNED  # <reserved>..<reserved>
1CC0..1CC7  ; FREE_PVAL   # SUNDA PUNCT BINDU SURYA..SUNDA PUNC
1CC8..1CCF  ; UNASSIGNED  # <reserved>..<reserved>
1CD0..1CD2  ; PVALID      # VED TONE KARSHANA..VED TONE PRENKHA
1CD3        ; FREE_PVAL   # VED SIGN NIHSHVASA
1CD4..1CF6  ; PVALID      # VED SIGN YAJURVEDIC MID SVARITA..VE
1CF7..1CFF  ; UNASSIGNED  # <reserved>..<reserved>
1D00..1D2B  ; PVALID      # LAT LET SM CAP A..CYR LET SM
1D2C..1D2E  ; FREE_PVAL   # MOD LET CAP A..MOD LET C
1D2F        ; PVALID      # MOD LET CAP BARRED B
1D30..1D3A  ; FREE_PVAL   # MOD LET CAP D..MOD LET C
1D3B        ; PVALID      # MOD LET CAP REV N
1D3C..1D4D  ; FREE_PVAL   # MOD LET CAP O..MOD LET S
1D4E        ; PVALID      # MOD LET SM TURNED I
1D4F..1D6A  ; FREE_PVAL   # MOD LET SM K..GREEK SUB SMA
1D6B..1D77  ; PVALID      # LAT SM LET UE..LAT SM LET TU
1D78        ; FREE_PVAL   # MOD LET CYR EN
1D79..1D9A  ; PVALID      # LAT SM LET INSULAR G..LAT SM LE
1D9B..1DBF  ; FREE_PVAL   # MOD LET SM TURNED ALPHA..MOD
1DC0..1DE6  ; PVALID      # COMB DOTTED GRAVE ACCENT..COMB LAT
1DE7..1DFB  ; UNASSIGNED  # <reserved>..<reserved>
1DFC..1E99  ; PVALID      # COMB DOUBLE INV BREVE BEL..LAT SM L
1E9A..1E9B  ; FREE_PVAL   # LAT SM LET A W R HALF RING..LAT SM
1E9C..1F15  ; PVALID      # LAT SM LET LONG S W DIAG STROKE..GR
1F16..1F17  ; UNASSIGNED  # <reserved>..<reserved>
1F18..1F1D  ; PVALID      # GREEK CAP LET EPSILON W PSILI..GRE
1F1E..1F1F  ; UNASSIGNED  # <reserved>..<reserved>
1F20..1F45  ; PVALID      # GREEK SM LET ETA W PSILI..GREEK SMA
1F46..1F47  ; UNASSIGNED  # <reserved>..<reserved>
```

```
1F48..1F4D  ; PVALID      # GREEK CAP LET OMICRON W PSILI..GRE
1F4E..1F4F  ; UNASSIGNED  # <reserved>..<reserved>
1F50..1F57  ; PVALID      # GREEK SM LET UPSILON W PSILI..GREEK
1F58        ; UNASSIGNED  # <reserved>
1F59        ; PVALID      # GREEK CAP LET UPSILON W DASIA
1F5A        ; UNASSIGNED  # <reserved>
1F5B        ; PVALID      # GREEK CAP LET UPSILON W DASIA AND
1F5C        ; UNASSIGNED  # <reserved>
1F5D        ; PVALID      # GREEK CAP LET UPSILON W DASIA AND
1F5E        ; UNASSIGNED  # <reserved>
1F5F..1F7D  ; PVALID      # GREEK CAP LET UPSILON W DASIA A..GR
1F7E..1F7F  ; UNASSIGNED  # <reserved>..<reserved>
1F80..1F87  ; PVALID      # GREEK SM LET ALPHA W PSILI AND YPOG
1F88..1F8F  ; FREE_PVAL   # GREEK CAP LET ALPHA W PSILI AND..GR
1F90..1F97  ; PVALID      # GREEK SM LET ETA W PSILI AND YP..GR
1F98..1F9F  ; FREE_PVAL   # GREEK CAP LET ETA W PSILI AND P..GR
1FA0..1FA7  ; PVALID      # GREEK SM LET OMEGA W PSILI AND ..GR
1FA8..1FAF  ; FREE_PVAL   # GREEK CAPL LET OMEGA W PSILI AN..GR
1FB0..1FB4  ; PVALID      # GREEK SM LET ALPHA W VRACHY..GREEK
1FB5        ; UNASSIGNED  # <reserved>
1FB6..1FBB  ; PVALID      # GREEK SM LET ALPHA W PERISPOMEN..GR
1FBC..1FBD  ; FREE_PVAL   # GREEK CAP LET ALPHA W PROSGEGRA..GR
1FBE        ; PVALID      # GREEK PROSGEGRAMMENI
1FBF..1FC1  ; FREE_PVAL   # GREEK PSILI..GREEK DIALYTIKA AND PE
1FC2..1FC4  ; PVALID      # GREEK SM LET ETA W VARIA AND YP..GR
1FC5        ; UNASSIGNED  # <reserved>
1FC6..1FCB  ; PVALID      # GREEK SM LET ETA W PERISPOMENI..GR
1FCC..1FCF  ; FREE_PVAL   # GREEK CAP LET ETA W PROSGEGRAM..GR
1FD0..1FD3  ; PVALID      # GREEK SM LET IOTA W VRACHY..GREEK S
1FD4..1FD5  ; UNASSIGNED  # <reserved>..<reserved>
1FD6..1FDB  ; PVALID      # GREEK SM LET IOTA W PERISPOMENI..GR
1FDC        ; UNASSIGNED  # <reserved>
1FDD..1FDF  ; FREE_PVAL   # GREEK DASIA AND VARIA..GREEK DASIA
1FE0..1FEC  ; PVALID      # GREEK SM LET UPSILON W VRACHY..GREE
1FED..1FEF  ; FREE_PVAL   # GREEK DIALYTIKA AND VARIA..GREEK VA
1FF0..1FF1  ; UNASSIGNED  # <reserved>..<reserved>
1FF2..1FF4  ; PVALID      # GREEK SM LET OMEGA W VARIA AND YPOG
1FF5        ; UNASSIGNED  # <reserved>
1FF6..1FFB  ; PVALID      # GREEK SM LET OMEGA W PERISPOMEN..GR
1FFC..1FFE  ; FREE_PVAL   # GREEK CAP LET OMEGA W PROSGEGRA..GR
1FFF        ; UNASSIGNED  # <reserved>
2000..200A  ; FREE_PVAL   # EN QUAD..HAIR SPACE
200B        ; DISALLOWED  # ZERO WIDTH SPACE
200C..200D  ; CONTEXTJ    # ZERO WIDTH NON-JOINER..ZERO WIDTH J
200E..200F  ; DISALLOWED  # LEFT-TO-RIGHT MARK..RIGHT-TO-LEFT M
2010..2027  ; FREE_PVAL   # HYPHEN..HYPHENATION POINT
2028..202E  ; DISALLOWED  # LINE SEP..RIGHT-TO-LEFT OVERRIDE
202F..205F  ; FREE_PVAL   # NARROW NO-BREAK SPACE..MED MATH SP
```

```
2060..2064  ; DISALLOWED  # WORD JOINER..INVISIBLE PLUS
2065        ; UNASSIGNED  # <reserved>
2066..206F  ; DISALLOWED  # LEFT-TO-RIGHT IS..NOM DIGIT SHAPES
2070..2071  ; FREE_PVAL   # SUPER ZERO..SUPER LAT SM LET I
2072..2073  ; UNASSIGNED  # <reserved>..<reserved>
2074..208E  ; FREE_PVAL   # SUPER FOUR..SUB RIGHT PARENTHESIS
208F        ; UNASSIGNED  # <reserved>
2090..209C  ; FREE_PVAL   # LAT SUB SM LET A..LAT SUB SM LET T
209D..209F  ; UNASSIGNED  # <reserved>..<reserved>
20A0..20BA  ; FREE_PVAL   # EURO-CURRENCY SIGN..TURKISH LIRA SI
20BB..20CF  ; UNASSIGNED  # <reserved>..<reserved>
20D0..20DC  ; PVALID      # COMB LEFT HARPOON ABOVE..COMB FOUR
20DD..20E0  ; FREE_PVAL   # COMB ENC CIRC..COMB ENC CIRC BACKS
20E1        ; PVALID      # COMB L R ARROW ABOVE
20E2..20E4  ; FREE_PVAL   # COMB ENC SCREEN..COMB ENC UPWARD PO
20E5..20F0  ; PVALID      # COMB REV SOLIDUS OVERLAY..COMB ASTE
20F1..20FF  ; UNASSIGNED  # <reserved>..<reserved>
2100..2129  ; FREE_PVAL   # ACCOUNT OF..TURNED GREEK SM LET IOT
212A..212B  ; PVALID      # KELVIN SIGN..ANGSTROM SIGN
212C..2125  ; FREE_PVAL   # SCRIPT CAP C..OUNCE SIGN
2126        ; PVALID      # OHM SIGN
2127..2131  ; FREE_PVAL   # INVERTED OHM SIGN..SCRIPT CAP F
2132        ; PVALID      # TURNED CAP F
2133..214D  ; FREE_PVAL   # SCRIPT CAP M..AKTIESELSKAB
214E        ; PVALID      # TURNED SM F
214F..2182  ; FREE_PVAL   # SYM FOR SAMAR SOURCE..ROM NUM TEN T
2183..2184  ; PVALID      # ROM NUM REV ONE HUNDRED..LAT SM LET
2185..2189  ; FREE_PVAL   # ROM NUM SIX LATE FORM..VULGAR FRACT
218A..218F  ; UNASSIGNED  # <reserved>..<reserved>
2190..23F3  ; FREE_PVAL   # LEFTWARDS ARROW..HOURGLASS W FLO
23F4..23FF  ; UNASSIGNED  # <reserved>..<reserved>
2400..2426  ; FREE_PVAL   # SYM FOR NULL..SYM FOR SUB FORM
2427..243F  ; UNASSIGNED  # <reserved>..<reserved>
2440..244A  ; FREE_PVAL   # OCR HOOK..OCR DOUBLE BACKSLASH
244B..245F  ; UNASSIGNED  # <reserved>..<reserved>
2460..26FF  ; FREE_PVAL   # CIRCLED DIG ONE..WHITE FLAG W HORIZ
2700        ; UNASSIGNED  # <reserved>
2701..2B4C  ; FREE_PVAL   # UP BLADE SCISSORS..RIGHTWARDS ARROW
2B4D..2B4F  ; UNASSIGNED  # <reserved>..<reserved>
2B50..2B59  ; FREE_PVAL   # WHITE MEDIUM STAR..HEAVY CIRCLED SA
2B5A..2BFF  ; UNASSIGNED  # <reserved>..<reserved>
2C00..2C2E  ; PVALID      # GLAG CAP LET AZU..GLAG CA
2C2F        ; UNASSIGNED  # <reserved>
2C30..2C5E  ; PVALID      # GLAG SM LET AZU..GLAG SMAL
2C5F        ; UNASSIGNED  # <reserved>
2C60..2C7B  ; PVALID      # LAT CAP LET L W DOUBLE BAR..LAT SM
2C7C..2C7D  ; FREE_PVAL   # LAT SUB SM LET J..MOD LET CAP V
2C7E..2CE4  ; PVALID      # LAT CAP LET S W SWASH TAIL..COPT SY
```

```
2CE5..2CEA  ; FREE_PVAL   # COPT SYM MI RO..COPT SYM SHIMA SIMA
2CEB..2CF3  ; PVALID      # COPT CAP LET CRYPTOGRAMMIC SHEI..CO
2CF4..2CF8  ; UNASSIGNED  # <reserved>..<reserved>
2CF9..2CFF  ; FREE_PVAL   # COPT OLD NUB FULL STOP..COPT MORPHO
2D00..2D25  ; PVALID      # GEORG SM LET AN..GEORG SM LET
2D26        ; UNASSIGNED  # <reserved>
2D27        ; PVALID      # GEORG SM LET YN
2D28..2D2C  ; UNASSIGNED  # <reserved>..<reserved>
2D2D        ; PVALID      # GEORG SM LET AEN
2D2E..2D2F  ; UNASSIGNED  # <reserved>..<reserved>
2D30..2D67  ; PVALID      # TIFINAGH LET YA..TIFINAGH LETTER YO
2D68..2D6E  ; UNASSIGNED  # <reserved>..<reserved>
2D6F..2D70  ; FREE_PVAL   # TIFINAGH MOD LET LABIALIZATION MARK
2D71..2D7E  ; UNASSIGNED  # <reserved>..<reserved>
2D7F..2D96  ; PVALID      # TIFINAGH CONS JOINER..ETHI SYL GGW
2D97..2D9F  ; UNASSIGNED  # <reserved>..<reserved>
2DA0..2DA6  ; PVALID      # ETHI SYL SSA..ETHI SYL SSO
2DA7        ; UNASSIGNED  # <reserved>
2DA8..2DAE  ; PVALID      # ETHI SYL CCA..ETHI SYL CCO
2DAF        ; UNASSIGNED  # <reserved>
2DB0..2DB6  ; PVALID      # ETHI SYL ZZA..ETHI SYL ZZO
2DB7        ; UNASSIGNED  # <reserved>
2DB8..2DBE  ; PVALID      # ETHI SYL CCHA..ETHI SYL CC
2DBF        ; UNASSIGNED  # <reserved>
2DC0..2DC6  ; PVALID      # ETHI SYL QYA..ETHI SYL QYO
2DC7        ; UNASSIGNED  # <reserved>
2DC8..2DCE  ; PVALID      # ETHI SYL KYA..ETHI SYL KYO
2DCF        ; UNASSIGNED  # <reserved>
2DD0..2DD6  ; PVALID      # ETHI SYL XYA..ETHI SYL XYO
2DD7        ; UNASSIGNED  # <reserved>
2DD8..2DDE  ; PVALID      # ETHI SYL GYA..ETHI SYL GYO
2DDF        ; UNASSIGNED  # <reserved>
2DE0..2DFF  ; PVALID      # COMB CYR LET BE..COMB CYRI
2E00..2E2E  ; FREE_PVAL   # RIGHT ANGLE SUB MARK..REV QUEST MAR
2E2F        ; PVALID      # VERT TILDE
2E30..2E3B  ; FREE_PVAL   # RING PNT..THREE-EM DASH
2E3C..2E7F  ; UNASSIGNED  # <reserved>..<reserved>
2E80..2E99  ; FREE_PVAL   # CJK RAD REPEAT..CJK RAD RAP
2E9A        ; UNASSIGNED  # <reserved>
2E9B..2EF3  ; FREE_PVAL   # CJK RAD CHOKE..CJK RAD C-SIMPLIFIED
2EF4..2EFF  ; UNASSIGNED  # <reserved>..<reserved>
2F00..2FD5  ; FREE_PVAL   # KANGXI RAD ONE..KANGXI RAD FLUTE
2FD6..2FEF  ; UNASSIGNED  # <reserved>..<reserved>
2FF0..2FFB  ; FREE_PVAL   # IDEO DESC CHAR LEFT TO RIGHT..IDEO
2FFC..2FFF  ; UNASSIGNED  # <reserved>..<reserved>
3000..3004  ; FREE_PVAL   # IDEO SPACE..JAPAN INDUST STAND
3005..3007  ; PVALID      # IDEO ITER MARK..IDEO NUMB ZERO
3008..3029  ; FREE_PVAL   # LEFT ANGLE BRACKET..HANGZH NUM NINE
```

```
302A..302D  ; PVALID      # IDEO LEVEL TONE MARK..IDEO ENT
302E..302F  ; DISALLOWED  # HANGUL SING DOT TONE MARK..WAVY DAS
3030        ; FREE_PVAL   # WAVY DASH
3031..3035  ; DISALLOWED  # VERT KANA REP MARK..VERT KANA REP M
3036..303A  ; FREE_PVAL   # CIRCLED POSTAL MARK..HANGZH NUM THI
303B        ; DISALLOWED  # VERT IDEO ITER MARK
303C        ; PVALID      # MASU MARK
303D..303F  ; FREE_PVAL   # PART ALTER MARK..IDEO HALF FILL
3040        ; UNASSIGNED  # <reserved>
3041..3096  ; PVALID      # HIRAGANA LET SM A..HIRAGANA LET SMA
3097..3098  ; UNASSIGNED  # <reserved>..<reserved>
3099..309A  ; PVALID      # COMB KAT-HIR VOICED SOUND
309B..309C  ; FREE_PVAL   # KAT-HIR VOICED SOUND MARK..KAT-HIR
309D..309E  ; PVALID      # HIRAGANA ITER MARK..HIRAGANA VOICED
309F..30A0  ; FREE_PVAL   # HIRAGANA DIGRAPH YORI..KAT-HIR DOU
30A1..30FA  ; PVALID      # KATAKANA LET SM A..KATAKANA LET VO
30FB        ; CONTEXTO    # KATAKANA MIDDLE DOT
30FC..30FE  ; PVALID      # KAT-HIR PROLONGED SOUND MARK..KATA
30FF        ; FREE_PVAL   # KATAKANA DIGRAPH KOTO
3100..3104  ; UNASSIGNED  # <reserved>..<reserved>
3105..312D  ; PVALID      # BOPOMOFO LET B..BOPOMOFO LET IH
312E..3130  ; UNASSIGNED  # <reserved>..<reserved>
3131..3163  ; FREE_PVAL   # HANGUL LET KIYEOK..HANGUL LET I
3164        ; DISALLOWED  # HANGUL FILLER
3165..318E  ; FREE_PVAL   # HANGUL LET SSANGNIEUN..HANGUL LET
318F        ; UNASSIGNED  # <reserved>
3190..319F  ; FREE_PVAL   # IDEO ANNO LINK MARK..IDEO ANNO MAN
31A0..31BA  ; PVALID      # BOPOMOFO LET BU..BOPOMOFO LET ZY
31BB..31BF  ; UNASSIGNED  # <reserved>..<reserved>
31C0..31E3  ; FREE_PVAL   # CJK STROKE T..CJK STROKE Q
31E4..31EF  ; UNASSIGNED  # <reserved>..<reserved>
31F0..31FF  ; PVALID      # KATAKANA LET SM KU..KATAKANA LET SM
3200..321E  ; FREE_PVAL   # PAREN HANGUL KIYEOK..PAREN KOREAN C
321F        ; UNASSIGNED  # <reserved>
3220..32FE  ; FREE_PVAL   # PAREN IDEO ONE..CIRCLED KATAKANA WO
32FF        ; UNASSIGNED  # <reserved>
3300..33FF  ; FREE_PVAL   # SQUARE APAATO..SQUARE GAL
3400..4DB5  ; PVALID      # <CJK Ideograph Extension A>
4DB6..4DBF  ; UNASSIGNED  # <reserved>..<reserved>
4DC0..4DFF  ; FREE_PVAL   # HEX FOR THE CREATIVE HEAVEN..HEX FO
4E00..9FCC  ; PVALID      # <CJK Ideograph>
9FCD..9FFF  ; UNASSIGNED  # <reserved>..<reserved>
A000..A48C  ; PVALID      # YI SYL IT..YI SYL YYR
A48D..A48F  ; UNASSIGNED  # <reserved>..<reserved>
A490..A4C6  ; FREE_PVAL   # YI RAD QOT..YI RAD KE
A4C7..A4CF  ; UNASSIGNED  # <reserved>..<reserved>
A4D0..A4FD  ; PVALID      # LISU LET BA..LISU LET TONE MYA JEU
A4FE..A4FF  ; FREE_PVAL   # LISU PUNCT COMMA..LISU PUNCT FUL
```

```
A500..A60C  ; PVALID      # VAI SYL EE..VAI SYL LENENER
A60D..A60F  ; FREE_PVAL   # VAI COMMA..VAI QUEST MARK
A610..A62B  ; PVALID      # VAI SYL NDOLE FA..VAI SYL NDOLE DO
A62C..A63F  ; UNASSIGNED  # <reserved>..<reserved>
A640..A66F  ; PVALID      # CYR CAP LET ZEMLYA..COMB CYR VZMET
A670..A673  ; FREE_PVAL   # COMB CYR TEN MILLIONS SIGN..SLAVON
A674..A67D  ; PVALID      # COMB CYR KAVYKA..COMB CYR PAYEROK
A67E        ; FREE_PVAL   # CYR KAVYKA
A67F..A697  ; PVALID      # CYR PAYEROK..CYR SM LET SHWE
A698..A69E  ; UNASSIGNED  # <reserved>..<reserved>
A69F..A6E5  ; PVALID      # COMB CYR LET IOTIFIED E..BAMUM LET
A6E6..A6EF  ; FREE_PVAL   # BAMUM LET MO..BAMUM LET KOGHOM
A6F0..A6F1  ; PVALID      # BAMUM COMB MARK KOQNDON..BAMUM COMB
A6F2..A6F7  ; FREE_PVAL   # BAMUM NJAEMLI..BAMUM QUEST MARK
A6F8..A6FF  ; UNASSIGNED  # <reserved>..<reserved>
A700..A716  ; FREE_PVAL   # MOD LET CHIN TONE YIN PING..MOD
A717..A71F  ; PVALID      # MOD LET DOT VERT BAR..MOD L
A720..A721  ; FREE_PVAL   # MOD LET STRESS AND HIGH TONE..MOD
A722..A76F  ; PVALID      # LAT CAP LET EGYPT ALEF..LAT SM LET
A770        ; FREE_PVAL   # MODIFIER LETTER US
A771..A788  ; PVALID      # LATIN SMALL LETTER DUM..MOD LET LOW
A789..A78A  ; FREE_PVAL   # MOD LET COLON..MOD LET SH EQUALS SI
A78B..A78E  ; PVALID      # LAT SM LET SALTILLO..LAT SM LET L W
A78F        ; UNASSIGNED  # <reserved>
A790..A793  ; PVALID      # LAT CAP LET N W DESC..LAT SM LET C
A794..A79F  ; UNASSIGNED  # <reserved>..<reserved>
A7A0..A7AA  ; PVALID      # LAT CAP LET G W OBLIQUE STROKE..LAT
A7AB..A7F7  ; UNASSIGNED  # <reserved>..<reserved>
A7F8..A7F9  ; FREE_PVAL   # MOD LET CAP H W STROKE..MOD LET SM
A7FA..A827  ; PVALID      # LAT LET SM CAP TURNED M..SYLOTI NA
A828..A82B  ; FREE_PVAL   # SYLOTI NAGRI POET MARK-1..SYLOTI NA
A82C..A82F  ; UNASSIGNED  # <reserved>..<reserved>
A830..A839  ; FREE_PVAL   # N INDIC FRACT ONE QUART..N INDIC QU
A83A..A83F  ; UNASSIGNED  # <reserved>..<reserved>
A840..A873  ; PVALID      # PHAGS-PA LET KA..PHAGS-PA LET CANDR
A874..A877  ; FREE_PVAL   # PHAGS-PA SINGLE HEAD MARK..PHAGS-PA
A878..A87F  ; UNASSIGNED  # <reserved>..<reserved>
A880..A8C4  ; PVALID      # SAUR SIGN ANUSVARA..SAUR SIGN VIRAM
A8C5..A8CD  ; UNASSIGNED  # <reserved>..<reserved>
A8CE..A8CF  ; FREE_PVAL   # SAUR DANDA..SAUR DOUBLE DANDA
A8D0..A8D9  ; PVALID      # SAUR DIG ZERO..SAUR DIG NINE
A8DA..A8DF  ; UNASSIGNED  # <reserved>..<reserved>
A8E0..A8F7  ; PVALID      # COMB DEVAN DIG ZERO..DEVAN SIGN CAN
A8F8..A8FA  ; FREE_PVAL   # DEVAN SIGN PUSHPIKA..DEVAN CARET
A8FB        ; PVALID      # DEVAN HEADSTROKE
A8FC..A8FF  ; UNASSIGNED  # <reserved>..<reserved>
A900..A92D  ; PVALID      # KAYAH LI DIG ZERO..KAYAH LI TONE CA
A92E..A92F  ; FREE_PVAL   # KAYAH LI SIGN CWI..KAYAH LI SIGN SH
```

```
A930..A953  ; PVALID      # REJANG LET KA..REJANG VIRAMA
A954..A95E  ; UNASSIGNED  # <reserved>..<reserved>
A95F        ; FREE_PVAL   # REJANG SECTION MARK
A960..A97C  ; DISALLOWED  # HANGUL CHO TIKEUT-MIUEM..HANGUL CHO
A97D..A97F  ; UNASSIGNED  # <reserved>..<reserved>
A980..A9C0  ; PVALID      # JAV SIGN PANYANGGA..JAV PANGKON
A9C1..A9CD  ; FREE_PVAL   # JAV LEFT RERENGGAN..JAV TURNED PADA
A9CE        ; UNASSIGNED  # <reserved>
A9CF..A9D9  ; PVALID      # JAV PANGRANGKEP..JAV DIG NINE
A9DA..A9DD  ; UNASSIGNED  # <reserved>..<reserved>
A9DE..A9DF  ; FREE_PVAL   # JAV PADA TIRTA TUMETES..JAV PADA I
A9E0..A9FF  ; UNASSIGNED  # <reserved>..<reserved>
AA00..AA36  ; PVALID      # CHAM LET A..CHAM CONS SIGN WA
AA37..AA3F  ; UNASSIGNED  # <reserved>..<reserved>
AA40..AA4D  ; PVALID      # CHAM LET FIN K..CHAM CONS SIGN FIN
AA4E..AA4F  ; UNASSIGNED  # <reserved>..<reserved>
AA50..AA59  ; PVALID      # CHAM DIG ZERO..CHAM DIG NINE
AA5A..AA5B  ; UNASSIGNED  # <reserved>..<reserved>
AA5C..AA5F  ; FREE_PVAL   # CHAM PUNCT SPIRAL..CHAM PUNCT TR
AA60..AA76  ; PVALID      # MYAN LET KHAMTI GA..MYAN LOGOGRAM K
AA77..AA79  ; FREE_PVAL   # MYAN SYM AITON EXCLAM..MYAN SYM AIT
AA7A..AA7B  ; PVALID      # MYAN LET AITON RA..MYAN SIGN PAO KA
AA7C..AA7F  ; UNASSIGNED  # <reserved>..<reserved>
AA80..AAC2  ; PVALID      # TAI VIET LET LOW KO..TAI VIET TONE
AAC3..AADA  ; UNASSIGNED  # <reserved>..<reserved>
AADB..AADD  ; PVALID      # TAI VIET SYM KON..TAI VIET SYM SAM
AADE..AADF  ; FREE_PVAL   # TAI VIET SYM HO HOI..TAI VIET SYM K
AAE0..AAEF  ; PVALID      # MEETEI MAYEK LET E..MEETEI MAYEK VO
AAF0..AAF1  ; FREE_PVAL   # MEETEI MAYEK CHEIKHAN..MEETEI MAYEK
AAF2..AAF6  ; PVALID      # MEETEI MAYEK ANJI..MEETEI MAYEK VIR
AAF7..AB00  ; UNASSIGNED  # <reserved>..<reserved>
AB01..AB06  ; PVALID      # ETHI SYL TTHU..ETHI SYL TTHO
AB07..AB08  ; UNASSIGNED  # <reserved>..<reserved>
AB09..AB0E  ; PVALID      # ETHI SYL DDHAA..ETHI SYL DDHO
AB0F..AB10  ; UNASSIGNED  # <reserved>..<reserved>
AB11..AB16  ; PVALID      # ETHI SYL DZU..ETHI SYL DZO
AB17..AB1F  ; UNASSIGNED  # <reserved>..<reserved>
AB20..AB26  ; PVALID      # ETHI SYL CCHHA..ETHI SYL CCHHO
AB27        ; UNASSIGNED  # <reserved>..<reserved>
AB28..AB2E  ; PVALID      # ETHI SYL BBAA..ETHI SYL BBO
AB2F..ABBF  ; UNASSIGNED  # <reserved>..<reserved>
ABC0..ABEA  ; PVALID      # MEETEI MAYEK LET KOK..MEETEI MAYEK
ABEB        ; FREE_PVAL   # MEETEI MAYEK CHEIKHEI
ABEC..ABED  ; PVALID      # MEETEI MAYEK LUM IYEK..MEETEI MAYEK
ABEE..ABEF  ; UNASSIGNED  # <reserved>..<reserved>
ABF0..ABF9  ; PVALID      # MEETEI MAYEK DIG ZERO..MEETEI MAYEK
ABFA..ABFF  ; UNASSIGNED  # <reserved>..<reserved>
AC00..D7A3  ; PVALID      # <Hangul Syllable>
```

```
D7A4..D7AF  ; UNASSIGNED  # <reserved>..<reserved>
D7B0..D7C6  ; DISALLOWED  # HANGUL JUNG O-YEO..HANGUL JUNG ARAE
D7C7..D7CA  ; UNASSIGNED  # <reserved>..<reserved>
D7CB..D7FB  ; DISALLOWED  # HANGUL JONG NIEUN-RIEUL..HANGUL JON
D7FC..D7FF  ; UNASSIGNED  # <reserved>..<reserved>
D800..F8FF  ; DISALLOWED  # <Non Private Use High Surrogate>
F900..FA6D  ; PVALID      # CJK COMP IDEO-F900..CJK COMP IDEO
FA6E..FA6F  ; UNASSIGNED  # <reserved>..<reserved>
FA70..FAD9  ; PVALID      # CJK COMP IDEO-FA70..CJK COMP IDEO
FADA..FAFF  ; UNASSIGNED  # <reserved>..<reserved>
FB00..FB06  ; FREE_PVAL   # LAT SM LIG FF..LAT SM LIG ST
FB07..FB12  ; UNASSIGNED  # <reserved>..<reserved>
FB13..FB17  ; FREE_PVAL   # ARMENIAN SM LIG MEN NOW..ARMENIAN SM
FB18..FB1C  ; UNASSIGNED  # <reserved>..<reserved>
FB1D..FB1F  ; PVALID      # HEBR LET YOD W HIRIQ..HEBR LIG YID Y
FB20..FB29  ; FREE_PVAL   # HEBR LET ALT AYIN..HEB LET ALT PLUS
FB2A..FB36  ; PVALID      # HEBR LET SHIN W SHIN DOT..HEBR LET Z
FB37        ; UNASSIGNED  # <reserved>
FB38..FB3C  ; PVALID      # HEBR LET TET W DAGESH..HEBR LET
FB3D        ; UNASSIGNED  # <reserved>
FB3E        ; PVALID      # HEBR LET MEM W DAGESH
FB3F        ; UNASSIGNED  # <reserved>
FB40..FB41  ; PVALID      # HEBR LET NUN W DAGESH..HEBR LET
FB42        ; UNASSIGNED  # <reserved>
FB43..FB44  ; PVALID      # HEBR LET FIN PE W DAGESH..HEBR L
FB45        ; UNASSIGNED  # <reserved>
FB46..FB4E  ; PVALID      # HEBR LET TSADI W DAGESH..HEBR LET P
FB4F..FBC1  ; FREE_PVAL   # HEBR LIG ALEF LAMED..ARAB SYM S
FBC2..FBD2  ; UNASSIGNED  # <reserved>..<reserved>
FBD3..FD3F  ; FREE_PVAL   # ARAB LET NG ISO FORM..ORNATE RIGHT
FD40..FD4F  ; UNASSIGNED  # <reserved>..<reserved>
FD50..FD8F  ; FREE_PVAL   # ARAB LIG TEH W JEEM W MEEM INIT
FD90..FD91  ; UNASSIGNED  # <reserved>..<reserved>
FD92..FDC7  ; FREE_PVAL   # ARAB LIG MEEM W JEEM W KHAH INI
FDC8..FDCF  ; UNASSIGNED  # <reserved>..<reserved>
FDD0..FDEF  ; DISALLOWED  # <noncharacter>..<noncharacter>
FDF0..FDFD  ; FREE_PVAL   # ARAB LIG SALLA USED..ARAB LIG BISMI
FDFE..FDFF  ; UNASSIGNED  # <reserved>..<reserved>
FE00..FE0F  ; DISALLOWED  # VAR SEL-1..VAR SEL-16
FE10..FE19  ; FREE_PVAL   # PRES FORM FOR VERT COMMA..PRES FORM
FE1A..FE1F  ; UNASSIGNED  # <reserved>..<reserved>
FE20..FE26  ; PVALID      # COMB LIG LEFT HALF..COMB CONJ MACRO
FE27..FE2F  ; UNASSIGNED  # <reserved>..<reserved>
FE30..FE52  ; FREE_PVAL   # PRES FORM FOR VERT TWO DOT LEAD..SM
FE53        ; UNASSIGNED  # <reserved>
FE54..FE66  ; FREE_PVAL   # SM SEMICOLON..SM EQUALS SIGN
FE67        ; UNASSIGNED  # <reserved>
FE68..FE6B  ; FREE_PVAL   # SM REV SOLIDUS..SM COMM AT
```

```
FE6C..FE6F  ; UNASSIGNED  # <reserved>..<reserved>
FE70..FE72  ; FREE_PVAL   # ARAB FATHATAN ISO FORM..ARAB DAMMAT
FE73        ; PVALID      # ARAB TAIL FRAGMENT
FE74        ; FREE_PVAL   # ARAB KASRATAN ISO FORM
FE75        ; UNASSIGNED  # <reserved>
FE76..FEFC  ; FREE_PVAL   # ARAB FATHA ISO FORM..ARAB LIG LAM W
FEFD..FEFE  ; UNASSIGNED  # <reserved>..<reserved>
FEFF        ; DISALLOWED  # ZERO WIDTH NO-BREAK SPACE
FF00        ; UNASSIGNED  # <reserved>
FF01..FF9F  ; FREE_PVAL   # FULLW EXCLAM MARK..HALFW KATA SE
FFA0        ; DISALLOWED  # HALFW HANGUL FILLER
FFA1..FFBE  ; FREE_PVAL   # HALFW HANGUL LET KIYEOK..HALFW H
FFBF..FFC1  ; UNASSIGNED  # <reserved>..<reserved>
FFC2..FFC7  ; FREE_PVAL   # HALFW HANGUL LET A..HALFW HANGUL
FFC8..FFC9  ; UNASSIGNED  # <reserved>..<reserved>
FFCA..FFCF  ; FREE_PVAL   # HALFW HANGUL LET YEO..HALFW HANGU
FFD0..FFD1  ; UNASSIGNED  # <reserved>..<reserved>
FFD2..FFD7  ; FREE_PVAL   # HALFW HANGUL LET YO..HALFW HANGUL
FFD8..FFD9  ; UNASSIGNED  # <reserved>..<reserved>
FFDA..FFDC  ; FREE_PVAL   # HALFW HANGUL LET EU..HALFW HANGUL
FFDD..FFDF  ; UNASSIGNED  # <reserved>..<reserved>
FFE0..FFE6  ; FREE_PVAL   # FULLW CENT SIGN..FULLW WON SIGN
FFE7        ; UNASSIGNED  # <reserved>
FFE8..FFEE  ; FREE_PVAL   # HALFW FORMS LIGHT VERT..HALFW WH
FFEF..FFF8  ; UNASSIGNED  # <reserved>..<reserved>
FFF9..FFFB  ; DISALLOWED  # INTERL ANNO ANCHOR..INTERL ANNO TER
FFFC..FFFD  ; FREE_PVAL   # OBJECT REPL CHAR..REPL CHAR
FFFE..FFFF  ; DISALLOWED  # <noncharacter>..<noncharacter>
10000..1000B; PVALID      # LIN B SYL B008 A..LIN B SYL
1000C       ; UNASSIGNED  # <reserved>
1000D..10026; PVALID      # LIN B SYL B036 JO..LIN B SYL
10027       ; UNASSIGNED  # <reserved>
10028..1003A; PVALID      # LIN B SYL B060 RA..LIN B SYL
1003B       ; UNASSIGNED  # <reserved>
1003C..1003D; PVALID      # LIN B SYL B017 ZA..LIN B SYL
1003E       ; UNASSIGNED  # <reserved>
1003F..1004D; PVALID      # LIN B SYL B020 ZO..LIN B SYL
1004E..1004F; UNASSIGNED  # <reserved>..<reserved>
10050..1005D; PVALID      # LIN B SYM B018..LIN B SYM B089
1005E..1007F; UNASSIGNED  # <reserved>..<reserved>
10080..100FA; PVALID      # LIN B IDEO B100 MAN..LIN B IDEO
100FB..100FF; UNASSIGNED  # <reserved>..<reserved>
10100..10102; FREE_PVAL   # AEG WORD SEP LINE..AEG CHECK MAR
10103..10106; UNASSIGNED  # <reserved>..<reserved>
10107..10133; FREE_PVAL   # AEG NUM ONE..AEG NUM NINETY THOU
10134..10136; UNASSIGNED  # <reserved>..<reserved>
10137..1018A; FREE_PVAL   # AEG WEIGHT BASE UNIT..GREEK ZERO SI
1018B..1018F; UNASSIGNED  # <reserved>..<reserved>
```

```
10190..1019B; FREE_PVAL   # ROM SEXTANS SIGN..ROM CENTURIAL SIG
1019C..101CF; UNASSIGNED  # <reserved>..<reserved>
101D0..101FC; FREE_PVAL   # PHAISTOS DISC SIGN PED..PHAISTOS DI
101FD       ; PVALID      # PHAISTOS DISC SIGN COMB OBLIQUE STR
101FE..1027F; UNASSIGNED  # <reserved>..<reserved>
10280..1029C; PVALID      # LYCIAN LET A..LYCIAN LET X
1029D..1029F; UNASSIGNED  # <reserved>..<reserved>
102A0..102D0; PVALID      # CARIAN LET A..CARIAN LET UUU3
102D1..102FF; UNASSIGNED  # <reserved>..<reserved>
10300..1031E; PVALID      # OLD ITAL LET A..OLD ITAL LET UU
1031F       ; UNASSIGNED  # <reserved>
10320..10323; FREE_PVAL   # OLD ITAL NUM ONE..OLD ITAL NUM F
10324..1032F; UNASSIGNED  # <reserved>..<reserved>
10330..10340; PVALID      # GOTH LET AHSA..GOTH LET PAIRTHRA
10341       ; FREE_PVAL   # GOTH LET NINETY
10342..10349; PVALID      # GOTH LET RAIDA..GOTH LET OTHAL
1034A       ; FREE_PVAL   # GOTH LET NINE HUNDRED
1034B..1037F; UNASSIGNED  # <reserved>..<reserved>
10380..1039D; PVALID      # UGAR LET ALPA..UGAR LET SSU
1039E       ; UNASSIGNED  # <reserved>
1039F       ; FREE_PVAL   # UGAR WORD DIVIDER
103A0..103C3; PVALID      # OLD PERS SIGN A..OLD PERS SIGN HA
103C4..103C7; UNASSIGNED  # <reserved>..<reserved>
103C8..103CF; PVALID      # OLD PERS SIGN AURAMAZDAA..OLD PERS
103D0..103D5; FREE_PVAL   # OLD PERS WORD DIVIDER..OLD PERS NUM
103D6..103FF; UNASSIGNED  # <reserved>..<reserved>
10400..1049D; PVALID      # DESERET CAP LET LONG I..OSMANYA LET
1049E..1049F; UNASSIGNED  # <reserved>..<reserved>
104A0..104A9; PVALID      # OSMANYA DIG ZERO..OSMANYA DIG NINE
104AA..107FF; UNASSIGNED  # <reserved>..<reserved>
10800..10805; PVALID      # CYPRIOT SYL A..CYPRIOT SYL JA
10806..10807; UNASSIGNED  # <reserved>..<reserved>
10808       ; PVALID      # CYPRIOT SYL JO
10809       ; UNASSIGNED  # <reserved>
1080A..10835; PVALID      # CYPRIOT SYL KA..CYPRIOT SYL WO
10836       ; UNASSIGNED  # <reserved>
10837..10838; PVALID      # CYPRIOT SYL XA..CYPRIOT SYL XE
10839..1083B; UNASSIGNED  # <reserved>..<reserved>
1083C       ; PVALID      # CYPRIOT SYL ZA
1083D..1083E; UNASSIGNED  # <reserved>..<reserved>
1083F..10855; PVALID      # CYPRIOT SYL ZO..IMP ARAM LET TAW
10856       ; UNASSIGNED  # <reserved>
10857..1085F; FREE_PVAL   # IMP ARAM SECT SIGN..IMP ARAM
10860..108FF; UNASSIGNED  # <reserved>..<reserved>
10900..10915; PVALID      # PHOEN LET ALF..PHOEN LET TAU
10916..1091B; FREE_PVAL   # PHOEN NUM ONE..PHOEN NUM THR
1091C..1091E; UNASSIGNED  # <reserved>..<reserved>
1091F       ; FREE_PVAL   # PHOEN WORD SEP
```

```
10920..10939; PVALID      # LYDIAN LET A..LYDIAN LET C
1093A..1093E; UNASSIGNED  # <reserved>..<reserved>
1093F       ; FREE_PVAL   # LYDIAN TRIANGULAR MARK
10940..1097F; UNASSIGNED  # <reserved>..<reserved>
10980..109B7; PVALID      # MERO HIER LET A..MERO CURS LET
109B8..109BD; UNASSIGNED  # <reserved>..<reserved>
109BE..109BF; PVALID      # MERO CURS LOG RMT..MERO CURS L
109C0..109FF; UNASSIGNED  # <reserved>..<reserved>
10A00..10A03; PVALID      # KHARO LET A..KHARO VOW SIGN V
10A04       ; UNASSIGNED  # <reserved>
10A05..10A06; PVALID      # KHARO VOW SIGN E..KHARO VOW SI
10A07..10A0B; UNASSIGNED  # <reserved>..<reserved>
10A0C..10A13; PVALID      # KHARO VOW LEN MARK..KHARO LET
10A14       ; UNASSIGNED  # <reserved>
10A15..10A17; PVALID      # KHARO LET CA..KHARO LET JA
10A18       ; UNASSIGNED  # <reserved>
10A19..10A33; PVALID      # KHARO LET NYA..KHARO LET TTT
10A34..10A37; UNASSIGNED  # <reserved>..<reserved>
10A38..10A3A; PVALID      # KHARO SIGN BAR ABOVE..KHARO SIGN D
10A3B..10A3E; UNASSIGNED  # <reserved>..<reserved>
10A3F       ; PVALID      # KHARO VIRAMA
10A40..10A47; FREE_PVAL   # KHARO DIG ONE..KHARO NUM ONE
10A48..10A4F; UNASSIGNED  # <reserved>..<reserved>
10A50..10A58; FREE_PVAL   # KHARO PUNCT DOT..KHARO PUNCT
10A59..10A5F; UNASSIGNED  # <reserved>..<reserved>
10A60..10A7C; PVALID      # OLD S ARAB LET HE..OLD SOUTH ARAB
10A7D..10A7F; FREE_PVAL   # OLD S ARAB NUM ONE..OLD SOUTH ARAB
10A80..10AFF; UNASSIGNED  # <reserved>..<reserved>
10B00..10B35; PVALID      # AVESTAN LET A..AVESTAN LET HE
10B36..10B38; UNASSIGNED  # <reserved>..<reserved>
10B39..10B3F; FREE_PVAL   # AVESTAN ABBR MARK..LARGE ONE RING O
10B40..10B55; PVALID      # INSCRIPT PARTHIAN LET ALEPH..INSCRI
10B56..10B57; UNASSIGNED  # <reserved>..<reserved>
10B58..10B5F; FREE_PVAL   # INSCRIPT PARTHIAN NUM ONE..INSCRIPT
10B60..10B72; PVALID      # INSCRIPT PAHLAVI LET ALEPH..INSCRIP
10B73..10B77; UNASSIGNED  # <reserved>..<reserved>
10B78..10B7F; FREE_PVAL   # INSCRIPT PAHLAVI NUM ONE..INSCRIPT
10B80..10BFF; UNASSIGNED  # <reserved>..<reserved>
10C00..10C48; PVALID      # OLD TURK LET ORKHON A..OLD TURK LET
10C49..10E5F; UNASSIGNED  # <reserved>..<reserved>
10E60..10E7E; FREE_PVAL   # RUMI DIG ONE..RUMI FRACTION TWO THI
10E7F..10FFF; UNASSIGNED  # <reserved>..<reserved>
11000..11046; PVALID      # BRAHMI SIGN CANDRABINDU..BRAHMI VIR
11047..1104D; FREE_PVAL   # BRAHMI DANDA..BRAHMI PUNCT LOTUS
1104E..11051; UNASSIGNED  # <reserved>..<reserved>
11052..11065; FREE_PVAL   # BRAHMI NUM ONE..BRAHMI NUM ONE THOU
11066..1106F; PVALID      # BRAHMI DIG ZERO..BRAHMI DIG NINE
11070..1107F; UNASSIGNED  # <reserved>..<reserved>
```

```
11080..110BA; PVALID      # KAITHI SIGN CANDRABINDU..KAITHI SIG
110BB..110BC; FREE_PVAL   # KAITHI ABBR SIGN..KAITHI ENUM SIGN
110BD       ; DISALLOWED  # KAITHI NUM SIGN
110BE..110C1; FREE_PVAL   # KAITHI SECT MARK..KAITHI DOUBLE DAN
110C2..110CF; UNASSIGNED  # <reserved>..<reserved>
110D0..110F8; PVALID      # SORA SOMPENG LETTER SAH..SORA SOMPE
110F9..110EF; UNASSIGNED  # <reserved>..<reserved>
110F0..110F9; PVALID      # SORA SOMPENG DIG ZERO..SORA SOMPENG DI
110FA..110FF; UNASSIGNED  # <reserved>..<reserved>
11100..11134; PVALID      # CHAKMA SIGN CANDRABINDU..CHAKMA MAAYY
11135       ; UNASSIGNED  # <reserved>
11136..1113F; PVALID      # CHAKMA DIG ZERO..CHAKMA DIG NINE
11140..11143; FREE_PVAL   # CHAKMA SECT MARK..CHAKMA QUEST MARK
11144..1117F; UNASSIGNED  # <reserved>..<reserved>
11180..111C4; PVALID      # SHARADA SIGN CANDRABINDU..SHARADA OM
111C5..111C8; FREE_PVAL   # SHARADA DANDA..SHARADA SEPARATOR
111C9..111CF; UNASSIGNED  # <reserved>..<reserved>
111D0..111D9; PVALID      # SHARADA DIG ZERO..SHARADA DIG NINE
111DA..1167F; UNASSIGNED  # <reserved>..<reserved>
11680..116B7; PVALID      # TAKRI LET A..TAKRI SIGN NUKTA
116B8..116BF; UNASSIGNED  # <reserved>..<reserved>
116C0..116C9; PVALID      # TAKRI DIGIT ZERO..TAKRI DIG NINE
116CA..1FFFF; UNASSIGNED  # <reserved>..<reserved>
12000..1236E; PVALID      # CUNEI SIGN A..CUNEI SIGN ZUM
1236F..123FF; UNASSIGNED  # <reserved>..<reserved>
12400..12462; FREE_PVAL   # CUNEI NUM SIGN TWO ASH..CUNEI NUM
12463..1246F; UNASSIGNED  # <reserved>..<reserved>
12470..12473; FREE_PVAL   # CUNEI PUNCT SIGN OLD ASSYRIAN WORD
12474..12FFF; UNASSIGNED  # <reserved>..<reserved>
13000..1342E; PVALID      # EGYPT HIERO A001..EGYPT HIERO AA032
1342F..167FF; UNASSIGNED  # <reserved>..<reserved>
16800..16A38; PVALID      # BAMUM LET PHASE-A NGKUE MFON..BAMUN LE
16A39..16EFF; UNASSIGNED  # <reserved>..<reserved>
16F00..16F44; PVALID      # MIAO LET PA..MIAO LET HHA
16F45..16F4F; UNASSIGNED  # <reserved>..<reserved>
16F50..16F7E; PVALID      # MIAO LET NAS..MIAO VOWEL SIGN NG
16F7F..16F8E; UNASSIGNED  # <reserved>..<reserved>
16F8F..16F9F; PVALID      # MIAO TONE RIGHT..MIAO LET REF TON
16FA0..1AFFF; UNASSIGNED  # <reserved>..<reserved>
1B000..1B001; PVALID      # KATA LET ARCH E..KATA LET ARCH YE
1B002..1CFFF; UNASSIGNED  # <reserved>..<reserved>
1D000..1D0F5; FREE_PVAL   # BYZ MUS SYM PSILI..BYZ MUS
1D0F6..1D0FF; UNASSIGNED  # <reserved>..<reserved>
1D100..1D126; FREE_PVAL   # MUS SYM SINGLE BARLINE..MUS SYMBOL
1D127..1D128; UNASSIGNED  # <reserved>..<reserved>
1D129..1D164; FREE_PVAL   # MUS SYM MULT MEASURE REST..MUS SYM ONE
1D165..1D169; PVALID      # MUS SYM COMB STEM..MUS SYM COMB TREMOL
1D16A..1D16C; FREE_PVAL   # MUS SYM FING TREM-1..MUS SYM FING TREM
```

```
1D16D..1D172; PVALID      # MUS SYM COMB AUG DOT..MUS SYM COMB FL
1D173..1D17A; DISALLOWED  # MUS SYM BEGIN BEAM..MUS SYM END PHRASE
1D17B..1D182; PVALID      # MUS SYM COMB ACCENT..MUS SYM COMB LOUR
1D183..1D184; FREE_PVAL   # MUS SYM ARP UP..MUS SYM ARP DOWN
1D185..1D18B; PVALID      # MUS SYM COMB DOIT..MUS SYM COMB TRIPLE
1D18C..1D1A9; FREE_PVAL   # MUS SYM RINFORZANDO..MUS SYM DEG SLASH
1D1AA..1D1AD; PVALID      # MUS SYM COMB DOWN BOW..MUS SYM COMB SN
1D1AE..1D1DD; FREE_PVAL   # MUS SYM PEDAL MARK..MUS SYM PES SUBPUN
1D1DE..1D1FF; UNASSIGNED  # <reserved>..<reserved>
1D200..1D241; FREE_PVAL   # GREEK VOCAL NOTATION SYM-1..GREEK INS
1D242..1D244; FREE_PVAL   # COMB GREEK MUS TRISEME..COMB GREEK MU
1D245       ; FREE_PVAL   # GREEK MUSICAL LEIMMA
1D246..1D2FF; UNASSIGNED  # <reserved>..<reserved>
1D300..1D356; DISALLOWED  # MONOG FOR EARTH..TETRAG FOR FOSTERING
1D357..1D35F; UNASSIGNED  # <reserved>..<reserved>
1D360..1D371; DISALLOWED  # COUNT ROD UNIT DIG ONE..COUNT ROD TE
1D372..1D3FF; UNASSIGNED  # <reserved>..<reserved>
1D400..1D454; FREE_PVAL   # MATH BOLD CAP A..MATH IT
1D455       ; UNASSIGNED  # <reserved>
1D456..1D49C; FREE_PVAL   # MATH ITAL SM I..MATH SC
1D49D       ; UNASSIGNED  # <reserved>
1D49E..1D49F; FREE_PVAL   # MATH SCRIPT CAP C..MATH
1D4A0..1D4A1; UNASSIGNED  # <reserved>..<reserved>
1D4A2       ; FREE_PVAL   # MATH SCRIPT CAP G
1D4A3..1D4A4; UNASSIGNED  # <reserved>..<reserved>
1D4A5..1D4A6; FREE_PVAL   # MATH SCRIPT CAP J..MATH
1D4A7..1D4A8; UNASSIGNED  # <reserved>..<reserved>
1D4A9..1D4AC; FREE_PVAL   # MATH SCRIPT CAP N..MATH
1D4AD       ; UNASSIGNED  # <reserved>
1D4AE..1D4B9; FREE_PVAL   # MATH SCRIPT CAP S..MATH
1D4BA       ; UNASSIGNED  # <reserved>
1D4BB       ; FREE_PVAL   # MATH SCRIPT SM F
1D4BC       ; UNASSIGNED  # <reserved>
1D4BD..1D4C3; FREE_PVAL   # MATH SCRIPT SM H..MATH SC
1D4C4       ; UNASSIGNED  # <reserved>
1D4C5..1D505; FREE_PVAL   # MATH SCRIPT SM P..MATH FR
1D506       ; UNASSIGNED  # <reserved>
1D507..1D50A; FREE_PVAL   # MATH FRAKTUR CAP D..MATH
1D50B..1D50C; UNASSIGNED  # <reserved>..<reserved>
1D50D..1D514; FREE_PVAL   # MATH FRAKTUR CAP J..MATH
1D515       ; UNASSIGNED  # <reserved>
1D516..1D51C; FREE_PVAL   # MATH FRAKTUR CAP S..MATH
1D51D       ; UNASSIGNED  # <reserved>
1D51E..1D539; FREE_PVAL   # MATH FRAKTUR SM A..MATH D
1D53A       ; UNASSIGNED  # <reserved>
1D53B..1D53E; FREE_PVAL   # MATH DOUBLE-STRUCK CAP D..MATHEM
1D53F       ; UNASSIGNED  # <reserved>
1D540..1D544; FREE_PVAL   # MATH DOUBLE-STRUCK CAP I..MATHEM
```

```
1D545       ; UNASSIGNED  # <reserved>
1D546       ; FREE_PVAL   # MATH DOUBLE-STRUCK CAP O
1D547..1D549; UNASSIGNED  # <reserved>..<reserved>
1D54A..1D550; FREE_PVAL   # MATH DOUBLE-STRUCK CAP S..MATHEM
1D551       ; UNASSIGNED  # <reserved>
1D552..1D6A5; FREE_PVAL   # MATH DOUBLE-STRUCK SM A..MATHEMAT
1D6A6..1D6A7; UNASSIGNED  # <reserved>..<reserved>
1D6A8..1D7CB; FREE_PVAL   # MATH BOLD CAP ALPHA..MATHEMATICA
1D7CC..1D7CD; UNASSIGNED  # <reserved>..<reserved>
1D7CE..1D7FF; FREE_PVAL   # MATH BOLD DIG ZERO..MATH M
1D800..1EDFF; UNASSIGNED  # <reserved>..<reserved>
1EE00..1EE03; FREE_PVAL   # ARAB MATH ALEF..ARAB MATH DAL
1EE04       ; UNASSIGNED  # <reserved>
1EE05..1EE1F; FREE_PVAL   # ARAB MATH WAW..ARAB MATH DOTLESS QAF
1EE20       ; UNASSIGNED  # <reserved>
1EE21..1EE22; FREE_PVAL   # ARAB MATH INIT BEH..ARAB MATH INIT JEE
1EE23       ; UNASSIGNED  # <reserved>
1EE24       ; FREE_PVAL   # ARAB MATH INIT HEH
1EE25..1EE26; UNASSIGNED  # <reserved>..<reserved>
1EE27       ; FREE_PVAL   # ARAB MATH INIT HAH
1EE28       ; UNASSIGNED  # <reserved>
1EE29..1EE32; FREE_PVAL   # ARAB MATH INIT YEH..ARAB MATH INIT QAF
1EE33       ; UNASSIGNED  # <reserved>
1EE34..1EE37; FREE_PVAL   # ARAB MATH INIT SHEEN..ARAB MATH INITIA
1EE38       ; UNASSIGNED  # <reserved>
1EE39       ; FREE_PVAL   # ARAB MATH INIT SHEEN
1EE3A       ; UNASSIGNED  # <reserved>
1EE3B       ; FREE_PVAL   # ARAB MATH INIT GHAIN
1EE3C..1EE41; UNASSIGNED  # <reserved>..<reserved>
1EE42       ; FREE_PVAL   # ARAB MATH TAILED JEEM
1EE43..1EE46; UNASSIGNED  # <reserved>..<reserved>
1EE47       ; FREE_PVAL   # ARAB MATH TAILED HAH
1EE48       ; UNASSIGNED  # <reserved>
1EE49       ; FREE_PVAL   # ARAB MATH TAILED YEH
1EE4A       ; UNASSIGNED  # <reserved>
1EE4B       ; FREE_PVAL   # ARAB MATH TAILED LAM
1EE4C       ; UNASSIGNED  # <reserved>
1EE4D..1EE4F; FREE_PVAL   # ARAB MATH TAILED NOON..ARAB MATH TAILE
1EE50       ; UNASSIGNED  # <reserved>
1EE51..1EE52; FREE_PVAL   # ARAB MATH TAILED QAF..ARAB MATH TAILED
1EE53       ; UNASSIGNED  # <reserved>
1EE54       ; FREE_PVAL   # ARAB MATH TAILED SHEEN
1EE55..1EE56; UNASSIGNED  # <reserved>..<reserved>
1EE57       ; FREE_PVAL   # ARAB MATH TAILED KHAH
1EE58       ; UNASSIGNED  # <reserved>
1EE59       ; FREE_PVAL   # ARAB MATH TAILED DAD
1EE5A       ; UNASSIGNED  # <reserved>
1EE5B       ; FREE_PVAL   # ARAB MATH TAILED GHAIN
```

```
1EE5C       ; UNASSIGNED  # <reserved>
1EE5D       ; FREE_PVAL   # ARAB MATH TAILED DOTLESS NOON
1EE5E       ; UNASSIGNED  # <reserved>
1EE5F       ; FREE_PVAL   # ARAB MATH TAILED DOTLESS GHAIN
1EE60       ; UNASSIGNED  # <reserved>
1EE61..1EE62; FREE_PVAL   # ARAB MATH STRETCHED BEH..ARAB MATH STR
1EE63       ; UNASSIGNED  # <reserved>
1EE64       ; FREE_PVAL   # ARAB MATH STRETCHED HEH
1EE65..1EE66; UNASSIGNED  # <reserved>..<reserved>
1EE67..1EE6A; FREE_PVAL   # ARAB MATH STRETCHED HAH..ARAB MATH STR
1EE6B       ; UNASSIGNED  # <reserved>
1EE6C..1EE72; FREE_PVAL   # ARAB MATH STRETCHED MEEM..ARAB MATH ST
1EE73       ; UNASSIGNED  # <reserved>
1EE74..1EE77; FREE_PVAL   # ARAB MATH STRETCHED SHEEN..ARAB MATH S
1EE78       ; UNASSIGNED  # <reserved>
1EE79..1EE7C; FREE_PVAL   # ARAB MATH STRETCHED DAD..ARAB MATH STR
1EE7D       ; UNASSIGNED  # <reserved>
1EE7E       ; FREE_PVAL   # ARAB MATH STRETCHED DOTLESS FEH
1EE7F       ; UNASSIGNED  # <reserved>
1EE80..1EE89; FREE_PVAL   # ARAB MATH LOOPED ALEF..ARAB MATH LOOPE
1EE8A       ; UNASSIGNED  # <reserved>
1EE8B..1EE9B; FREE_PVAL   # ARAB MATH LOOPED LAM..ARAB MATH LOOPED
1EE9C..1EEA0; UNASSIGNED  # <reserved>..<reserved>
1EEA1..1EEA3; FREE_PVAL   # ARAB MATH DOUBLE-STRUCK BEH..ARAB MATH
1EEA4       ; UNASSIGNED  # <reserved>
1EEA5..1EEA9; FREE_PVAL   # ARAB MATH DOUBLE-STRUCK WAW..ARAB MATH
1EEAA       ; UNASSIGNED  # <reserved>
1EEAB..1EEBB; FREE_PVAL   # ARAB MATH DOUBLE-STRUCK LAM..ARAB MATH
1EEBC..1EEEF; UNASSIGNED  # <reserved>..<reserved>
1EEF0..1EEF1; FREE_PVAL   # ARAB MATH OP MEEM W HAH W TATWHEEL..AR
1EEF2..1EFFF; UNASSIGNED  # <reserved>..<reserved>
1F000..1F02B; FREE_PVAL   # MAHJONG TILE EAST WIND..MAHJONG TILE B
1F02C..1F02F; UNASSIGNED  # <reserved>..<reserved>
1F030..1F093; FREE_PVAL   # DOMINO TILE HORIZ BACK..DOMINO TILE VE
1F094..1F09F; UNASSIGNED  # <reserved>..<reserved>
1F0A0..1F0AE; FREE_PVAL   # PLAY CARD BACK..PLAY CARD KING OF SPAD
1F0AF..1F0B0; UNASSIGNED  # <reserved>..<reserved>
1F0B1..1F0BE; FREE_PVAL   # PLAY CARD ACE OF HEARTS..PLAY CARD KIN
1F0BF..1F0C0; UNASSIGNED  # <reserved>..<reserved>
1F0C1..1F0CF; FREE_PVAL   # PLAY CARD ACE OF DIAMONDS..PLAY CARD B
1F0D0       ; UNASSIGNED  # <reserved>
1F0D1..1F0DF; FREE_PVAL   # PLAY CARD ACE OF CLUBS..PLAY CARD WHIT
1F0E0..1F0FF; UNASSIGNED  # <reserved>..<reserved>
1F100..1F10A; FREE_PVAL   # DIG ZERO FULL STOP..DIG NINE COMMA
1F10B..1F10F; UNASSIGNED  # <reserved>..<reserved>
1F110..1F12E; FREE_PVAL   # PARENTHESIZED LAT CAP LET A..CIRCLE
1F12F       ; UNASSIGNED  # <reserved>
1F130..1F16B; FREE_PVAL   # SQUARED LAT CAP LET A..RAISED MD SIGN
```

```
1F16C..1F16F; UNASSIGNED  # <reserved>..<reserved>
1F170..1F19A; FREE_PVAL   # NEG SQ LAT CAP LET A..SQUARED VS
1F19B..1F1E5; UNASSIGNED  # <reserved>..<reserved>
1F1E6..1F202; FREE_PVAL   # REG IND SYMB LET A..SQ KATAKANA SA
1F203..1F20F; UNASSIGNED  # <reserved>..<reserved>
1F210..1F23A; FREE_PVAL   # SQ CJK UNIF IDEO-624B..SQ CJK UNIF IDE
1F23B..1F23F; UNASSIGNED  # <reserved>..<reserved>
1F240..1F248; FREE_PVAL   # TORT SH BRACK CJK UNIF IDEO-672C..TORT
1F249..1F24F; UNASSIGNED  # <reserved>..<reserved>
1F250..1F251; FREE_PVAL   # CIRC IDEO ADVANTAGE..CIRC IDEO ACCEPT
1F252..1F2FF; UNASSIGNED  # <reserved>..<reserved>
1F300..1F320; FREE_PVAL   # CYCLONE..SHOOTING STAR
1F321..1F32F; UNASSIGNED  # <reserved>..<reserved>
1F330..1F335; FREE_PVAL   # CHESTNUT..CACTUS
1F336       ; UNASSIGNED  # <reserved>
1F337..1F37C; FREE_PVAL   # TULIP..BABY BOTTLE
1F37D..1F37F; UNASSIGNED  # <reserved>..<reserved>
1F380..1F393; FREE_PVAL   # RIBBON..GRADUATION CAP
1F394..1F39F; UNASSIGNED  # <reserved>..<reserved>
1F3A0..1F3C4; FREE_PVAL   # CAROUSEL HORSE..SURFER
1F3C5       ; UNASSIGNED  # <reserved>
1F3C6..1F3CA; FREE_PVAL   # TROPHY..SWIMMER
1F3CB..1F3DF; UNASSIGNED  # <reserved>..<reserved>
1F3E0..1F3F0; FREE_PVAL   # HOUSE BUILDING..EUROPEAN CASTLE
1F3F1..1F3FF; UNASSIGNED  # <reserved>..<reserved>
1F400..1F43E; FREE_PVAL   # RAT..PAW PRINTS
1F43F       ; UNASSIGNED  # <reserved>
1F440       ; FREE_PVAL   # EYES
1F441       ; UNASSIGNED  # <reserved>
1F442..1F4F7; FREE_PVAL   # EAR..CAMERA
1F4F8       ; UNASSIGNED  # <reserved>
1F4F9..1F4FC; FREE_PVAL   # VIDEO CAMERA..VIDEOCASSETTE
1F4FD..1F4FF; UNASSIGNED  # <reserved>..<reserved>
1F500..1F53D; FREE_PVAL   # TWISTED RIGHTWARDS ARROWS..DOWN-POINTI
1F53E..1F53F; UNASSIGNED  # <reserved>..<reserved>
1F540..1F543; FREE_PVAL   # CIRCLED CROSS POMMEE..NOTCHED LEFT SEM
1F544..1F54F; UNASSIGNED  # <reserved>..<reserved>
1F550..1F567; FREE_PVAL   # CLOCK FACE ONE OCLOCK..CLOCK FACE TWEL
1F568..1F5FA; UNASSIGNED  # <reserved>..<reserved>
1F5FB..1F640; FREE_PVAL   # MOUNT FUJI..WEARY CAT FACE
1F641..1F644; UNASSIGNED  # <reserved>..<reserved>
1F645..1F650; FREE_PVAL   # FACE W NO GOOD GESTURE..PERSON W FO
1F650..1F67F; UNASSIGNED  # <reserved>..<reserved>
1F680..1F6C5; FREE_PVAL   # ROCKET..LEFT LUGGAGE
1F6C6..1F6FF; UNASSIGNED  # <reserved>..<reserved>
1F700..1F773; FREE_PVAL   # ALCHEMICAL SYMBOL FOR QUINTESSENCE..AL
1F774..1FFFF; UNASSIGNED  # <reserved>..<reserved>
20000..2A6D6; PVALID      # <CJK Ideograph Extension B>
```

```
2A6D7..2A6FF; UNASSIGNED  # <reserved>..<reserved>
2A700..2B734; PVALID      # <CJK Ideograph Extension C>
2A735..2A739; UNASSIGNED  # <reserved>..<reserved>
2A740..2B81D; PVALID      # <CJK Ideograph Extension D>
2B81E..2F7FF; UNASSIGNED  # <reserved>..<reserved>
2F800..2FA1D; PVALID      # CJK COMP IDEO-2F800..CJK COMPA
2FA1E..2FFFD; UNASSIGNED  # <reserved>..<reserved>
2FFFE..2FFFF; DISALLOWED  # <noncharacter>..<noncharacter>
30000..3FFFD; UNASSIGNED  # <reserved>..<reserved>
3FFFE..3FFFF; DISALLOWED  # <noncharacter>..<noncharacter>
40000..4FFFD; UNASSIGNED  # <reserved>..<reserved>
4FFFE..4FFFF; DISALLOWED  # <noncharacter>..<noncharacter>
50000..5FFFD; UNASSIGNED  # <reserved>..<reserved>
5FFFE..5FFFF; DISALLOWED  # <noncharacter>..<noncharacter>
60000..6FFFD; UNASSIGNED  # <reserved>..<reserved>
6FFFE..6FFFF; DISALLOWED  # <noncharacter>..<noncharacter>
70000..7FFFD; UNASSIGNED  # <reserved>..<reserved>
7FFFE..7FFFF; DISALLOWED  # <noncharacter>..<noncharacter>
80000..8FFFD; UNASSIGNED  # <reserved>..<reserved>
8FFFE..8FFFF; DISALLOWED  # <noncharacter>..<noncharacter>
90000..9FFFD; UNASSIGNED  # <reserved>..<reserved>
9FFFE..9FFFF; DISALLOWED  # <noncharacter>..<noncharacter>
A0000..AFFFD; UNASSIGNED  # <reserved>..<reserved>
AFFFE..AFFFF; DISALLOWED  # <noncharacter>..<noncharacter>
B0000..BFFFD; UNASSIGNED  # <reserved>..<reserved>
BFFFE..BFFFF; DISALLOWED  # <noncharacter>..<noncharacter>
C0000..CFFFD; UNASSIGNED  # <reserved>..<reserved>
CFFFE..CFFFF; DISALLOWED  # <noncharacter>..<noncharacter>
D0000..DFFFD; UNASSIGNED  # <reserved>..<reserved>
DFFFE..DFFFF; DISALLOWED  # <noncharacter>..<noncharacter>
E0000       ; UNASSIGNED  # <reserved>
E0001       ; DISALLOWED  # LANGUAGE TAG
E0002..E001F; UNASSIGNED  # <reserved>..<reserved>
E0020..E007F; DISALLOWED  # TAG SPACE..CANCEL TAG
E0080..E00FF; UNASSIGNED  # <reserved>..<reserved>
E0100..E01EF; DISALLOWED  # VAR SEL-17..VAR SEL-256
E01F0..EFFFD; UNASSIGNED  # <reserved>..<reserved>
EFFFE..10FFFF; DISALLOWED # <noncharacter>..<noncharacter>
```

3. IANA Considerations

See the IANA considerations section of the PRECIS framework specification [I-D.ietf-precis-framework].

4. Security Considerations

See the security considerations section of the PRECIS framework specification [I-D.ietf-precis-framework].

5. Normative References

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

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

Appendix A. Acknowledgements

Thanks to Joe Hildebrand and Takahiro Nemoto for the code they wrote to help check the table values.

Author’s Address

Peter Saint-Andre
&yet
P.O. Box 787
Parker, CO 80134
USA

Email: peter@andyet.net