Universal Acceptance

Universal Acceptance Compliance of Some Programming Language Libraries and Frameworks

14 September 2020

Report UASG018A

TABLE OF CONTENTS

Introduction

This document describes the results of the second phase of the work done before. The results of the previous phase are available on the UASG site. This new phase expands the previous work by adding new languages and frameworks and also verifying the compliance of internationalized email (EAI).

The first phase of the work verified the following languages and libraries for Universal Acceptance readiness:

Language	Framework/Library
Java	Commons Validator
Java	Guava
Java	ICU
Java	JRE
Python3	Django_auth
Python3	Encodings_Idna
Python3	Idna
Rust	Idna

These frameworks were essentially using domain names as their primary objects, so internationalized email was not tested.

This document describes the results of verifying the following languages and frameworks, including the testing of internationalized email:

Language	Framework/Library
C	libcurl
C	libidn2
C#	Mailkit
C#	Microsoft
Go	Idna
Go	Mail
Go	Smtp
Java	Commons-Validator
Java	Guava

Java	ICU
Java	Javamail/JakartaMail
Java	JRE
Javascript	Idna-uts46
Javascript	Nodemailer
Javascript	Validator
Python3	Django_auth
Python3	Email_Validator
Python3	Encodings_Idna
Python3	Idna
Python3	Smtplib
Rust	Idna
Rust	Lettre

The same libraries and frameworks from the first phase were verified again to see if there are improvements or regressions against their newest versions. Additional libraries and frameworks for the same language were added mostly because the new ones support email addresses.

In summary, most libraries and frameworks from previous phase were not improved for the purpose of universal acceptance. The following table list the new versions verified under this work.

Language	**Framework/Library**	**Previous Phase**	**This phase**
Java	Commons-validator	1.6	1.6
Java	Guava	26	28
Java	ICU	51.1	67.1
Java	JRE	10	11
Python3	Django_auth	2.7	3.0.7
Python3	Encodings_idna	3.7	3.8

Python3	Idna	2.7	2.9
Rust	Idna	0.1.4	0.2.0

New Libraries and Frameworks

This work verifies compliance of 4 new languages and 14 new libraries and frameworks, as listed below.

Language	Library/Framework	Version
C	libcurl	7.70.0
C	libidn2	2.3.0
C#	mailkit	2.7.0
C#	microsoft (IdnMapping)	.net core 3.1.301
Go	idna	1.14.4
Go	mail	1.14.4
Go	smtp	1.14.4
Java	Jakarta Mail	1.6.5
Js	Idna-uts46	1.1.0
Js	nodemailer	10.7.1
Js	validator	13.1.1
Python3	email_validator	1.1.1
Python3	smtplib	3.8
Rust	lettre	0.9.3

Mobile platforms languages and libraries will be tested in a follow-up work.

Methodology

In order to verify Universal Acceptance readiness, 5 datasets of sample internationalized domain names & email addresses were used. The next section gives a short description of each one. These datasets are described in detail in UASG004 and UASG018

For email address internationalization (EAI), a dummy SMTP server, based on the popular Mailhog SMTP server, was used to verify the support of the SMTPUTF8 SMTP option by the mailer libraries and frameworks. However, Mailhog does not support SMTPUTF8 so we used a fork that enhance it to support SMTPUTF8. This dummy server, running within a docker, simulates communication with a real SMTP server from the library/framework perspective and checks if it behaves as expected.

Datasets

H_DNS

Performs a syntactic check on a domain name. Determines whether the name appears to be correctly formed. If any part of the name already appears to be in ASCII form (an A-label), verify it can be converted to Unicode. Ref. RFC5891, RFC1035, SAC053

H_ES (to check EAI)

Performs a syntactic check on an email address. Determines whether the address appears to be correctly formed. Ref. RFC5891, RFC6531

H_ID

Compares the identifier stored in the system against the one used to authenticate by the user. The test cases aim to validate proper handling of internationalized identifiers by applications. Ref. RFC8264

L_A2U

Converts a domain name in ASCII to Unicode using the process described in RFC5891. If the domain name, or any constituent label, is already in Unicode or an ASCII label does not begin with the ACE prefix, the original label should not be altered. Ref. RFC5891

L_U2A

Converts a domain name in Unicode to ASCII using the process described in RFC5891 for domain name lookup. If the domain name, or any constituent label, is already in ASCII, the ASCII should not be altered. Ref. RFC5891, UTS#46

Results

Here is the complete list of libraries with their compliance with the corresponding dataset, with a color indicating if they are UA ready or not. Yellow color indicates that some edge cases are not supported or the library needs to be used along with another one to be compliant.

Legend

UA ready
UA ready but developer needs to be careful
UA not ready

Language	Lib Name	Compliance on dataset (%)	Datasets
c	libcurl	84.3	HEs
c	libidn2	95.2	LA2U ,LU2A
csharp	mailkit	84.3	HEs
csharp	microsoft	83.9	LA2U ,LU2A
go	idna	79	LA2U ,LU2A
go	mail	100	HEs
go	smtp	19.6	HEs
java	commons-validator	85.5	HEs ,HDns
java	guava	77.8	HDns

java	icu	93.5	LA2U ,LU2A
java	jakartamail	82.4	HEs
java	jre	71	LA2U ,LU2A
js	idna-uts46	85.5	LA2U ,LU2A
js	nodemailer	84.3	HEs
js	validator	94.2	HEs ,HDns
python3	django_auth	48.1	HEs ,HId
python3	email_validator	86.3	HEs
python3	encodings_idna	67.7	LU2A ,LA2U
python3	idna	100	LA2U ,LU2A
python3	smtplib	84.3	HEs
rust	idna	87.1	LA2U ,LU2A
rust	lettre	7.8	HEs

Discussion

Detailed results are available at https://uasg.tech/software.

C - Libcurl (EAI)

UA ready but developer needs to be careful

Even if libcurl sends properly the SMTPUTF8 flag, validations done by libcurl are insufficient: we were able to send email to multiple invalid addresses. Developers need to use another library to validate the address before sending it.

Libcurl was chosen because it is by far the most popular and used library in C for sending & downloading content.

C - Libidn2 (IDNA2008)

UA Ready

Supports well IDNA2008 as advertised.

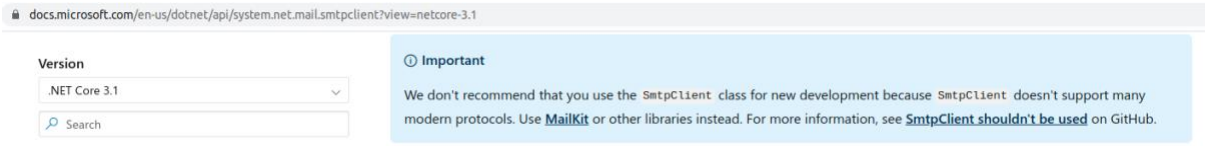
Libidn2 was chosen because it is the next generation of the well established Gnu Libidn (IDNA2003).

C# - Mailkit (EAI)

UA ready but developer needs to be careful

Similar to libcurl, this library supports the STMPUTF8 flag, but multiple invalid email addresses were allowed.

Mailkit was chosen because it is recommended by Microsoft over native C# SmtpClient, since the latter is deprecated by
Microsoft:



C# - Microsoft - System.Globalization.IdnMapping (IDNA2008)

UA Ready

Microsoft provides natively in the .NET framework utilities to convert A-label & U-label. It supports IDNA2008 as advertised. We found some false positives where the library converts domains known as unconvertable.

Go - Idna (IDNA2008)

UA ready but developer needs to be careful

Go provides a library natively compatible with IDNA2008 as advertised. However, we found not only false positives, but also valid domains that net/idna was not able to convert.

Go - Mail (EAI)

UA ready but developer needs to be careful

Go native mail validation/parsing/mailer package. This library is very good for validating and parsing EAI. However, it should not be used to send email since it is based on net/smtp which does not send SMTPUTF8. See next.

Go - Smtp (EAI)

UA not ready

Go native mailer package. Does not support STMPUTF8 flag.

Java - commons-validators (EAI, IDNA2008)

UA not ready

A popular Apache package to validate email addresses, domains and many other things. Since the validation is based on a static list of TLDs, it is not recommended to use as the static list is always not to date. No new version since February 2017.

Java - guava (IDNA2008)

UA not ready

A popular utilities library from Google. As it is advertised: "validation against RFC 3490 ("Internationalizing Domain Names in Applications") is skipped" (IDNA2003). It supports many U-Label since "validation against RFC 1035 is relaxed", see "InternetDomainName.from(input)" method.

Java - ICU (IDNA2008)

UA Ready

Library from the Unicode Consortium. Developers must use a combination of flags to correctly support IDNA2008. The following code show how to initialize the validator:

```
IDNA validator = IDNA.getUTS46Instance(
    IDNA.NONTRANSITIONAL_TO_ASCII
  | IDNA.NONTRANSITIONAL_TO_UNICODE
  | IDNA.CHECK_BIDI
  | IDNA.CHECK_CONTEXTJ
  | IDNA.CHECK_CONTEXTO
  | IDNA.USE_STD3_RULES)
```

Java - Jakartamail (EAI)

UA Ready

Native library (previously named Javamail) to send mail. Now under the stewardship of the Eclipse foundation. Supports correctly EAI since 1.6.4. Version 1.6.5 was verified.
Since email address validation is made automatically before sending the email, this library is a good choice.

Java - JRE - java.net.IDN (IDNA2008)

UA not ready

Native lib to convert U-label and A-label. Based on IDNA2003. Not recommended.

Javascript - idna-uts46 (IDNA2008)

UA ready but developer needs to be careful

Converter for U-label and A-label. Despite its name, it is possible to configure the lib for IDNA2008 only (transitional = false). It doesn't implement Bidi and contextual rules for validation.

This library seems the only one available for the task.

Javascript - nodemailer (EAI)

UA ready but developer needs to be careful

Mailer for NodeJs. Supports SMTPUTF8 flag. Nodemailer doesn't validate email addresses thoroughly, therefore it must be used with an email validation library like "validator", see next.

Javascript - validator (EAI)

UA Ready

Popular javascript package for various validation. Very good compliance on our dataset, can be used to validate & normalize email before sending email with nodemailer for instance.

Python 3 - django auth (EAI)

UA not ready

Popular web package to manage authentication with emails. Non compliant over basic international email addresses. Maintainers seems to have rejected the following pull request that would have make the lib compliant (identified in UASG018):

https://github.com/django/django/pull/7039/commits/2fd6c41461fbcb837b24fab2c2626f9d19e65c60

Here is an extract of the discussion made on the issue:

"RFC 6531 does define a new SMTPUTF8 extension (http://tools.ietf.org/html/rfc6531) to allow (notably) non-ASCII chars in email addresses. Usage seems to be very scarce however at this time. Allowing non-ASCII chars when 95% of mail servers do currently not seem to support that is debatable." - Claude Paroz
https://code.djangoproject.com/ticket/21859#comment:5

Python 3 - email_validator (EAI)

UA Ready

Popular lib to validate & normalize email addresses. Very good compliance over the dataset.

Python 3 - encodings_idna (IDNA2008)

UA not ready

Native converter for U-label and A-label. Based on IDNA2003. Not recommended.

Python 3 - idna (IDNA2008)

UA Ready

Very good IDNA2008 compliant library. The most compliant library with our dataset we tested until now. Compliant with IDNA2008 as advertised. Highly recommended.

Python 3 - smtplib (EAI)

UA ready but developer needs to be careful

Native library for sending email. Supports SMTPUTF8 flag. Needs to be used with an email validation/normalization library like email_validator before sending the email.

Rust - idna (IDNA2008)

UA Ready

Native library for conversion between U-label & A-label. Supports IDNA2008, good compliance on the datasets.

Rust - lettre (EAI)

UA not ready

Native library for sending email. Since Lettre uses an EmailAddress object that works only with ASCII addresses, Lettre does not support EAI even if SMTUTF8 supports is advertised. See https://docs.rs/emailaddress/0.4.0/emailaddress/.

Conclusion

Many of the new tested libraries claim to support or support recently IDNA2008 and EAI. To name a few:

- C - libcurl starts to support EAI this year at version 7.69.0 (we tested 7.70.0), see:https://github.com/curl/curl/pull/4892;
- Java - jakarta mail (previously java mail) starts to stabilize the support at version 1.6.4 (last year, see bug fixes on UTF-8 handling https://github.com/eclipse-ee4j/mail/releases/tag/1.6.4), we tested v1.6.5.
- C# - Microsoft IdnMapping was supporting IDNA2003 prior to Windows 8 et make the switch;
- Python 3 - smtplib starts to support EAI at version of Python 3.5, we tested 3.8;

Therefore, it is good to see progression towards Universal Acceptance.

A useful feature for developers would be that SMTP client libraries validate and normalize the email address before sending it. We noticed that Jakarta Mail seems to be the only one to do both tasks. Some libs, like libcurl, transform the domain part to an A-label beforehand, but don't check the local part, nor normalize. Nevertheless, really good libs to validate and normalize are available.

Dependencies were found to be troublesome. For example, Go net/mail rely on a smtp lib that doesn't support SMTPUTF8 even if net/mail is perfectly compliant with internationalization RFCs for parsing. Rust Lettre claims to send SMTPUTF8 flag but uses internally an EmailAddress object supporting only ASCII addresses.

Detailed results of the tests are available at: https://uasg.tech/wp-content/uploads/documents/Test-Report-1.html