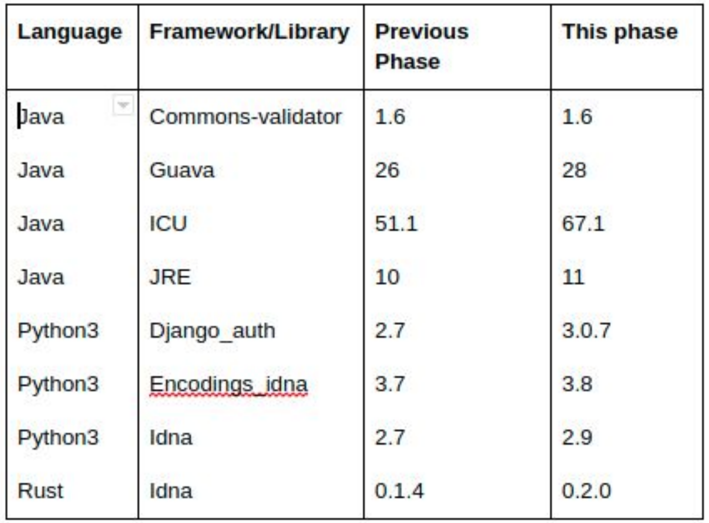
Universal Acceptance Compliance of Some Programming Language Libraries and Frameworks

Guillaume Blanchet, Marc Blanchet
guillaume.blanchet@viagenie.ca, marc.blanchet@viagenie.ca
Viagénie

December 2020

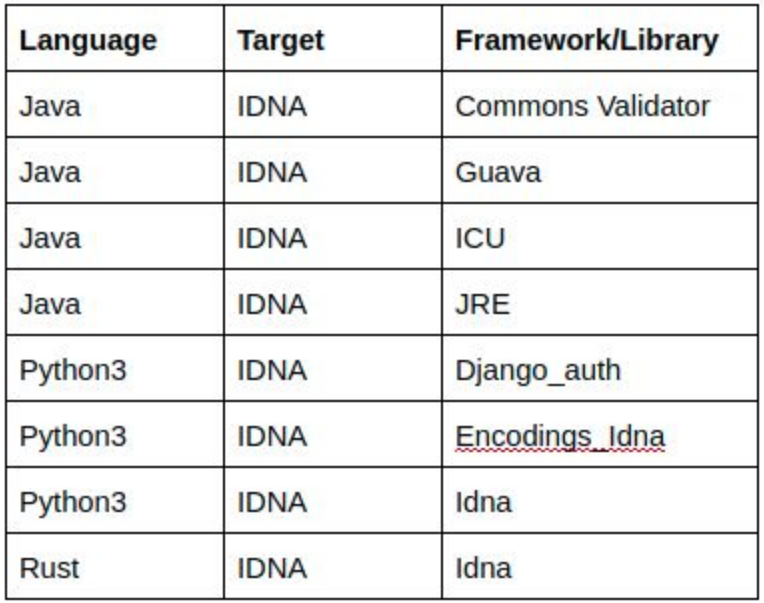
Compliance of Libraries and Frameworks

- Second phase: added libraries & frameworks;
- Introducing checks on email address internationalization (EAI);
- Revalidate old validations with corresponding library new versions:



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

Phase 1 (8 libs from 3 languages)



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

Phase 2 (22 libs from 7 languages)

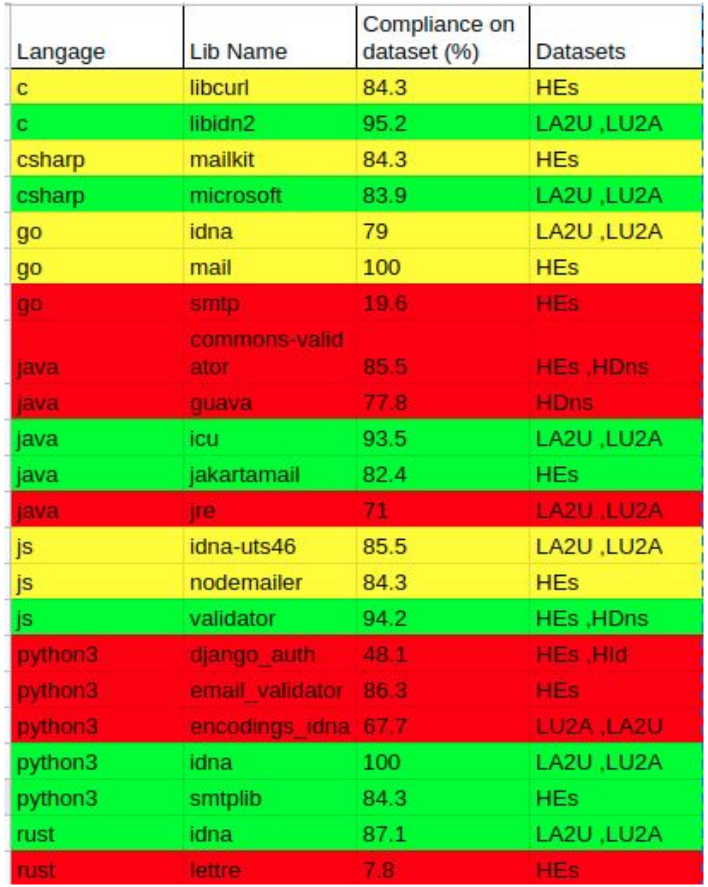
- Use of latest libs version during testing
- No regression from Phase 1
- Use of 5 datasets of sample IDNs and EAIs
- Use of fork supporting SMTPUTF8 from the fake SMTP server Mailhog
- Favor native libs when available
- Use of docker to isolate all these languages & libs

Language	Target	Framework/Library
C	EAI	libcurl
C	IDNA	libidn2
C#	EAI	Mailkit
C#	IDNA	Microsoft
Go	IDNA	Idna
Go	EAI	Mail
Go	EAI	Smtp
Java	IDNA	Commons-Validator
Java	IDNA	Guava
Java	IDNA	ICU
Java	EAI	Javamail/JakartaMail
Java	IDNA	JRE
Javascript	IDNA	Idna-uts46
Javascript	EAI	Nodemailer
Javascript	EAI	Validator
Python3	IDNA	Django_auth
Python3	EAI	Email_Validator
Python3	IDNA	Encodings_Idna
Python3	IDNA	Idna
Python3	EAI	Smtplib
Rust	IDNA	Idna
Rust	EAI	Lettre

Results

- Pitfalls: we saw 2 libs (Rust Lettre & Go mail) claiming to support EAI without truly supporting it due to their non compliant dependencies;
- Many libs do part of the job: e.g. libcurl & mailkit supports SMTPUTF8, but don't normalize email local part or don't validate the address properly;
- We see a progression toward universal acceptance (e.g. Microsoft was supporting IDNA2003 in Windows 8 and make the switch to IDNA2008)
- We didn't see a lib that does the complete job for EAI, i.e.:
    - Validating and normalizing the localpart & domain part
    - Transforming the domain part into an A-label

Summary



Langage	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-valid ator	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

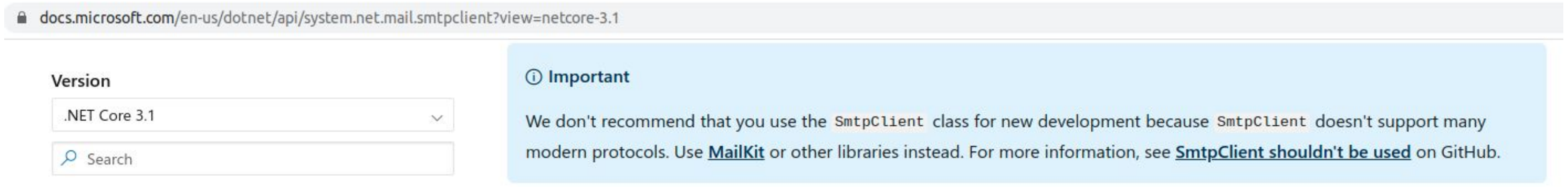
Breakdown: C language

🟨 Libcurl (EAI): developers needs to validate address before sending, supports SMTPUTF8 flag.

🟩 Libidn2 (IDNA2008): Supports well IDNA2008 as advertised

Breakdown: C# language

🟨 Mailkit (EAI): Like libcurl. Replace the native SmtpClient from Microsoft:


docs.microsoft.com/en-us/dotnet/api/system.net.mail.smtpclient?view=netcore-3.1

Version: .NET Core 3.1

Search

Important

We don't recommend that you use the `SmtpClient` class for new development because `SmtpClient` doesn't support many modern protocols. Use MailKit or other libraries instead. For more information, see SmtpClient shouldn't be used on GitHub.

🟩 Microsoft - System.Globalization.IdnMapping (IDNA2008): Supports well IDNA2008 as advertised. Found some false positives: converts domains known as unconvertible

Breakdown: Go language

🟨 Idna (IDNA2008): Some valid domains were not convertible

🟥 Smtp (EAI): Does not support STMPUTF8 flag

🟨 Mail (EAI): very good for validating and parsing EAI but doesn’t support SMTPUTF8 as it relies on Go - Smtp...

Breakdown: Java language

- 🟥 commons-validators (EAI, IDNA2008): validation is based on a static list of TLDs outdated by definition
- 🟥 guava (IDNA2008): "validation against RFC 3490 ("Internationalizing Domain Names in Applications") is skipped" (IDNA2003)
- 🟥 JRE - java.net.IDN (IDNA2003): Based on IDNA2003
- 🟩 ICU (IDNA2008): Library from the Unicode Consortium. Developers must use the right combination of parameters to validate with IDNA2008.
- 🟩 Jakartamail (EAI): Supports correctly EAI since 1.6.4. Version 1.6.5 was verified
- 🟩 Xcode (IDN2008): Lib from Verisign. Implements perfectly IDNA2008. Processing of domains is slow (takes up to 5 seconds to process certain domains). No maven repository.

Breakdown: Javascript language

🟨 idna-uts46 (IDNA2008): doesn't implement Bidi and contextual rules for validation

🟨 nodemailer (EAI): Supports SMTPUTF8 flag. Nodemailer doesn't validate email addresses

🟩 Javascript - validator (EAI): Very good compliance on our dataset, can be used to validate & normalize email before sending email with nodemailer for instance.

Breakdown: Python language

🟥 django auth (EAI): Non compliant over basic international email addresses

🟥 encodings_idna (IDNA2003): native lib, IDNA2003 only

🟨 smtplib (EAI): Supports SMTPUTF8 flag. Needs to be used with an email validation/normalization library like email_validator before sending

🟩 idna (IDNA2008): The most compliant library with our dataset we tested until now.

🟩 email_validator (EAI): Popular lib to validate & normalize email addresses. Very good compliance over the dataset

Breakdown: Rust language

🟥 Rust - lettre (EAI): Since Lettre uses an EmailAddress object that works only with ASCII addresses, Lettre does not support EAI even if SMTUTF8 supports is advertised

🟩 idna (IDNA2008): Supports IDNA2008, good compliance on the datasets

Questions?

Report references:

- Report: https://uasg.tech/wp-content/uploads/documents/UASG018A-en-digital.pdf
- Viagénie ua: https://viagenie.ca/ua/
- Detail results: https://viagenie.ca/ua/test-results-20200814.html
- Verisign Xcode lib: https://www.verisign.com/en_US/channel-resources/domain-registry-products/idn-sdks/index.xhtml
- Other libs are available with the standard means (Maven for Java, Pip for Python, etc.)

UA References

- Use Cases for UA Readiness Evaluation, UASG-0004
- http://uasg.tech
- Reviewing Programming Languages and Frameworks for Compliance with Universal Acceptance Good Practice, UASG-018
- Evaluation of Software libraries for UA Readiness: http://uasg.tech/software
- Universal Acceptance Readiness Report: UASG-FY20

IDN References

- Klensin, J., "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", RFC 5890, DOI 10.17487/RFC5890, August 2010, <https://www.rfc-editor.org/info/rfc5890>.
- Klensin, J., "Internationalized Domain Names in Applications (IDNA): Protocol", RFC 5891, DOI 10.17487/RFC5891, August 2010, <https://www.rfc-editor.org/info/rfc5891>.
- Faltstrom, P., Ed., "The Unicode Code Points and Internationalized Domain Names for Applications (IDNA)", RFC 5892, DOI 10.17487/RFC5892, August 2010, <https://www.rfc-editor.org/info/rfc5892>.
- Alvestrand, H., Ed., and C. Karp, "Right-to-Left Scripts for Internationalized Domain Names for Applications (IDNA)", RFC 5893, DOI 10.17487/RFC5893, August 2010, <https://www.rfc-editor.org/info/rfc5893>.
- Klensin, J., "Internationalized Domain Names for Applications (IDNA): Background, Explanation, and Rationale", RFC 5894, DOI 10.17487/RFC5894, August 2010, <https://www.rfc-editor.org/info/rfc5894>.
- Costello, A., "Punycode: A Bootstring encoding of Unicode for Internationalized Domain Names in Applications (IDNA)", RFC 3492, DOI 10.17487/RFC3492, March 2003, <https://www.rfc-editor.org/info/rfc3492>.

EAI References

- Klensin, J. and Y. Ko, "Overview and Framework for Internationalized Email", RFC 6530, DOI 10.17487/RFC6530, February 2012, <https://www.rfc-editor.org/info/rfc6530>.
- Yao, J. and W. Mao, "SMTP Extension for Internationalized Email", RFC 6531, DOI 10.17487/RFC6531, February 2012, <https://www.rfc-editor.org/info/rfc6531>.
- Yang, A., Steele, S., and N. Freed, "Internationalized Email Headers", RFC 6532, DOI 10.17487/RFC6532, February 2012, <https://www.rfc-editor.org/info/rfc6532>.
- Hansen, T., Ed., Newman, C., and A. Melnikov, "Internationalized Delivery Status and Disposition Notifications", RFC 6533, DOI 10.17487/RFC6533, February 2012, <https://www.rfc-editor.org/info/rfc6533>.
- Levine, J. and R. Gellens, "Mailing Lists and Non-ASCII Addresses", RFC 6783, DOI 10.17487/RFC6783, November 2012, <https://www.rfc-editor.org/info/rfc6783>.
- Resnick, P., Ed., Newman, C., Ed., and S. Shen, Ed., "IMAP Support for UTF-8", RFC 6855, DOI 10.17487/RFC6855, March 2013, <https://www.rfc-editor.org/info/rfc6855>.
- Gellens, R., Newman, C., Yao, J., and K. Fujiwara, "Post Office Protocol Version 3 (POP3) Support for UTF-8", RFC 6856, DOI 10.17487/RFC6856, March 2013, <https://www.rfc-editor.org/info/rfc6856>.
- Fujiwara, K., "Post-Delivery Message Downgrading for Internationalized Email Messages", RFC 6857, DOI 10.17487/RFC6857, March 2013, <https://www.rfc-editor.org/info/rfc6857>.
- Gulbrandsen, A., "Simplified POP and IMAP Downgrading for Internationalized Email", RFC 6858, DOI 10.17487/RFC6858, March 2013, <https://www.rfc-editor.org/info/rfc6858>.