

# Programming (in Java) to Support Universal Acceptance

**Regional Universal Acceptance Training Program  
NARALO-ICANN Collaboration**

Sarmad Hussain, ICANN  
Akshat Joshi, ThinkTrans

3 February 2022



# Many thanks!



[www.naralo.org](http://www.naralo.org)

<https://atlarge.icann.org/ralos/naralo>



<https://uasg.tech>

# Housekeeping

---

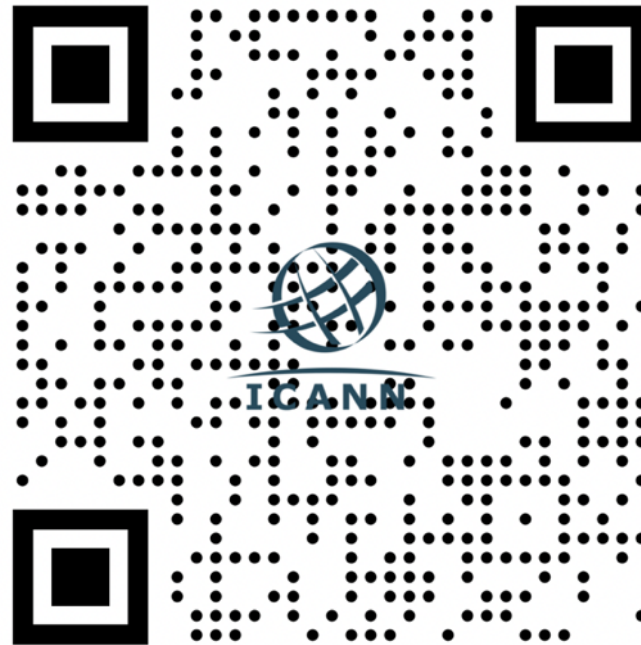
1. Please mute your mic when you are not speaking
2. Please type your question into the chat box or raise your hand. We will answer questions at the end of each section
3. ICANN Standards of Behavior

# Housekeeping

---

Join us on Telegram!

1. **Install Telegram app [do this before scanning the code below]:** <https://telegram.org/> (or visit your favorite app store)
2. Scan the QR code below or click here <https://t.me/+WRht2Xzql5BkZGJh>



# Universal Acceptance (UA) training

## Overview

Date	Session	Audience	Description
20 January 2022 12:00 PM EST 17:00 PM UTC	Introduction to UA	General	An introduction to the fundamentals of UA and EAI.
27 January 2022 12:00 PM EST 17:00 PM UTC	EAI Configuration	Technical (email and system administrators)	A detailed training on how to configure email systems to support EAI.
3 February 2022 12:00 PM EST 17:00 PM UTC	UA for Java Software and Application Developers	Technical (software and application developers)	A detailed training on how to design and develop applications and systems to support UA.
10 February 2022 12:00 PM EST 17:00 PM UTC	Outreach and Engagement: The Life of a UA Ambassador	General	A session to discuss how participants can stay involved in UA efforts across the NA region.



# Welcome Dr. Sarmad Hussain and Prof. Akshat Joshi



Dr. Sarmad Hussain  
Sr. Director IDN & UA  
Programs  
ICANN  
sarmad.Hussain@icann.org



Akshat Joshi  
Founder - ThinkTrans  
<https://www.linkedin.com/in/akshat-joshi-a56100108/>

# Programming (in Java) to Support Universal Acceptance

**Regional Universal Acceptance Training Program  
NARALO-ICANN Collaboration**

Sarmad Hussain, ICANN  
Akshat Joshi, ThinkTrans

3 February 2022



- ⦿ Introduction (5 minutes)
- ⦿ Overview of Universal Acceptance (10 minutes)
  - Quiz (5 minutes)
- ⦿ Fundamentals for IDNs and EAI (20 minutes)
  - Quiz (5 minutes)
- ⦿ Programming for UA using Java (45 minutes)
  - Processing Domain Names
  - Processing Email Address
  - Storing Domain Names and Email Addresses
  - Domain Names and Email Addresses on Android Platform
- ⦿ Conclusion (10 minutes)
- ⦿ Q/A



# Overview of Universal Acceptance

## **Goal**

All domain names and email addresses work in all software applications.

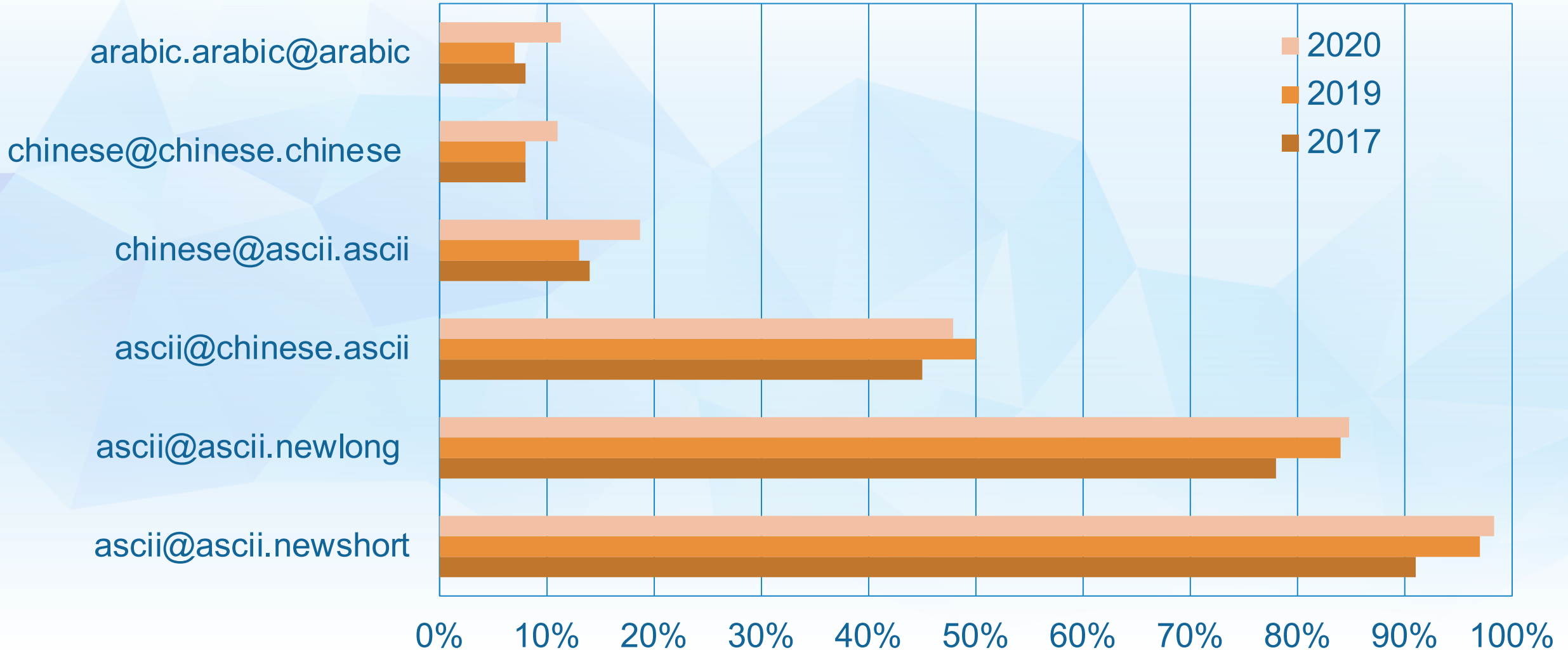
## **Impact**

Promote consumer choice, improve competition, and provide broader access to end users.

- ⦿ It's now possible to have domain names and email addresses in local languages using UTF8.
  - Internationalized Domain Names (IDNs)
  - Email Address Internationalization (EAI)
- ⦿ Domain names
  - **Newer** top-level domain names      example.sky
  - **Longer** top-level domain names      example.abudhabi
  - **Internationalized** Domain Names      παράδειγμα.eu
- ⦿ Internationalized email addresses (EAI)
  - ASCII@IDN      marc@société.org
  - UTF8@ASCII      angéle@example.com
  - UTF8@IDN      测试@普遍接受-测试.世界
  - UTF@IDN; right-to-left scripts      ای-میل@مثال.موقع

# Acceptance of Email Addresses by Websites Globally

For details, see [UASG027](#)



## 1. Support All Domain Names



Accept



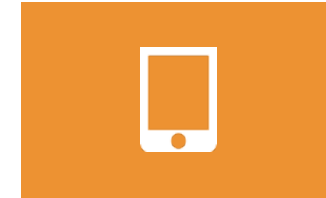
Validate



Process



Store



Display

## 2. Support All Email Addresses



Accept



Validate



Process



Store



Display

## Applications and Websites

- Wikipedia.org, ICANN.org, Amazon.com, custom websites globally
- PowerPoint, Google-Docs, Safari, Acrobat, custom apps

## Social Media and Search Engines

- Chrome, Bing, Safari, Firefox, local (e.g., Chinese) browsers
- Facebook, Instagram, Twitter, Skype, WeChat, WhatsApp, Viber

## Programming Languages and Frameworks

- JavaScript, Java, Swift, C#, PHP, Python
- Angular, Spring, .NET core, J2EE, WordPress, SAP, Oracle

## Platforms, Operating Systems and System Tools

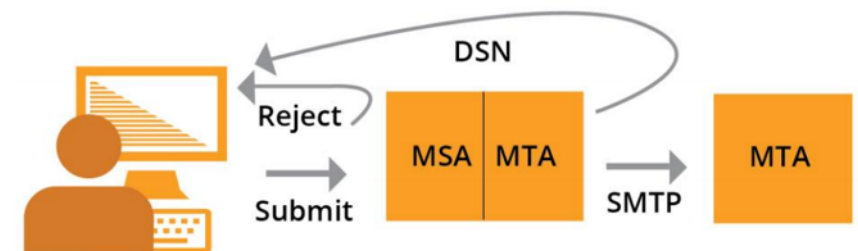
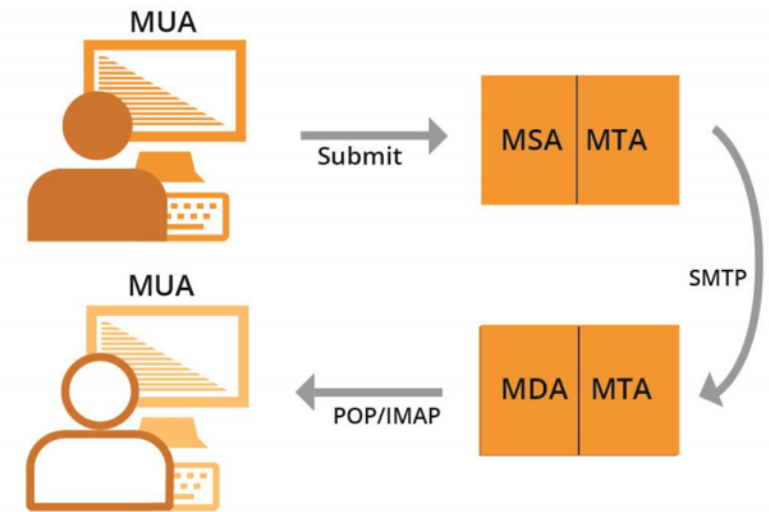
- iOS, Windows, Linux, Android, App Stores
- Active Directory, OpenLDAP, OpenSSL, Ping, Telnet

## Standards and Best Practices

- IETF RFCs, W3C HTML, Unicode CLDR, WHATWG
- Industry-based standards (health, aviation, ...)

Accept, validate, process,  
store and display  
all domain names and  
email addresses.

- All email agents must be configured to send and receive internationalized email addresses. See [EAI: A Technical Overview](#) for details.
  - **MUA** – Mail User Agent: A client program that a person uses to send, receive, and manage mail.
  - **MSA** – Mail Submission Agent: A server program that receives mail from a MUA and prepares it for transmission and delivery.
  - **MTA** – Mail Transmission Agent: A server program that sends and receives mail to and from other Internet hosts. An MTA may receive mail from an MSA and/or deliver mail to an MDA.
  - **MDA** – Mail Delivery Agent: A server program that handles incoming mail and typically stores it in a mailbox or folder.



# Quiz



- ⦿ To enhance systems to be Universal Acceptance (UA) ready, which of the following categories of domain names and email addresses are relevant?
  1. ASCII domain names.
  2. Internationalized Domain Names (IDNs).
  3. Internationalized email addresses (EAI).
  4. All the above.
  5. Only 2 and 3.

# Fundamentals for Internationalized Domain Names and Email Addresses

- ⦿ Supporting Unicode input, processing and display, in UTF-8 format.
- ⦿ Normalization of Unicode string (in NFC form) for string comparison, etc.
- ⦿ Conversion between U-label and A-label forms of an internationalized domain name.
- ⦿ Validating labels based on Unicode and IDNA2008.
- ⦿ Validating new, longer and internationalized top-level domains.
- ⦿ Resolving internationalized domain names.
- ⦿ Validating internationalized email addresses.
- ⦿ Sending emails using internationalized email addresses.

- ⦿ Unicode encodes glyphs into codepoints for different scripts of the world.
  - Codepoints shown in hex using the U+XXXX notation.
  - Unicode files typically in UTF8 format, using a variable number of bytes for a codepoint.
  - ASCII is used as is in Unicode: `e = ASCII 65 = U+0065`.
- ⦿ There are multiple ways to encode certain glyphs in Unicode:
  - `è = U+00E8`
  - `e + ` = è = U+0065 + U+0300`
- ⦿ Normalization ensures that the end representation is the same, even if users type differently.
  - IDN standards recommend using [Normalization Form C \(NFC\)](#).
  - Generates `U+00E8` for both input versions above.

- ⦿ A domain name is an ordered set of labels or strings: [www.example.co.uk](http://www.example.co.uk).
  - The top-level domain (TLD) is the rightmost label: "uk"
  - Initially, TLDs were only two or three characters long (e.g., [.ca](http://.ca), [.com](http://.com)).
  - Now TLDs can be longer strings (e.g., [.info](http://.info), [.google](http://.google), [.engineering](http://.engineering)).
  - TLDs delegated in the [root zone](#) can change over time, so a fixed list can get outdated.
- ⦿ Domain names can also be internationalized when one of the labels contains at least one non-ASCII character.
  - For example: [www.exâmples.ca](http://www.exâmples.ca) or [普遍接受-测试.世界.](#)
- ⦿ Use the latest IDN standard called IDNA2008 for IDNs.
  - Do not use libraries for the outdated IDNA2003 version.

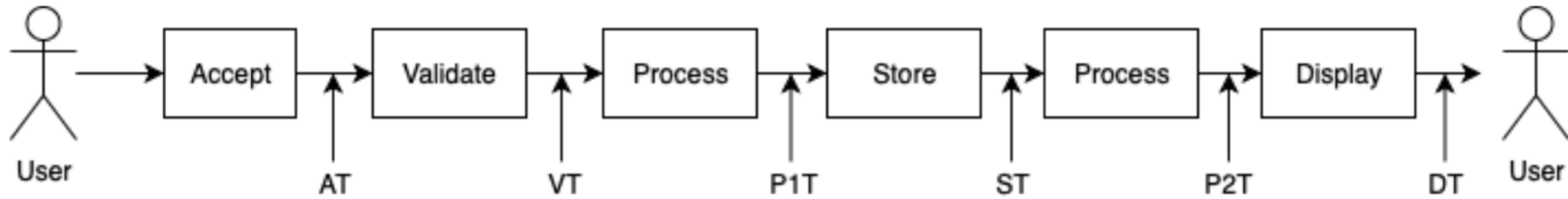
- ⦿ There are two equivalent forms of IDN domain labels: U-label and A-label.
  - Human users use the IDN version called U-label (using UTF-8 format): [exâmp<sup>le</sup>](#)
  - Applications or systems internally use an ASCII equivalent called A-label:
    1. Take user input and normalize and check against IDNA2008 to form IDN U-label.
    2. Convert U-label to punycode (using RFC3492).
    3. Add the “xn--” prefix is added to identify the ASCII string as an IDN A-label.
      - [exâmp<sup>le</sup>](#) => [exmp<sup>le</sup>-xta](#) => [xn--exmp<sup>le</sup>-xta](#)
      - [普遍接受-测试](#) => [--f38am99bqvcd5liy1cxsg](#) => [xn----f38am99bqvcd5liy1cxsg](#)

- ⦿ Email address syntax: `mailboxName@domainName`
  - EAI has the mailboxName in Unicode (in UTF8 format).
  - The domainName can be ASCII or IDN.
    - For example: `kévin@example.org` or `すし@快手.游戏.`

- ⦿ Some applications are still verifying domain names incorrectly by using one of the outdated methods:
  - Check for a fixed length of TLD between 2-4 characters (TLD can be up to 63 characters).
  - Check from a fixed set of TLDs, e.g., using static list of strings.
  - Check for only ASCII characters.
  
- ⦿ Some applications do not cater to additional requirements for validating EAI:
  - Check mailbox name to be a valid string in UTF-8 format.
  - DomainName can be ASCII or IDN.



- Based on [UASG026](#), the application components can be generalized to put emphasis on the processing of internationalized identifiers. Each gate has its own set of requirements and processing.



- AT: Accept test
- VT: Validate test
- P1T: Process test on the input
- ST: Store test
- P2T: Process test on the output
- DT: Display test

- Test data for domain names and email addresses provided in [UASG004](#). Developers are strongly encouraged to use these test cases for unit testing.
- Domain Names
  - 다국어도메인이용환경테스트.한국
  - מבחן-קבלה-אוניברסיטת-קולומביה
  - どこでもつかえる.みんな
  - ಸಾರ್ವತ್ರಿಕ-ಸ್ವೀಕಾರಾರ್ಹತೆ-ಪರೀಕ್ಷೆ.ಭಾರತ
  - ユニバーサルアクセプタンス.クラウド
  - ສາກົນ-ກາບຮອມສ໌ບ-ທົດລອງ.ລາວ
  - Universales-Akzeptanz-Test.vermögensberatung
  - സാർവത്രിക-സ്വീകാര്യതാ-പരിശോധന.ഭാരതം
  - ...
- Email Addresses
  - email-test@universal-acceptance-test.international
  - email-test@universal-acceptance-test.icu
  - تجربة-بريد-الالكتروني@تجربة-القبول-الشامل.موريتانيا
  - Էլփոստ-թեստ@համընդհանուր-ընկալում-թեստ.hայ
  - почта-тест@универсальное-принятие-тест.москва
  - ईमेल-परीक्षण@सार्वभौमिक-स्वीकृति-परीक्षण.संगठन
  - ფოსტის-ტესტი@უნივერსალური-თავსობადობის-ტესტი.გე
  - ...

- ⦿ Validating user input, or any input, is extremely useful for various reasons, some of which include: a better user experience, increased security, and avoiding irrelevant issues.
- ⦿ Validating domain names and email addresses is useful.
- ⦿ Some validation methods for domain names and email addresses:
  - Basic syntax checks: is the syntax of the string correct?
    - Does the domain name contain '.' ?
    - Does the email address contain '@' and a valid domain name part?
  - Functional checks: does the domain name or email address work?
    - Is the top-level domain (TLD) in use?
    - Is the whole domain name in use?
    - Is the email in use?

- ⦿ Validating syntax:
  - ASCII: RFC1035
    - Composed of letters, digits, and hyphen.
    - Max length is 255 octets with each label up to 63 octets.
  - IDN: IDNA2008 (RFCs 5890-5894)
    - Valid A-labels
    - Valid U-labels
  
- ⦿ Validating function:
  - Is the top-level domain (TLD) in use?
    - Verify against the list of TLDs.
    - Verify using a DNS request.
  - Is the whole domain name in use?
    - Verify using a DNS request.

- ⦿ After validation, a software would then use the domain name identifier as:
  - A domain name to be resolved in the DNS.
- ⦿ Therefore, to be UA compliant, the software has to use proper methods that support UA.
  - For example, passing a U-Label to the traditional functions or methods may not succeed, as it is not expecting a UTF8 domain name.

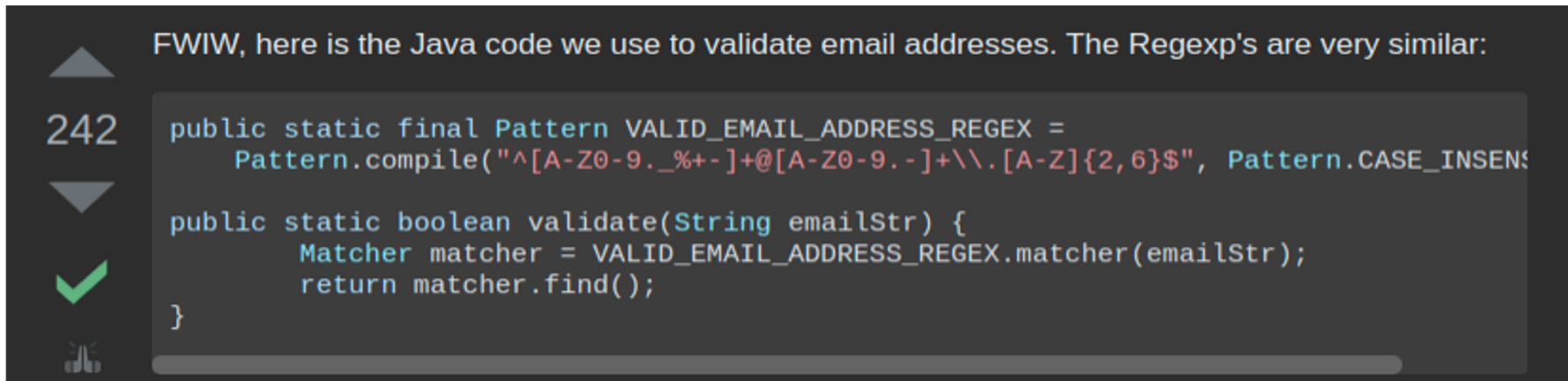
- ⦿ An email address is composed of: mailboxName@domainName
- ⦿ Validating syntax:
  - For domainName, see earlier discussion.
  - For mailboxName:
    - ASCII
    - UTF8 (for EAI)
- ⦿ Validating function:
  - Is the domain name set up to send and receive emails?
  - Is the mailbox name able to send and receive emails?

- ⦿ After validation, a software would then use the email identifier as:
  - An email-address based user id.
  - To send an email.
- ⦿ Therefore, to be UA compliant, the software must use proper methods that support UA.
  - For example, passing an UTF8 mailbox name email address to a mail sender may not succeed, as it is not expecting a UTF8 mailbox name in the email address.

# Quiz



- A company built a website where international consumers can subscribe via their email. Since the subscription form is user input, developers validated the email address before trying to send the email.
  - Developers went to Stackoverflow and found a regular expression (regex) to perform the validation:



FWIW, here is the Java code we use to validate email addresses. The Regexp's are very similar:

```
public static final Pattern VALID_EMAIL_ADDRESS_REGEX =
    Pattern.compile("^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,6}$", Pattern.CASE_INSENSITIVE);

public static boolean validate(String emailStr) {
    Matcher matcher = VALID_EMAIL_ADDRESS_REGEX.matcher(emailStr);
    return matcher.find();
}
```

- The regex limits mailbox to letters A-Z, digits 0-9, and some symbols, the domain labels to letters, digits and hyphen, and the top-level domain to letters only with length 2-6.
- Would this regex work for the company's website? Why or why not?

# Programming for UA using Java

- ⦿ Code examples were tested against Java 11 (Oracle version) and Android API/SDK 26 where applicable.
  - It is possible that older versions have issues.
- ⦿ Some libraries may require (not necessarily because of UA) newer versions of Java.
- ⦿ Unless explicitly stated, this tutorial should apply to any VM flavor:
  - Oracle, OpenJDK, or Android (Dalvik/ART).

- ⦿ This tutorial shows many libraries as found in the field. While the list is not exhaustive, it is still comprehensive to help you assess which and how to use a library, specially if the software has already been developed.
  - UASG report provides detailed information about the libraries compliant with UA. See <https://uasg.tech/wp-content/uploads/documents/UASG018A-en-digital.pdf>.
- ⦿ Libraries shown in this tutorial have been tested on the current version available at the time of writing.
  - It's possible that new versions of some libraries have fixed issues or made enhancements that would change the recommendations.
  - Please check at the time of your development the status of these libraries.

- ⦿ UA identifiers are domain names and email addresses.
  - These identifiers may contain Unicode UTF8 data.
- ⦿ Java [String type](#) is well-suited to hold those identifiers as it is natively supporting Unicode. Therefore, most libraries expect the String type.
- ⦿ Default charset (`Charset.defaultCharset()`) is typically UTF8 in most systems. Verify (`java -XshowSettings`) or change the default charset in the Java VM you are using.
  - For more info, see [JEP](#).

- ⦿ The code examples throughout the tutorial will use these inputs, which provide basic UA test cases (non-exhaustive):

```
List<String> testDomains = List.of(
    "example.org",           // ascii.ascii
    "example.undefinedtld", // unexistant tld
    "example.recentTld",    // recently allocated tld
    "example.accountants", // allocated longer than 7 char tld
    "exâmples.org",        // ulabel.ascii
    "xn--exmples-xta.org",  // alabel.ascii
    "exâmples.ไทย",       // ulabel.ulabel
    "exâmples.xn--o3cw4h",  // ulabel.alabel
    "xn--exmples-xta.xn--o3cw4h" // alabel.alabel
);
List<String> testLocalParts = List.of(
    "user",
    "k evin"
);
```

# Processing Domain Names

- ⦿ Traditional way of doing hostname resolution and sockets resolution:

```
import java.net.InetAddress;  
getByName(String host);  
getAllByName(String host);  
Socket(String host, int port);
```

- Uses underlying `getByName()`
- ⦿ Throws a `UnknownHostException` for any failure:
  - No IP addresses returned
  - Invalid host
  - Bad syntax
  - ...
- ⦿ Passes the host String as is to the underlying OS, without validation.
  - Therefore, invalid domains (such as invalid U-labels) are passed.
  - Depends on the implementation of the underlying OS.



- ⦿ Do not use as-is with a hostname. Instead:
  - Prepare the hostname (e.g., convert IDN U-label to A-label) using another library and then use these base calls.
  - Validate the hostname before calling `getByName()`
    - To avoid delays waiting for answers of queries that will throw anyway.
    - Provide better feedback to the user because the throw will not tell you if the hostname was wrong or if the hostname was OK, but the query did not return data.

- ⦿ The gold standard library for Unicode. It was developed by IBM and is now managed by Unicode. In sync with Unicode standards.
  - ICU4J (<http://site.icu-project.org/home>) is not really Java-ized. It is a direct mapping to the C version. Java developers may not like it for that reason.
  - IDNA Conversion is based on Unicode [UTS46](#), which supports transition from IDNA2003 to IDNA2008. However, it is possible to configure not to support transition (recommended).
  - IDNA Conversion includes normalization as per IDNA (good!).
  - The output of the methods may contain bad domain names as disallowed characters are replaced by U+FFFD.
  - Check if there are errors in the conversion by calling `info.hasErrors()`.
  - For IDNs, set the options to restrict the validation and use to IDNA2008.
  - The static methods implement IDNA2003, and non-static methods implement IDNA2008.

## ◎ Maven Repository:

```
<dependency>
  <groupId>com.ibm.icu</groupId>
  <artifactId>icu4j</artifactId>
  <version>65.1</version>
</dependency>
```

```
import com.ibm.icu.text.IDNA;
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);
StringBuilder output = new StringBuilder();
IDNA.Info info = new IDNA.Info();
validator.nameToASCII(input, output, info);
if (info.hasErrors()) {}
```

Option not to use [UTS46](#) transitional feature and to enhance validation.

- Utility library developed by Verisign. Has an "Idna" object.
  - [https://www.verisign.com/en\\_US/channel-resources/domain-registry-products/idn-sdks/index.xhtml](https://www.verisign.com/en_US/channel-resources/domain-registry-products/idn-sdks/index.xhtml)
  - No Maven Repository (only a zip downloadable with a jar file in it).
  - Slow (tests show the processing of a domain can take up to 5 seconds) but implements IDNA2008 perfectly.

```
import com.vgrs.xcode.common.Unicode;
import com.vgrs.xcode.idna.Idna;
import com.vgrs.xcode.idna.Punycode;
Idna idna = new Idna(new Punycode(), true, true);
int[] output = idna.domainToUnicode(input.toCharArray());
    // see domainToAscii for roundtrip
String domain = new String(Unicode.decode(output));
```

- ⦿ JRE-IDN
  - Included in JRE but implements IDNA2003.
  
- ⦿ Apache Common Validator
  - Has domain and email validators.
  - Do not use as it relies on a static list of TLDs! **OUTDATED!**
  
- ⦿ Guava
  - Utility library developed by Google.
  - Not useful for validation.
  - If used, be aware that it depends on the public suffix list, statically set into the library, so you would need to update the library frequently.

# Making an HTTP Request

HTTP Client	Accept UTF8 Form	Normalization	Automatic Conversion	IDNA2008
Java 1.1 HttpURLConnection	x	x	x	x
Apache HTTP Client	x	x	x	x
OkHTTP	✓	✓	✓	x
Java 11 HTTP Client	x	x	x	x
Google Java HTTP Client	x	x	x	x

# Processing Email Addresses

# Validating Email



- ◉ Basic: something@something
  - `^(.+)+@(.+)$`
- ◉ From [owasp.org](https://owasp.org) (security):
  - `[^[a-zA-Z0-9_+&*~]+(?:\.[a-zA-Z0-9_+&*~]+)*@(?:[a-zA-Z0-9-]+\.)+[a-zA-Z]{2,7}$`
    - Does not support EAI, i.e., mailbox name in UTF8 not allowed: `[a-zA-Z0-9_+&*~]`
    - Does not support ASCII TLD longer than 7 characters: `[a-zA-Z]{2,7}`
    - Does not support U-labels in IDN TLD: `[a-zA-Z]`
  - But OWASP is THE reference for security.
    - Therefore, you may end up fighting with your security team to use a UA-compatible Regex instead of the “standard” one from OWASP.

- Example of Regex suggested in various forums: ex: [List of proposals](#)
  - `^[A-Za-z0-9+_.-]+@(.+)$` **does not support UTF8 in mailbox name**
  - `^[a-zA-Z0-9_!#$%&'*/+=?`{|}~^.-]+@[a-zA-Z0-9.-]+$` **does not support U-labels**
  - `^[a-zA-Z0-9_!#$%&'*/+=?`{|}~^.-]+(?:\\. [a-zA-Z0-9_!#$%&'*/+=?`{|}~^.-]+)*@[a-zA-Z0-9-]+(?:\\. [a-zA-Z0-9-]+)*$` **does not support U-labels**
  - `^[\\w!#$%&'*/+=?`{|}~^.-]+(?:\\. [\\w!#$%&'*/+=?`{|}~^.-]+)*@(?:[a-zA-Z0-9-]+\\. )+[a-zA-Z]{2,6}$` **have length restrictions for the TLD between 2 – 6 characters**
- One can come up with an EAI-IDN compatible regex using various Unicode codepoints characteristics.
  - For IDN it would be like a reimplementaion of the IDNA protocol tables in regex!
- Given that both sides of an EAI may have UTF8, then one regex for an EAI could be `.*@.*` which is only verifying the presence of the '@' character.

- ⦿ Most used Java package to send email; also has a `validate()` method for email addresses.
- ⦿ `validate()` does a good job in validating email addresses, especially the local part:
  - It verifies illegal characters such as: `()<>,;:"[]\` , whitespaces, etc.
  - It verifies the characters are only digit and letters per the definition of Unicode classes.
  - It does not validate IDN, so add IDN validation and preparation as an additional step.

- ◉ `import javax.mail`

- ◉ **Maven:**

```
<dependency>  
  <groupId>com.sun.mail</groupId>  
  <artifactId>jakarta.mail</artifactId>  
  <version>1.6.5</version>  
</dependency>
```

```
import javax.mail.internet.InternetAddress;  
try {  
  InternetAddress emailAddr = new InternetAddress(input);  
  emailAddr.validate();  
} catch (AddressException e) {}
```

- ◉ Apache Commons Validator
  - Has domain and email validators.
  - Do not use as it relies on a static list of TLDs! **OUTDATED!**
- ◉ EmailValidator4J
  - Claims to support EAI! (not tested)
  - <https://github.com/egulias/EmailValidator4J>

# Sending Email

- ◉ Same library and Maven as before.

```
Properties properties = System.getProperties();
properties.setProperty("mail.smtp.host", host);
Session session = Session.getDefaultInstance(properties);
try {
    MimeMessage message = new MimeMessage(session);
    message.setFrom(new InternetAddress(from));
    message.addRecipient(Message.RecipientType.TO, new InternetAddress(to));
    message.setSubject("This is the Subject Line!");
    message.setText("This is actual message");
    Transport.send(message);
} catch (MessagingException e) {
}
```

- ⦿ Jakarta Mail wrapper; simpler calls to send emails.
  - Uses (and includes) another library for email validation.
  - Uses various Regex.
  - Considers any UTF8 as invalid, therefore no support for U-label as domain or EAI.
  - Internal validation based on the obsolete RFC2822.



- ◉ Dev team realized that the package `com.sun.mail:javax.mail:1.5.6` used to send email subscription confirmation via SMTP already had a "validate" function. They rewrote the `isValidEmail` method:

```
public static boolean isValidEmail(String email) {
    try {
        var iEmail = new javax.mail.internet.InternetAddress(email);
        iEmail.validate();
        return true;
    } catch (AddressException e) {
        return false;
    }
}
```

- ◉ However, they found the method is still failing.

- They found this internationalization feature was corrected in a newer version, so they upgraded to: [com.sun.mail:jakarta.mail v1.6.5](#)

```
public static boolean isValidEmail(String email) {
    try {
        var iEmail = new javax.mail.internet.InternetAddress(email);
        iEmail.validate();
        return true;
    } catch (AddressException e) {
        return false;
    }
}
```

- Finally, by inspecting these fixes in javamail and its renamed version jakartamail, they realized they needed to also modify the subscribe function and their SMTP server to support a new "SMTPUTF8" flag.

- ⦿ The company dev team did part of the job. Emails using IDNs will still be rejected by Jakarta Mail.
- ⦿ Here is the complete example, preparing email address with A-labels in domain, which is then used as input to any libraries/frameworks.
- ⦿ The local part remains UTF8 which may not be working in the mail delivery path.

```
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);

IDNA.Info info = new IDNA.Info();
String localpart = email.substring(0, email.lastIndexOf("@"));
String domain = email.substring(email.lastIndexOf("@") + 1);
StringBuilder output = new StringBuilder();
validator.nameToASCII(domain, output, info);
email = localpart + "@" + output.toString();

if (isEmailValid(email)) {
    subscribe();
} else {
    throw new Exception("Invalid email address, please review it and submit again");
}
```

# Storing Domain Names and Email Addresses

- ⦿ SQL, e.g., MySQL, Oracle, Microsoft SQL Server.
  - Set domain names to max: 255 octets, 63 octets per label.
    - In UTF8 native, variable length.
  - Recommendation to use variable length String columns.
  - Consider/verify the object-relational mapping (ORM) driver/tool if you are using one.
- ⦿ noSQL, e.g., MongoDB, CouchDB, Cassandra, HBase, Redis, Riak, Neo4J.
  - Already UTF8 variable length.

# Domain Names and Email Addresses on Android Platform

- ⦿ Same ICU library, integrated into android OS.
  - No need to add packages dependencies.
  - <https://developer.android.com/reference/android/icu/text/IDNA>
  - Same considerations as discussed before for the ICU4J library.

# Conclusion



# Prog. Languages Support

## UASG018A

LANGUAGE	LIB NAME	COMPLIANCE (%)	Type
Javascript	Idna-Uts46	85.5	IDN
Javascript	Nodemailer	84.3	Mail
Javascript	Validator	94.2	Mail
Python3	Django_Auth	48.1	Mail
Python3	Email_Validator	86.3	Mail
Python3	Encodings_Idna	67.7	IDN
Python3	<u>Idna</u>	100	IDN
Python3	<u>Smtplib</u>	84.3	Mail
Rust	<u>Idna</u>	87.1	IDN
Rust	<u>Lettre</u>	7.8	Mail

LANGUAGE	LIB NAME	COMPLIANCE (%)	Type
C	Libcurl	84.3	Mail
C	Libidn2	95.2	IDN
C#	Mailkit	84.3	Mail
C#	Microsoft	83.9	IDN
Go	Mail	100	Mail
Go	<u>Idna</u>	79	IDN
Go	Smtplib	19.6	Mail
Java	Commons-Validator	85.5	Mail, IDN
Java	Guava	77.8	IDN
Java	ICU	93.5	IDN
Java	Jakartamail	82.4	Mail
Java	JRE	71	IDN

# Prog. Languages Support (Including Mobile Platforms)

LANGUAGE	LIB NAME		PLATFORM
PHP	cURL*		Windows
			Linux
PHP	mail		Windows
PHP	emailValidator		Windows & Linux
PHP	Guzzle		Windows & Linux
PHP	intl		Windows & Linux
PHP	PHPMailer		Windows & Linux
PHP	Symfony	Http-client	Windows & Linux
		Polyfill-intl-idn	Windows & Linux
		Mailer	Windows & Linux

LANGUAGE	LIB NAME	PLATFORM
Swift	MessageUI	IOS
Swift	URLSession	IOS
Swift	Alamofire	IOS
Swift	IDNA-Cocoa	IOS
Kotlin	okHttp	Android
Kotlin	HttpURLConnection	Android
Kotlin	Retrofit	Android
Kotlin	fuel	Android
Kotlin	Volley	Android
Kotlin	Apache HttpClient	Android
Kotlin	Jakarta Mail	Android
Kotlin	Email Intent	Android

- ⦿ Be aware that UA identifiers may not be fully supported in software and libraries.
- ⦿ Use the right libraries and frameworks.
- ⦿ Adapt your code to properly support UA.
- ⦿ Do unit and system testing using UA test cases to ensure that your software is UA ready.

**Get Involved!**

- ⦿ For more information on UA, email [info@uasg.tech](mailto:info@uasg.tech) or [UAProgram@icann.org](mailto:UAProgram@icann.org)
- ⦿ Access all UASG documents and presentations at: <https://uasg.tech>

---

- ⦿ Access details of ongoing work from wiki pages: <https://community.icann.org/display/TUA>
- ⦿ Register to participate or listen in the UA discussion list at: <https://uasg.tech/subscribe>
- ⦿ Register to participate in UA working groups [here](#).

- ⦿ See <https://uasg.tech> for a complete list of reports.
  - Universal Acceptance Quick Guide: [UASG005](#)
  - Introduction to Universal Acceptance: [UASG007](#)
  - Quick Guide to EAI: [UASG014](#)
  - EAI – A Technical Overview: [UASG012](#)
  - UA Compliance of Some Programming Language Libraries and Frameworks – [UASG018A](#)
  - EAI – Evaluation of Major Email Software and Services: [UASG021B](#)
  - Universal Acceptance Readiness Framework: [UASG026](#)
  - Considerations for Naming Internationalized Email Mailboxes: [UASG028](#)
  - Evaluation of EAI Support in Email Software and Services Report: [UASG030](#)
  - UA of Content Management Systems (CMS) Phase 1 – WordPress: [UASG032](#)

# Engage with ICANN – Thank You and Questions



One World, One Internet

Visit us at [icann.org](https://icann.org)

Email: [sarmad.hussain@icann.org](mailto:sarmad.hussain@icann.org)



[@icann](https://twitter.com/icann)



[facebook.com/icannorg](https://facebook.com/icannorg)



[youtube.com/icannnews](https://youtube.com/icannnews)



[flickr.com/icann](https://flickr.com/icann)



[linkedin/company/icann](https://linkedin/company/icann)



[slideshare/icannpresentations](https://slideshare/icannpresentations)



[soundcloud/icann](https://soundcloud/icann)