Universal Acceptance

UA Remediation for Top Global Websites

May 2023

TABLE OF CONTENTS

About EVARIS

EVARIS Systems LLP – An Evolutionary Artificial Intelligence Solutions and Systems LLP – is based in India. It features a team with experience in the field of ICT Solutions including quality assurances, and more specifically, Internationalized Domain Names (IDNs) and email addresses.

The EVARIS Systems LLP team members have experience in the field of Indian language computing, natural language processing, including LTR, RTL scripts/languages, and formulation of IDN policy in close coordination with the Ministry of Government for Indian languages. Activities also include work on variant generation, homographic for identifying confusingly similar characters, reserved list and finally implementation by registries by way of APIs.

Introduction

The goal of Universal Acceptance (UA) is to ensure that every valid domain name and email address can be used correctly and consistently by all Internet-enabled applications, devices, and systems, regardless of script or length. This includes both new generic top-level domains (gTLDs) and Internationalized Domain Names (IDNs), and Email Address Internationalization (EAI). While it may be assumed that these work in the same manner as legacy TLDs, that is not the case and problems with acceptance are very common.

As an example, the goal is that an email such as web-test@ሁለንአቀፍ-ተቀባይነት-ሙከራ.com or تجربة-الويب@تجربة-القبول-الشامل.موريتانيا should have the same rate of acceptance as test@ua-test19.com.

This evaluation was commissioned by the Universal Acceptance Steering Group (UASG) under the Statement of Work (SOW) "Pilot Project for Large-Scale UA Remediation Campaign (EAI Support of Global Websites)".

Executive Summary

The Universal Acceptance Steering Group (UASG) anticipates a need to conduct large-scale remediation campaigns, develop a prototype process for such UA remediation campaigns, gain experience with the process, and learn lessons about the remediation process. The UASG chose to use the results from its EAI testing of the top 2,000 global websites in its remediation efforts.

The UASG025 report, "Global Evaluation of Websites for Acceptance of E-mail Addresses in 2019", was a follow-up to a similar test done in 2017. It was part of a broader initiative to further the community's understanding of the bottlenecks and key issues surrounding widespread compatibility of all domain names currently available. The 2017 and 2019 surveys tested the 1,000 most popular global websites and provided an informative global overview of UA-readiness.

The UASG027 report, "Country-Based Evaluation of Websites for Acceptance of Email Addresses in 2020", was a follow-up to testing done in 2017 and 2019. The testing done in UASG027 ascertained the acceptance rates of email addresses by websites in different countries. To do this, approximately 50 popular websites in 20 different countries were tested and compared for UA-readiness.

A total of 2,010 websites were tested after merging the websites tested in UASG025 and UASG027, and additional 380 global websites determined by using Alexa Top Global Websites API. The result of testing for EAI acceptance of these 2,010 websites is published in the UASG039 report.

This report, UASG046, focuses on the outcomes of remediation activities as well as the UA remediation process undertaken to make websites UA-ready. It included:

- Understanding the platform and technology stack used while developing the websites.
- Possible changes for making the website UA-ready.
- Suggesting the changes along with suitable pseudo code and use of requisite library.

UA Remediation Methodology

- Contact the website owners or webmasters to report the issue(s) identified.
- Since the campaigns did not result in positive responses, we started working on a strategy for more engagement with the website authorities and developers.
- Initiated one to one communication with the organizations who have attended awareness/remediation online workshops conducted by the EVARIS team and have shown interest in UA implementation. More than 26 online and offline awareness/remediation workshops and coding sessions were conducted which has resulted in responses from 56 organizations for possible UA implementation. Organizational-level talks were initiated and carried out for UA awareness and compliance. This also resulted in organizations showing interest in UA.
- With the remediation efforts of the EVARIS team, 22 out of 56 websites became UA-ready.

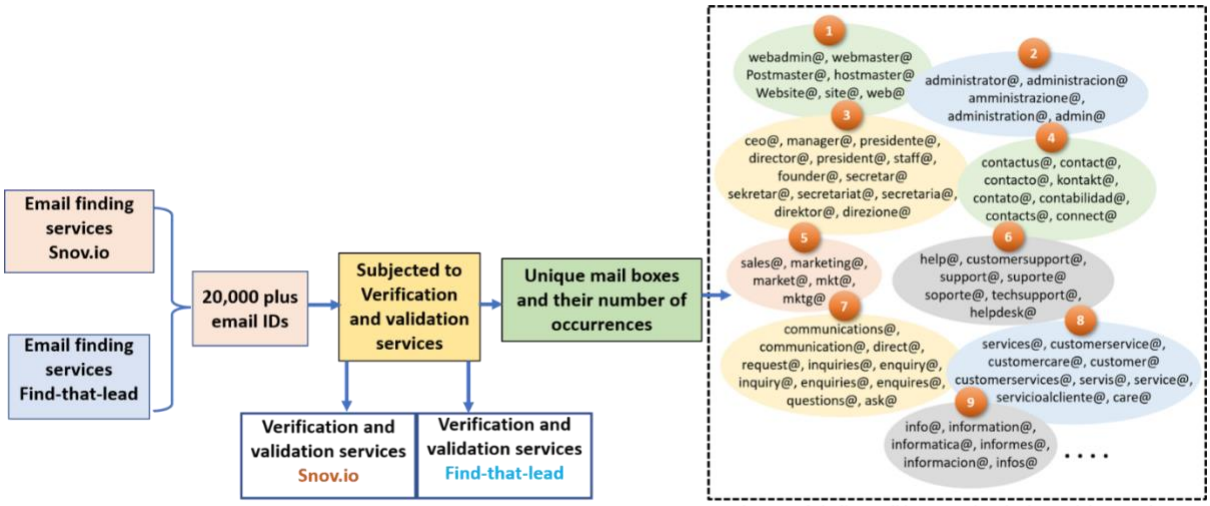
Contact Database Creation

It was a challenge to get the contact email IDs of the web authorities, webmasters, and web-admins to establish communication and provide possible remediation. For this report, email IDs were extracted from publicly available information. The same were validated to ensure positive delivery of the initial communication email.

Creation of a database of contact email IDs for 2,010 websites.

- Email IDs extraction
- Email verification

Email ID Extraction and Verification Activity



Clusters of similar mail boxes and assigning weightage. Also, assigning weightage within cluster.

The following methodology was used to get relevant contact email IDs of the 2,010 websites.

1. Two different third-party services were used to extract contact email IDs.
2. For the websites where emails were not extracted from the first service provider, the same were subjected to extraction through the second third-party provider.
3. Through this exercise, a total of 20,000-plus emails were extracted.
4. These emails were subject to a round of verification and validation with the tools provided by the service provider.
5. Found unique mailboxes along with their occurrences for these verified email IDs.
6. Made clusters of similar mailboxes and priority given to clusters as well as mailboxes within the cluster. If no email found from top priority clusters, then available singular email ID was allocated having priority 9999.
7. The main idea behind this was to pull out relevant emails to get positive responses. As an example, webmaster and customer-support were given more priority than press or abuse.
8. Minimum of one to maximum of three email IDs per website were identified.
9. Validated and relevant email IDs for 1,800-plus websites were obtained.
10. For around 790 websites where valid email IDs were not found through both the software, the same were subjected to manual extraction. These manually extracted email IDs were also included in the clusters.

11. For the remaining 180 websites, the following activities were undertaken:
    a. By filling the registration/contact form on the website to establish further email communication.
    b. Identifying the technical head/CEO/founder's name with Google search, Wikipedia, or on websites. Identifying LinkedIn profile and establishing communication through LinkedIn or direct email.
    c. Using online paid services for identifying the email IDs using the name of the technical head, CEO, or founder.
    d. Identifying the Twitter handle of the website and seeking the relevant person's contact email ID.
    e. Wherever available, online chat also led to getting contact details.
12. As per the GDPR policy and advice from the UASG, personal emails were removed from the contact database. Completed online available forms (e.g., registration, subscription, feedback, support) and others on the websites manually to communicate with them.

Domain Authentication

We leveraged the services of Mailchimp and snoiv.io for running and tracking the campaigns. For sending bulk emails, domain authentication was undertaken with the help of the UASG team, which also ensured that the emails reached recipient's inbox. Domain authentication helps maintain and grow an engaged audience.

Campaign Initiation and Status

As per the SOW, to touch base with the website authorities and web developers, various email campaigns were carried out with the extracted emails for 2,000-plus websites. There were hardly any responses from the organizations.

It was observed that the mail delivery rate was around 85% and open rate was 60%, which was encouraging. However, it was also observed that 25% were automatic responses out of the 60% open rate. The rest of the responses were related to ticket generation, advice to generate a ticket, or redirection.

Campaign -1 (with email -1 IDs) – Initiated on 24 April 2022 with first draft of email. Total 2,010 websites with 1,925 email IDs, and 1,758 total recipients with around 30% open rate and 10% click rate.

Campaign -2 (with email -2 & 3 IDs) - Initiated on 2 August 2022 with first draft of email. 1,791 emails were sent with a 53% open rate and 18% click rate.

Campaign - 3 (Indian language community) - Initiated on 8 August 2022 with the first draft of email. Total of 492 emails delivered with over 50% open rate. This was in addition to the 2010 websites, since responses from campaign-1 and campaign-2 were not that encouraging and we thought of touch basing known entities.

Campaign - 4 (with email-1 IDs) - Initiated on 2 September 2022 with second draft of email. Total of 1,040 emails sent with 39% open rate and 15% click rate.

Campaign - 5 (with email-1 IDs) - Initiated on 15 September 2022 for 140 mails re-campaigning.

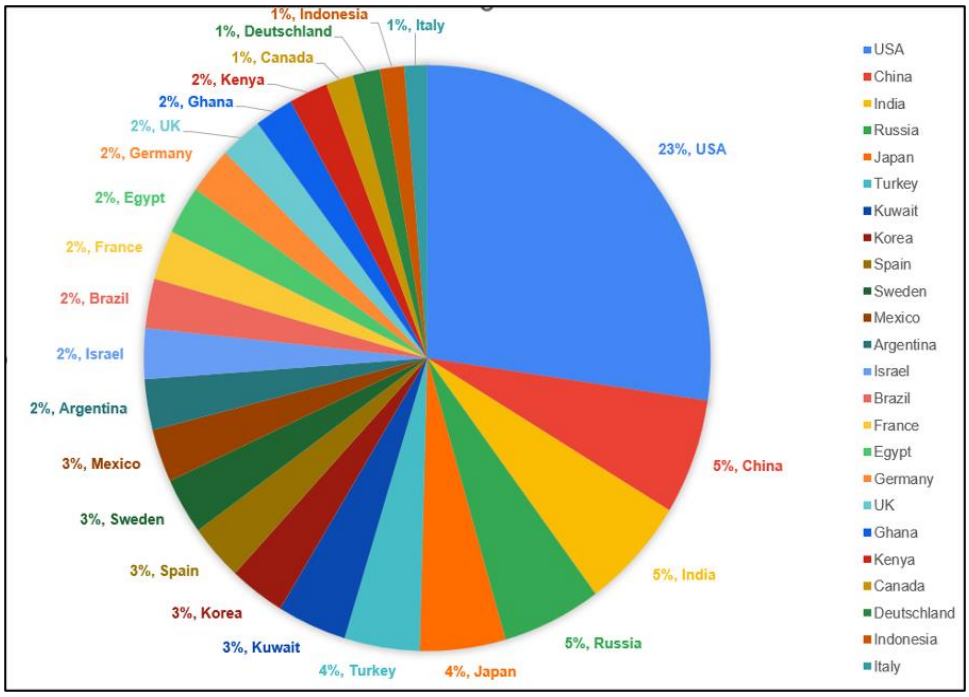
Campaign – Overall status:

- Delivery Rate: 85%
- Open Rate: 60%
- Link click rate: 35%
- Automated Responses: 25%
- Manual response: 0.4%

Country Distribution of Websites Under UASG039 Testing

We were able to find out country details for 1,796 of the 2,010 websites that were undertaken for testing. The table below shows the breakdown of the 1,796 websites by country.

SN	Country	Website count	Percentage out of 1,796
1	USA	420	23%
2	China	98	5%
3	India	95	5%
4	Russia	87	5%
5	Japan	75	4%
6	Turkey	66	4%
7	Kuwait	61	3%
8	Korea	49	3%
9	Spain	48	3%
10	Sweden	48	3%
11	Mexico	45	3%
12	Argentina	43	2%
13	Israel	43	2%
14	Brazil	42	2%
15	France	42	2%
16	Egypt	41	2%
17	Germany	40	2%
18	UK	37	2%
19	Ghana	34	2%
20	Kenya	34	2%
21	Canada	24	1%
22	Deutschland	24	1%
23	Indonesia	21	1%
24	Italy	20	1%



Campaign Status Automation

MailChimp provides a status of the campaign on login as well as through APIs. Different APIs provided by MailChimp for status of the email (i.e., delivered, read, bounced, clicked and others) were used and the final status was prepared in the form of .csv files which are being used for dashboard updates on a regular basis.

UA-Readiness Status of Websites

We conducted several UA awareness and remediation workshops. 56 organizations have shown interest for UA implementation.

	Number	Status
Unable to proceed further	19	− Website developers backed out from implementation without citing any reason
Unable to evaluate	7	− Problem while sending the form. − Email field not found. − Email field not available, however only Google form was available. − Provided Facebook URL instead of website.
Unable to implement fully	8	Contact form 7 plugin related issues
UA-ready	22	Result of remediation efforts undertaken by project team (refer to the Annexure - II for details)
TOTAL	56	

UA Implementation of 22 Websites

Out of 56 websites, 22 websites were made UA-ready, of which, 10 of the sites were developed in PHP and Laravel and 2 were developed in PHP. PHP-based remediation was recommended by the EVARIS team and implemented by the websites.

A further 5 websites were designed in HTML Bootstrap and 4 websites were in WordPress. For these HTML Bootstrap and WordPress sites, we recommended a JS-based solution. For the Technology Development for Indian Languages (TDIL) website, a JS-based solution was implemented.

For JavaScript-based solutions, we referred to the JavaScript library for conversion of Unicode to Punycode. Code snippets for the PHP-based solution is available in Annexure III: UA Compliance Sample Code for PHP.

HTML5's Email Input Type Observations

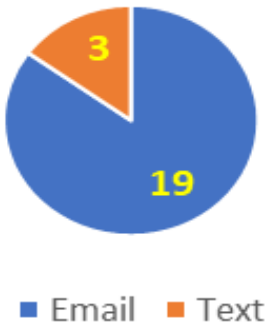
Out of the 22 websites, 19 websites used HTML input element attribute as email, while 3 websites used input element type attribute as text.

While designing forms in the websites for capturing user details including email IDs, the HTML5's "email" input type allows the collection of properly formatted ASCII-only email addresses.

The input type "email" makes web development easier since the ASCII email validation work is done by the browser. However, the browser's built-in email validation may vary slightly across different browsers and versions, and it is recommended to perform additional server-side validation to ensure the entered email is truly valid and associated with a real user account. The HTML5 input type "email" allows validation of ASCII emails only and does not support emails with Unicode characters in the input.

Since 19 websites have used input type "email" for collecting email from user, these websites failed to accept IDN emails, while 3 websites used "text" input type, which allows Unicode characters in the input, but does not validate the ASCII as well as EAI email addresses.

HTML Input Element Attribute



Overall Feedback from Implementors

Feedback was obtained from all of the 22 UA-ready websites about their experiences in implementing UA by sharing a standard template titled "Universal Acceptance Case Study Questions and Thought Starters", as provided by the UASG team.

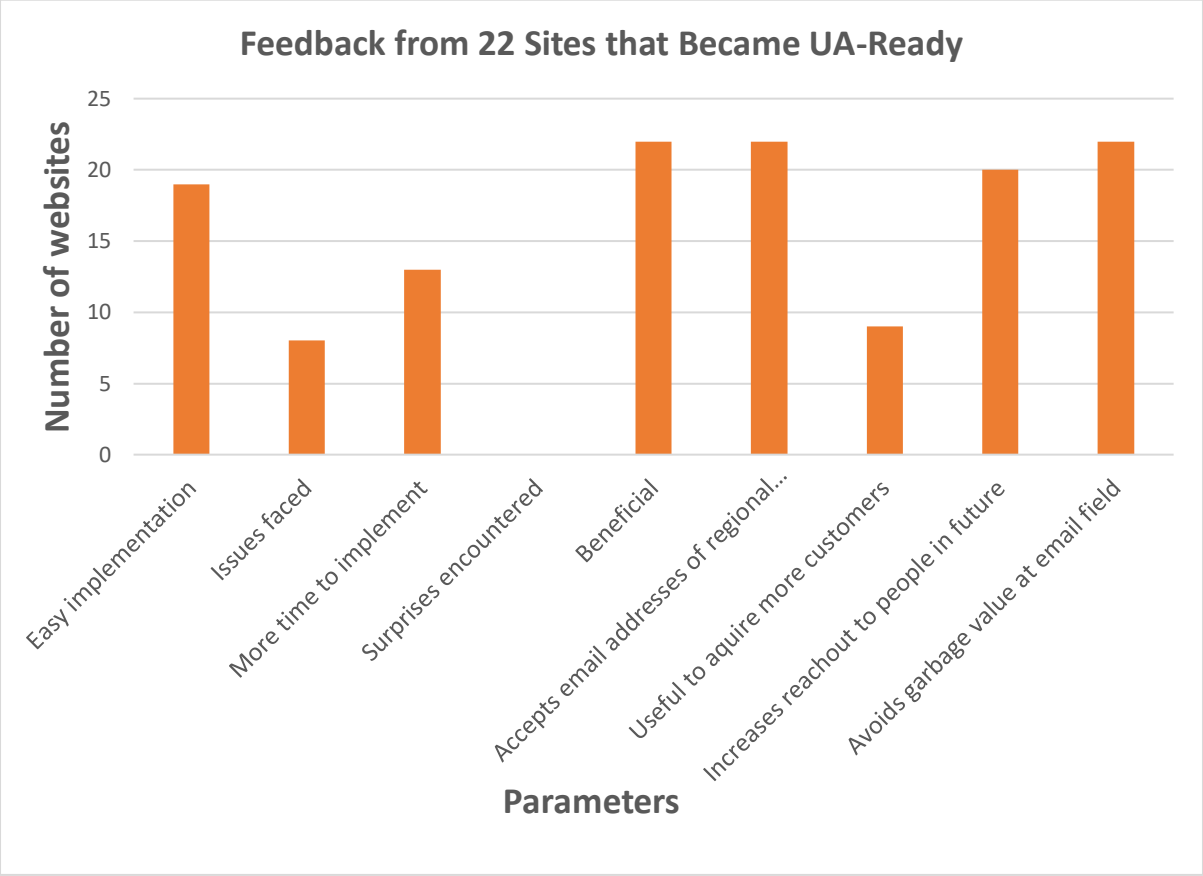
As can be seen from the feedback, all 22 websites felt that implementing UA is beneficial for their business growth. Also, after making websites UA-ready they now can allow EAI in email field and avoid "invalid email format."

None of the websites encountered any surprises while implementing UA. This was because of online coding sessions as well as continuous support provided by the project team. This indicates that rigorous assistance is required during UA implementation.

Around 8 websites faced issues (mostly related to obtaining permissions) while implementing, which resulted in taking more time to implement. However, they were made UA-ready with continuous follow-up and persuasion.

Around 20 websites felt that implementing UA will be useful for increasing outreach to people in the future, while 9 websites felt that it will be useful to acquire more customers.

In summary, continuous assistance is required technically with the implementation team, while continuing to persuade and follow up with decision makers.



Contact Form 7 Plugin-Related Issues

Out of 56 websites, 8 websites made use of a third-party WordPress plugin, namely "Contact Form -7". We tried to have a workaround for the same but did not yield the desired results. The following are our observations with implementing IDNA library in contact form 7 or other contact forms in WordPress.

- Plugin provider implements the server-side validation. In case we override the client-side validation using our UI API and validate, then also, after validation contact form plugin sends data to the server.
- After receiving the data at server side, WordPress plugin again validates the data at server side as per server-side implementation.
- Developer is not permitted to change the server-side code because of copyright issues, security issues where in source code is now allowed to change.

In cases where hosting companies rely on readymade/free plugins, they have two issues: one they do not have requisite knowledge of enhancing the plugin, and two, they are not allowed to make the changes because of copyright and security issues.

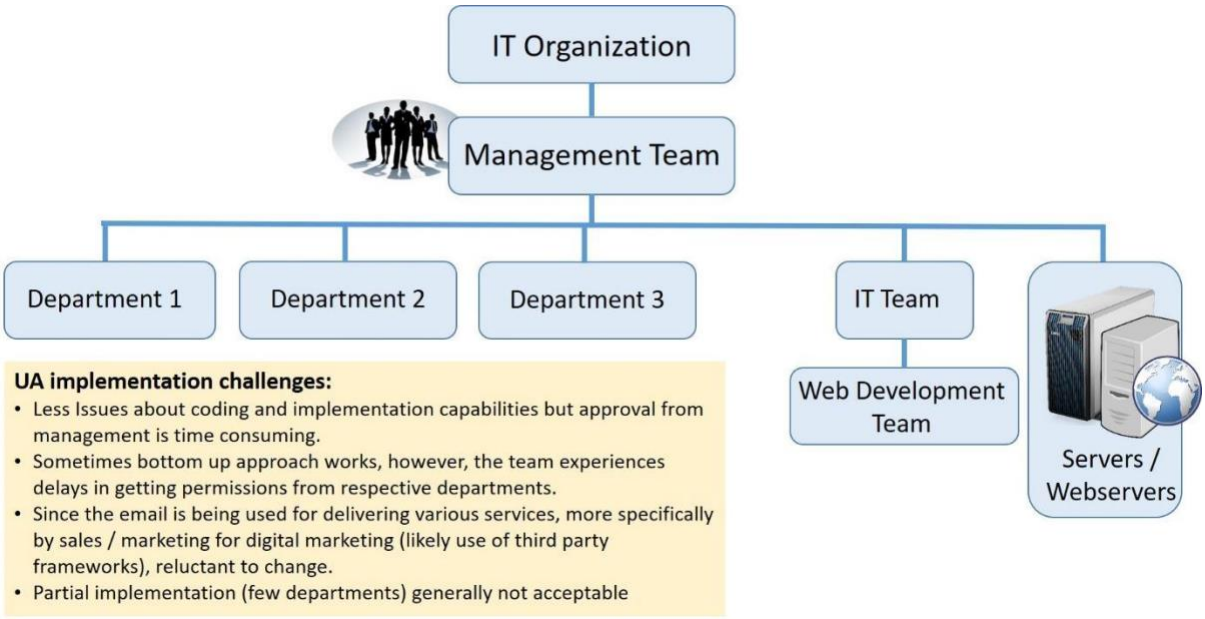
In such cases, we need to approach the plugin owner/developer for to educate them about the need for a UA compliance plugin.

Overall Challenges in Implementation

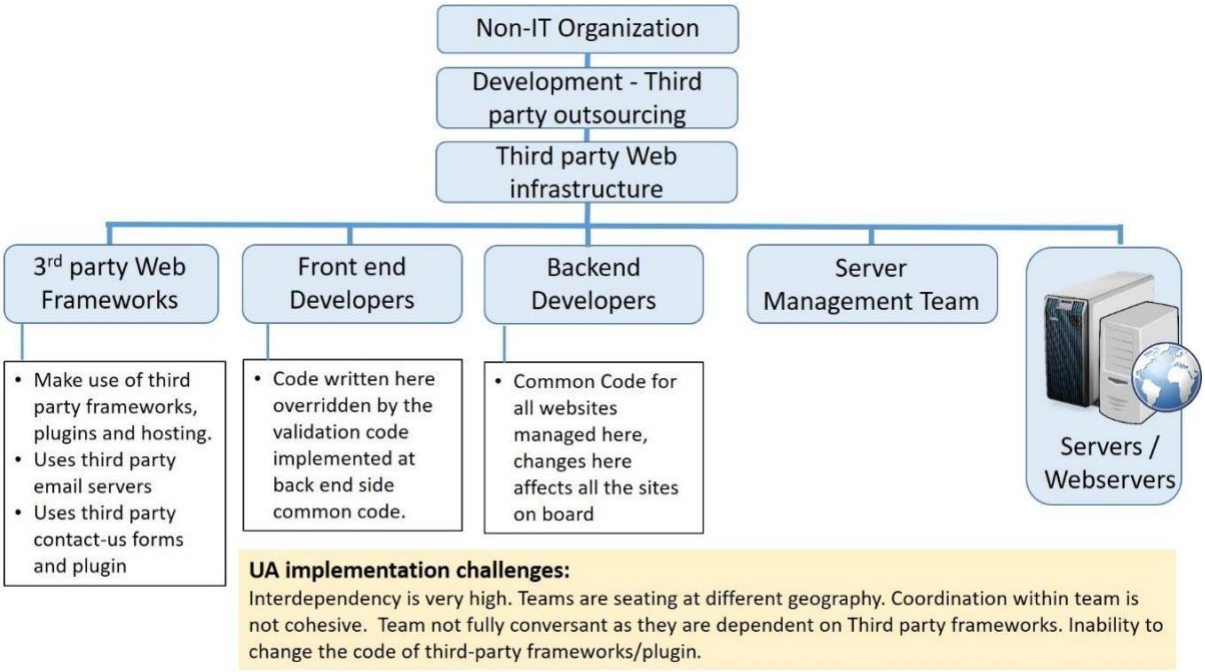
- It takes time to get permissions from the client or website owner for the changes to be done on their website.
- Since certain tasks are allocated to specific developer(s), availability of these developers is also a challenge. Takes time to get developers time to implement.
- Since developers do not have much knowledge about Universal Acceptance and EAI, it took several calls and meetings to explain it to them and also walk through the implementation details including pseudo code and procedure.
- It was time consuming to take necessary permissions and developers' availability (window period for implementation) since UA implementation was not a priority.
- Daily follow ups were required with respective parties through WhatsApp messages, email, and voice calls.

The following is the broad level of challenges faced in a typical IT organization and non-IT organization setup, and some of the IT organizations leverage third-party services for website development, hosting, and maintenance. It was observed that different departments within an organization leverage internal, third-party or both services, specifically, HR, finance, marketing, and pre-sales/post sales systems. In some cases, organizations that have a global presence use regional development teams, resources, and frameworks. In such cases unless the decision to implement UA is made globally, regional websites shy away from implementation.

A Typical IT Organization Setup



A Typical Non-IT Organization Setup



UA Implementation Challenges Faced with IT and Non-IT Organizations

1. At first instance, website owners, though they understood the importance of UA compliance, did not show interest in implementing it immediately. Rather they were reluctant to make the changes in existing working sites because of a variety if reasons:
    a. Do not perceive as an immediate need.
    b. Lack competent resources to handle the necessary implementation.
    c. Third-party services were used to develop and maintain the site.
    d. Making requisite changes would involve getting permissions from authorities, which can be challenging as team members involved may be in different cities or countries.
    e. Do not wish to make changes since multiple departments are leveraging the emails for offering services.

2. Since the website development and maintenance is outsourced (more specifically in non-IT organizations) they were unable to allocate the task and make developers available. Issues were also related to financial involvement for changes and priority of activity.

3. In the case of certain sites, implementations were required to be done at the frontend and backend. In certain cases, the frontend working team and backend working teams were different and only have respective permissions. Frontend developers/teams cannot make the changes in backend applications. In some cases, the teams are in different cities and getting permissions are tough or time consuming.

a. It was also observed for UA implementation that there is a need to involve both the frontend and backend (server-side) developers, since these were handled by different development teams.
b. Needed permissions from server team for backend implementation.
c. After getting the permissions and requisite access, then only we were able to start the implementation.

4. Some of the websites use third-party plugins/frameworks like 'WordPress Contact us form 7 plugins' and they rely mostly on the existing implementation of this form and are reluctant to make any changes in the form or they do not have in-depth understanding of the third-party plugin’s functionality.

5. Although the owner finds it good to have, they do not find it a priority job and they are unable to deploy a developer to complete it in one go. As per this exercise, they are making developers available when they find the time. It was time consuming to follow up with them. The overall time taken is very high in some cases.

6. Although UA implementation looks simple technically, and the pseudo code, IDN libraries, and requisite knowledge is readily available, they find it time consuming as implementation is required in many places (frontend, backend, database, job services, email service configuration, and compliance testing).

7. Most of the non-technical website development is done by outsourcing and they are mostly based on the plugins and third-party frameworks These people are reluctant to make any changes.

8. Another Issue faced was that there was a common validation method on the server side that was applicable to their multiple websites. In such cases, we were not allowed to make the changes for particular websites. The validations done at the frontend site were overruled by the backend validations.

Observations

In IDNA, the term Internationalized Domain Name (IDN) is a domain name that contains at least one A-label or U-label. Since 'A-label' and 'U-label' are specialist jargon, you could use the terms 'one portion which has characters beyond letters A-Z, digits 0-9, and the hyphen (a "U-label"), or a portion beginning with the special code 'xn--'. Refer RFC 5890, "Internationalized Domain Names for Applications (IDNA): Definitions and Document Framework", for details.

The levels of implementing IDNA for websites can vary depending on the specific circumstances. In general, implementing IDNA can be somewhat challenging and requires careful consideration of several technical and operational factors.

One of the main challenges of implementing IDNA is ensuring that the domain name is correctly encoded and decoded between the user's browser and the web server. This involves using the Punycode algorithm to convert non-ASCII characters to a compatible ASCII representation that can be used in the Domain Name System (DNS). In addition,

implementing IDNA also requires ensuring that the web server, DNS, and other relevant systems are configured to handle IDNs correctly.

Another potential challenge is ensuring that IDNA is supported by all relevant software and systems. While most modern browsers, programming languages, and operating systems have built-in support for IDNA, the use of older versions can result in compatibility issues for users. For example:

- PHP versions prior to 5.3.3 (release date 22 July 2010) do not have full IDNA support in their built-in functionality.
- Python 3.3 (release date 29 September 2012) and earlier versions have limited support for IDNA. The IDNA module was not included in the standard library until Python 3.4.
- Joomla started supporting IDNA with the release of Joomla 3.7.0 (release date 25 April 2017.) but had issues of IDNA convert. Joomla version 3.9 was released on 30 October 2018. Joomla does not have built-in support for IDNA 2008. The IDNA support in Joomla is based on the PHP programming language, specifically the "intl" extension.
- Drupal began supporting IDNA starting from Drupal 8.3.0 (released on 5 April 2017). Drupal relies on the underlying programming language, PHP, for IDNA support.
- WordPress has integrated IDNA support since version 4.6, released on 16 August 2016, WordPress uses the PHP programming language, and the IDNA support in PHP is provided by the "intl" extension.

Overall, implementing IDNA can require a significant amount of technical expertise and effort, particularly for large-scale websites or those with complex infrastructure. However, with careful planning and execution, it is possible to successfully implement IDNA for all sites.

Broad Steps for UA Compliance of Websites

To attain UA compliance for a website, the following is the list of actions to consider:

- Register and make functional IDNs for the website that support non-ASCII characters and is compatible with IDNA 2008 protocol. It's not mandatory to have an IDN for a website’s UA-readiness, but it ensures that the website is accessible to users from different language communities.
- Website to accept EAI emails in various forms such as subscriptions, feedback, registrations, and other user interactions.
- Validate EAI emails using IDNA libraries or similar tools to ensure proper handling and conversion of non-ASCII characters in email addresses. Verify the top-level domain (TLD) against the TLD list published by the Internet Assigned Numbers Authority (IANA). This additional validation ensures that the email address uses a valid TLD.
- Storing EAI emails in database: Modify your database schema and storage systems to accommodate EAI email addresses. Ensure that the database can handle and store non-ASCII characters properly.
- Ensure that the search and indexing mechanisms can handle and maintain the correct sort order of IDNs. May require specialized search algorithms or sorting techniques as per the Unicode collation rules.
- Update website's user interfaces to reflect the support for IDNs and EAI emails. Provide clear instructions on how users can enter and interact with non-ASCII characters in their email addresses.
- Ensure sending and receiving of EAI emails.

Learnings: Phased Approach for UA Compliance

Achieving full Universal Acceptance (UA) compliance, including the ability to validate, store, search, send, and receive Unicode email addresses, is a process that may require significant resources, planning, and phased implementation.

Maintaining an up-to-date TLD list can be challenging, which amounts to implementing a cron job to regularly fetch and update the TLD list which adds complexity to the validation process.

Implementing UA can be challenging, especially when websites are developed using third-party frameworks or tools. The third-party may not prioritize or easily accommodate such changes. Additionally, when EAI emails captured by the website are used by various departments within an organization, making changes for UA becomes more complex. Implementing UA requires coordination and buy-in from various departments within an organization that utilize the captured email addresses for providing various services to their customers.

Incremental implementation, instead of making widespread changes across the entire organization all at once, is a phased approach that can be considered. By adopting a strategic and phased approach, engaging stakeholders, and leveraging available resources and communities, it is possible to make progress towards UA while mitigating potential barriers.

Merely accepting Unicode email addresses does not suffice, it is necessary to understand and address the limitations and challenges in the email server infrastructure for successful delivery of IDN emails.

Developers are looking for a simple solution to accept Unicode characters in email addresses without making significant changes or undergoing a security audit of website.

Hence, starting with accepting Unicode email addresses using Regex as a first step is a practical approach, allowing gradual progress towards full UA compliance. Use of Regex allows accommodation of a broader range of email addresses without immediately undertaking all the complexities of full UA compliance. However, please note that email address validation is a complex task, and it is difficult to achieve 100% accuracy with a regular expression alone. Email address formats can vary, and there are some edge cases that may not be covered by this basic pattern. It is recommended to combine regular expressions with server-side validation and additional checks to ensure thorough email address validation.

By prioritizing, allocating resources, and iterating on the implementation, websites can evolve and become fully UA compliant over time. However, it's necessary to communicate the limitations and potential future steps, such as transitioning to IDNA libraries for full IDN email support to the users.

As the website evolves, subsequent phases can focus on searching, sending-receiving Unicode email addresses. Each phase can build upon the previous one, gradually improving the website's UA compliance over time. This iterative approach allows for thorough testing, addressing challenges, and fine-tuning the implementation.

Various Libraries and Overall IDN Implementation Flow

- There are many libraries available for IDN to ASCII converters.
- IDNA libraries are available for various languages and frameworks. Developers face challenges in using the one suitable to them.
- The conversion and API implementation of each library varies. Certain libraries convert the complete string, which may include subdomain, domain, and TLD, while others only convert a single label. Some libraries require the use of specific flags, such as UTS46 and Transit, for which developers must invest considerable effort to comprehend their importance and purpose.
- Additionally, these libraries anticipate that the input Unicode string is in an IDN-compatible format. Each library will either convert the Unicode input string to a Punycode (ASCII) string or produce an error message regarding the incompatible IDN input.
- The actual implementation of UA is not limited to using the IDNA library, but to follow entire process from taking input from text box, splitting into subdomains and TLDs. Implementation of TLD verification, implementing required corn jobs, splitting every label, and validating as proper IDN composition. This sometime poses challenges to developers and overall understanding and pseudo code is required.
- Implementors are encouraged to visit https://github.com/icann/ua-code-samples repository which contains code samples showing how to use certain programming languages libraries correctly to support internationalization, specifically EAI and IDNs (IDNA2008).
- **PHP Universal Compliance code samples:** Implementors are encouraged to visit https://github.com/icann/ua-code-samples/tree/master/readiness-sample-code/php which contains code samples to provide guidelines for UA for PHP libraries.

Annexure - I

The following is the list of website URLs that have shown interest in UA implementation.

No.	Website	Email Input Field Type	Technology
1	https://icrmsoftware.com	email	WordPress
2	https://icrmmarketing.com	email	WordPress
3	https://b2bmedium.com	email	HTML Bootstrap
4	https://micromatix.net	email	PHP
5	https://executive81.com	email	PHP
6	https://sweekcollection.com	email	PHP
7	https://skorganic.co.in	email	
8	https://markspacemedia.com/	email	PHP
9	https://kanishkathakur.com/	email	PHP
10	https://jaidadaengineering.in/	email	PHP
11	https://layouts.icrmsoftware.com/	email	PHP
12	http://hawkwings.in/	email	PHP
13	https://cosmeticonline.in/	email	PHP
14	https://donspectacularis.com/		
15	https://dordressonrent.com/	email	PHP
16	https://mebazzar.com/	email	PHP
17	https://cartrefs.com/	email	PHP
18	https://pcianalytics.in/		PHP
19	https://www.bipinpharmaequipment.com/	email	WordPress
20	https://www.indiatechpharmaexporters.com/	email	WordPress
21	https://sparkweld.net/	email	HTML Bootstrap
22	https://www.technosearchprocess.com/	email	WordPress
23	https://www.inditechsystems.com/	email	WordPress
24	https://www.mackauraadrugs.com/	text	HTML Bootstrap
25	https://www.pnsafetyind.com/	email	HTML Bootstrap
26	https://www.vibroscreens.net/	email	HTML Bootstrap
27	https://srlabinstruments.com/	email	WordPress
28	https://www.vmecranes.in/	email	WordPress
29	https://rishikeshexports.in/	email	WordPress
30	https://www.labtopinstruments.com/	email	WordPress
31	https://www.smartmark.co.in/	email	WordPress
32	https://www.thermopac.in/	email	WordPress
33	https://www.manasvihitech.com/	email	HTML Bootstrap
34	www.bbllogistics.com/	email	WordPress
35	https://www.careersin.in/	email	WordPress
36	https://www.genotekbio.com/	email	WordPress
37	https://kamadhenuelevators.com/	email	WordPress
38	https://liscio.in/	email	WordPress
39	https://www.marlinemarine.com/	email	WordPress
40	https://www.fireballsupplier.com/	email	WordPress
41	https://www.vaastuurjja.com/	email	WordPress
42	https://www.ficci-ilia.in/	text	PHP
43	www.perfectcomputereducation.com	-	HTML CSS
44	www.cncentrecdac.in	text	WordPress
45	https://dot-tech.in	email	PHP
46	https://cdaccomputerclass.business.site/	email	

47	https://markeducation.info	email	WordPress
48	https://facebook.com/cdac.dahod.scope	-	-
49	www.bypt.in	email	WordPress
50	https://cdaclawgardenahmedabad.com/	email	HTML Bootstrap
51	https://tdil-dc.in	email	PHP
52	https://www.arrowcomputers.in/	email	PHP
53	https://cdacgujarat.com/	email	PHP
54	https://productsearchinfotech.com/	email	WordPress
55	https://globaluaday.in/	Text	WordPress
56	https://evarissystems.com	Text	WordPress

Annexure - II

UA-Ready Websites

SN	Website	Email Input Field Type prior to UA implementation	Website developed with	Implementation for UA compliance	Changes at frontend and backend for making website UA ready	"Yes" - stored as U-label in database "No" - Database field not updated to store UTF-8 character	Implemented cron job for validating TLD against IANA TLD list	Mail sending utility "Y" - supports EAI, "N" -does not support EAI	UA readiness date
1	https://cartrefs.com/	email	PHP	PHP Based	• Backend APIs to validate domain part and TLD. • Frontend Script to consume these API's. • Use PHP function for converting domain part Unicode to Punycode.	Y	N	• N	8th February 2023
2	https://kanishkathakur.com/	email	PHP	PHP Based	• Backend APIs to validate domain part and TLD.	Y	N	• N	8th February 2023

SN	Website	Email Input Field Type prior to UA implementation	Website developed with	Implementation for UA compliance	Changes at frontend and backend for making website UA ready	"Yes" - stored as U-label in database "No" - Database field not updated to store UTF-8 character	Implemented cron job for validating TLD against IANA TLD list	Mail sending utility "Y" - supports EAI, "N" -does not support EAI	UA readiness date
					• Frontend Script to consume these API's. • Use PHP function for converting domain part Unicode to Punycode.				
3	https://sweekcollection.com	email	PHP	PHP based Solution	• Backend APIs to validate domain part and TLD. • Frontend Script to consume these APIs. • Use PHP function for converting domain part Unicode to Punycode.	Y	N	Y • Received email on ASCII but not on EAI email address upon form submission.	16th Dec 2022

SN	Website	Email Input Field Type prior to UA implementation	Website developed with	Implementation for UA compliance	Changes at frontend and backend for making website UA ready	"Yes" - stored as U-label in database "No" - Database field not updated to store UTF-8 character	Implemented cron job for validating TLD against IANA TLD list	Mail sending utility "Y" - supports EAI, "N" -does not support EAI	UA readiness date
4	https://markspacemedia.com/	email	PHP	PHP based Solution	• Backend APIs to validate domain part and TLD. • Frontend Script to consume these APIs. • Use PHP function for converting domain part Unicode to Punycode.	Y	N	N • No acknowledgment mail setup.	16th Dec 2022
5	https://jaidadaengineering.in/	email	PHP	PHP based Solution	• Backend APIs to validate domain part and TLD. • Frontend Script to consume these APIs. • Use PHP function for converting	Y	N	N • No acknowledgment mail setup.	16th Dec 2022

SN	Website	Email Input Field Type prior to UA implementation	Website developed with	Implementation for UA compliance	Changes at frontend and backend for making website UA ready	"Yes" - stored as U-label in database "No" - Database field not updated to store UTF-8 character	Implemented cron job for validating TLD against IANA TLD list	Mail sending utility "Y" - supports EAI, "N" -does not support EAI	UA readiness date
					domain part Unicode to Punycode.				
6	https://layouts.icrmsoftware.com/	email	PHP	PHP based Solution	• Backend APIs to validate domain part and TLD. • Frontend Script to consume these APIs. • Use PHP function for converting domain part Unicode to Punycode.	Y	N	N • No mail received on ASCII and EAI email addresses upon form submission.	16th Dec 2022
7	http://hawkwings.in/	email	PHP	PHP based Solution	• Backend APIs to validate domain part and TLD. • Frontend Script to consume these APIs.	Y	N	Y • Received email on ASCII but not on EAI email address upon form submission.	16th Dec 2022

SN	Website	Email Input Field Type prior to UA implementation	Website developed with	Implementation for UA compliance	Changes at frontend and backend for making website UA ready	"Yes" - stored as U-label in database "No" - Database field not updated to store UTF-8 character	Implemented cron job for validating TLD against IANA TLD list	Mail sending utility "Y" - supports EAI, "N" -does not support EAI	UA readiness date
					• Use PHP function for converting domain part Unicode to Punycode.				
8	https://cosmeticonline.in/	email	PHP	PHP based Solution	• Backend APIs to validate domain part and TLD. • Frontend Script to consume these APIs. • Use PHP function for converting domain part Unicode to Punycode.	Y	N	Y • Received email on ASCII but not on EAI email address upon form submission.	24th Dec 2022
9	https://dordressonrent.com/	email	PHP	PHP based Solution	• Backend APIs to validate domain part and TLD.	Y	N	N • No mail received on ASCII and EAI email	24th Dec 2022

SN	Website	Email Input Field Type prior to UA implementation	Website developed with	Implementation for UA compliance	Changes at frontend and backend for making website UA ready	"Yes" - stored as U-label in database "No" - Database field not updated to store UTF-8 character	Implemented cron job for validating TLD against IANA TLD list	Mail sending utility "Y" - supports EAI, "N" -does not support EAI	UA readiness date
					• Frontend Script to consume these APIs. • Use PHP function for converting domain part Unicode to Punycode.			addresses upon form submission	
10	https://mebazzar.com/	email	PHP	PHP based Solution	• Backend APIs to validate domain part and TLD. • Frontend Script to consume these APIs. • Use PHP function for converting domain part Unicode to Punycode.	Y	N	N • No mail received on ASCII and EAI email addresses upon form submission.	24th Dec 2022

SN	Website	Email Input Field Type prior to UA implementation	Website developed with	Implementation for UA compliance	Changes at frontend and backend for making website UA ready	"Yes" - stored as U-label in database "No" - Database field not updated to store UTF-8 character	Implemented cron job for validating TLD against IANA TLD list	Mail sending utility "Y" - supports EAI, "N" -does not support EAI	UA readiness date
11	https://sparkweld.net/	email	HTML Bootstrap	JS based Solution	• APIs at backend for validating domain and TLD part. • JavaScript library used to convert Unicode to Punycode. • Write script at frontend to consume these APIs.	N	N	Y • Mail received on ASCII email and EAI addresses	8th Dec 2022
12	https://www.mackauraadrugs.com/	text	HTML Bootstrap	JS based Solution	• Server-side APIs for validating the domain and TLD part. • Script to consume these APIs and Regex pattern which validate bare	N	N	Y • Mail received on ASCII email and EAI addresses	6th Dec 2022

SN	Website	Email Input Field Type prior to UA implementation	Website developed with	Implementation for UA compliance	Changes at frontend and backend for making website UA ready	"Yes" - stored as U-label in database "No" - Database field not updated to store UTF-8 character	Implemented cron job for validating TLD against IANA TLD list	Mail sending utility "Y" - supports EAI, "N" -does not support EAI	UA readiness date
					minimum structure of Email Address. ● JavaScript library used for non- ascii domain part.				
13	https://www.pnsafetyind.com/	email	HTML Bootstrap	JS based Solution	● APIs at backend for validating domain and TLD part. ● JavaScript library used to convert Unicode to Punycode. ● Write script at frontend to consume these APIs.	N	N	Y ● Mail received on ASCII email and EAI addresses	6th Dec 2022
14	https://www.vibroscreens.net/	email	HTML Bootstrap	JS based Solution	● Server-side APIs for validating the	N	N	N ● No mail received on	8th Dec 2022

SN	Website	Email Input Field Type prior to UA implementation	Website developed with	Implementation for UA compliance	Changes at frontend and backend for making website UA ready	"Yes" - stored as U-label in database "No" - Database field not updated to store UTF-8 character	Implemented cron job for validating TLD against IANA TLD list	Mail sending utility "Y" - supports EAI, "N" -does not support EAI	UA readiness date
					domain and TLD part. ● Script to consume these APIs and Regex pattern which validate bare minimum structure of Email Address. ● JavaScript library used for non- ascii domain part.			ASCII and EAI email addresses upon form submission.	
15	https://www.manasvihitech.com/	email	HTML Bootstrap	JS based Solution	● Backend APIs to validate domain part and TLD. ● Frontend Script to consume these APIs.	N	N	Y ● Mail received on ASCII email and EAI addresses	6th Dec 2022

SN	Website	Email Input Field Type prior to UA implementation	Website developed with	Implementation for UA compliance	Changes at frontend and backend for making website UA ready	"Yes" - stored as U-label in database "No" - Database field not updated to store UTF-8 character	Implemented cron job for validating TLD against IANA TLD list	Mail sending utility "Y" - supports EAI, "N" -does not support EAI	UA readiness date
					• JS based library 'idna-uts46-hx' used to convert Unicode to respective Punycode.				
16	https://www.ficci-ilia.in/	text	PHP	PHP based Solution	• Regex pattern shared by us. • Server-side APIs to validate domain part and TLD. • Frontend Script to consume these APIs. • Use PHP function for converting domain part Unicode to Punycode.		Y	N	29th Nov 2022

SN	Website	Email Input Field Type prior to UA implementation	Website developed with	Implementation for UA compliance	Changes at frontend and backend for making website UA ready	"Yes" - stored as U-label in database "No" - Database field not updated to store UTF-8 character	Implemented cron job for validating TLD against IANA TLD list	Mail sending utility "Y" - supports EAI, "N" -does not support EAI	UA readiness date
17	www.cncentrecdac.in	text	WordPress	JS based Solution	• Backend APIs to validate domain part and TLD. • Frontend Script to consume these APIs. • JS based library used to convert Unicode to respective Punycode.	N	Y	N	26th Dec 2022
18	https://dot-tech.in	email	PHP	PHP based Solution	• Regex pattern shared by us. • Server-side APIs to validate domain part and TLD. • Frontend Script to consume these APIs.	Y	Y	N • No functionality.	23rd Dec 2022

SN	Website	Email Input Field Type prior to UA implementation	Website developed with	Implementation for UA compliance	Changes at frontend and backend for making website UA ready	"Yes" - stored as U-label in database "No" - Database field not updated to store UTF-8 character	Implemented cron job for validating TLD against IANA TLD list	Mail sending utility "Y" - supports EAI, "N" -does not support EAI	UA readiness date
					• Use PHP function for converting domain part Unicode to Punycode.				
19	https://productsearchinfotech.com/	email	WordPress	JS based Solution	• Backend APIs to validate domain part and TLD. • Frontend Script to consume these APIs. • JS based library 'idna-uts46-hx' used to convert Unicode to respective Punycode.	N	N	N • No mail received on ASCII and EAI email addresses upon form submission.	26th Dec 2022
20	https://tdil-dc.in/	email		JS based Solution				Y • Mail was received on ASCII but not	28th September 2022

SN	Website	Email Input Field Type prior to UA implementation	Website developed with	Implementation for UA compliance	Changes at frontend and backend for making website UA ready	"Yes" - stored as U-label in database "No" - Database field not updated to store UTF-8 character	Implemented cron job for validating TLD against IANA TLD list	Mail sending utility "Y" - supports EAI, "N" -does not support EAI	UA readiness date
								on EAI email address	
21	https://globaluaday.in/	Text	WordPress	JS based Solution		Y	Y	Y	25th March 2023
22	evarissystems.com	Text	WordPress	JA based Solution		Y	Y	Y	8th Sept 2022

Annexure – III

Making PHP-Based Websites UA Compliant

The following is information on how to make PHP-based websites UA compliant.
The code provided here is compatible for PHP Version: 5 >= 5.3.0, PHP 7, PHP 8.

Acceptance of EAI in email field in website form

The following steps and code snippets need to be implemented at the frontend and backend. Service also needs to be implemented for validation of top-level domains against the official database (https://data.iana.org/TLD/tlds-alpha-by-domain.txt) from iana.org.

EAI acceptance in an email field can be achieved using the following steps:

1. **Frontend allowing Unicode characters in the email field.**

   As stated above, HTML5 input type “email”, allows validation of ASCII emails only and does not support emails having Unicode characters in the input, hence HTML input element attribute “text” to be used instead.

   Once we change the attribute to “text”, it allows to enter any arbitrary Unicode characters in the input field. To make this input field to accept valid format of email regex pattern can be used. The regex pattern can be written in JavaScript. The regex can be used for any other language, with certain changes because of some variations in the available features and methods.

Regex pattern:

```
/^(([^<>()[\]\.,;:\s@\"]+(\.[^<>()[\]\.,;:\s@\"]+)*)|(\".+\"))@(([^<>()[\]\.,;:\s@\"]+\.)+[^<>()[\]\.,;:\s@\"]{2,})$/i
```

This regular expression checks for the following basic email address requirements:

- The local part (before the @ symbol) can contain any characters except for specific special characters (<, >, (,), [,], ,, ;, :, \s, @, and "). Quoted local parts are also allowed.
- The domain part (after the @ symbol) must have at least one dot and be followed by at least two characters (TLD).
- The regular expression is case-insensitive (i flag at the end).

Please note that email address validation is a complex task, and it is difficult to achieve 100% accuracy with a regular expression alone. Email address formats can vary, and there are some edge cases that may not be covered by this basic pattern. It is recommended to combine regular expressions with server-side validation and additional checks to ensure thorough email address validation.

2. **Code for converting domain part Unicode to Punycode (PHP)**

- Split the email into local part and domain part (local part: left part of the @, and domain part: right of the @ symbol)
- Convert domain part into respective Punycode (A-Label) from Unicode (U-Label).
- Use the PHP built in function idn_to_ascii() to convert to Punycode. There are additional functions available in PHP. May like to refer the documentation is available online. PHP: idn_to_ascii – Manual
- The function idn_to_ascii() returns the string encoded in ASCII-compatible form (Punycode) or returns false on failure.
- This code needed to be run as backend API and should be invoked on the submit button.
- **Refer the following sample code for converting Unicode string to Punycode.**

```
// IDN to ASCII
function convert_i2a($domain_part) {
	$ascii_result = idn_to_ascii($domain_part);
	$response = array();
	if($ascii_result) {
		$response[“result”] = “1”;
		$response[“status”] = “âœ”ï¸ Domain part is valid : ” . $domain_part;
	}
	else {
			$response[“result”] = “0”;
			$response[“status”] = “â Œ Error in domain part : ” . $domain_part;
	}
	 return $response;
}
```

3. **Top-level domain (TLD) validation**

- The TLD this the end part of the email address for example in the domain name "evaris.co.in" [.in] is the TLD part.
- TLD against the official database made available on https://data.iana.org/TLD/tlds-alpha-by-domain.txt by iana.org which is updated daily.
- This code needs to run as cron job which will download the latest TLD.txt onto the website’s server daily.

The following is the code that downloads a TLD file and stores it in the folder:

```
<?php
			$file_url_TLD = “https://data.iana.org/TLD/tlds-alpha-by-domain.txt”;
			$file_name_TLD = “TLD.txt”;
			$file_contents_TLD = file_get_contents($file_url_TLD);
			$write_file_TLD = file_put_contents($file_name_TLD,
$file_contents_TLD);
			date_default_timezone_set(‘Asia/Kolkata’);
			$date_time = date(‘d-m-y h:i:s’);
```

```
                    $flie_log_TLD = fopen('./downloadTLDFileLog.txt', 'a');
                    $string_log = "";
                    if($write_file_TLD) {
                        $string_log = $date_time . " Success : TLD File was downloaded.\n";
                        echo $string_log;
                        fwrite($flie_log_TLD, $string_log) ;
                    }
                    else {
                        unlink('TLD.txt');
                        $string_log = $date_time . " Failure: TLD File downloading was failed.
\n";
                        echo $string_log;
                        fwrite($flie_log_TLD, $string_log);
                    }
                    fclose($flie_log_TLD);
 ?>
```

After downloading and storing the TLD.txt file in website server, the TLD part of the domain name is checked against this file. If present then the TLD validation is success, else not.

For IDN TLDs, pass the TLD to the **idn_to_ascii()** function for conversion to Punycode and then validate against the downloaded **TLD.txt** file.

This code needs to run as backend API and invoked on submit button.

The code for checking the TLD is given below:

```
//Check TLD
function check_TLD($domain_part) {
    $response = array();
    $omain_part_array = explode(".", trim($domain_part));
    $TLD_string = end($domain_part_array);
    $idn_validated_TLD = idn_to_ascii($TLD_string);
    if($idn_validated_TLD) {
        $TLD_file = "./TLD.txt";
        $TLD_file_contents = file_get_contents($TLD_file);
        $TLD_array = explode("\n", $TLD_file_contents);
        array_shift($TLD_array);
        array_pop($TLD_array);
        $tld = strtoupper(trim($idn_validated_TLD));
        $result = in_array($tld, $TLD_array);
        if($result) {
            $response["result"] = "1";
            $response["status"] = "âœ"ï¸ Valid TLD : " . $TLD_string;
        }
        else {
            $response["result"] = "0";
```

```
                $response[“status”] = “â Œ Invalid TLD : ” . $TLD_string;
            }
        }
        else {
            $response[“result”] = “0”;
            $response[“status”] = “â Œ Error in TLD.” . $TLD_string;
        }
        return $response;
}
```

Note: To use regex-free code, visit https://github.com/icann/ua-code-samples.
For PHP Universal Compliance code samples, visit https://github.com/icann/ua-code-samples/tree/master/readiness-sample-code/php.