Universal Acceptance

Email Address Internationalization (EAI): Evaluation of Major Email Software and Services

Between

Universal Acceptance Steering Group (UASG)

&

Catalyst.Net Limited

September 2018
Version 1.0

UASG021A

2018-09-19

TABLE OF CONTENTS

1 Background

The Universal Acceptance Steering Group (UASG) is a community-based organisation working toward the goal of Universal Acceptance (UA), the idea that all domain names and email addresses should be treated correctly and consistently by Internet-enabled applications, devices, and systems. Specifically, this requirement includes new generic top-level domains (gTLDs), internationalized domain names (IDNs), and internationalized email addresses, which must be accepted, validated, stored, processed, and displayed as well as their traditional ASCII-based counterparts.

Because email messages constitute a significant part of online communications, it's important that email software and email service providers meet these requirements to achieve Universal Acceptance. Email Address Internationalization (EAI) is the protocol that allows email addresses that incorporate IDN or Unicode components to function correctly within the email software ecosystem. The purpose of this project is to evaluate the existing EAI capabilities of that ecosystem, in order to measure its “EAI Readiness”.

1.1 Project Scope

To achieve this evaluation, a diverse set of components from the email software ecosystem will be analyzed in order to determine the degree to which each component is “EAI Ready”, that is, whether it currently handles EAI messages, addresses, and domain names correctly.

The first phase of the project, Discovery and Analysis, will determine how the evaluation should proceed. In particular, it will:

1. Specify the tests and expected results to use for the analysis,
2. Specify the criteria to use when selecting email software and services to analyze, and
3. Estimate the effort required to perform a pilot evaluation of a select few software and service offerings chosen according to these criteria.

The second phase of the project will execute a pilot evaluation, in order to validate and refine the criteria and procedures defined in this report.

The third phase will execute a more robust evaluation, analyzing a much larger number of components.

It is important to understand that these evaluations do not constitute any sort of approval or certification of EAI Readiness, regardless of results. The information gathered by this project is intended only as a snapshot of current capabilities, is not exhaustive in its coverage of EAI-related behaviors, and will of course fall out of date as the email software ecosystem develops over time.

1.2 About This Document

This document describes the Discovery and Analysis phase of the EAI Evaluation project. It provides project background, software selection criteria, a set of candidate software components for analysis, and a description of the procedure to use when executing the evaluation. Accompanying this

document is a spreadsheet[1] detailing the individual test cases and test data to use for the analysis. Finally, a pilot evaluation is proposed, including an estimate of the effort involved. It's expected that the procedures described in this document will evolve as experience is gained throughout the project.

1.3 Further Background

An introduction to UA is provided by UASG 005 (Universal Acceptance Quick Guide).

An introduction to EAI is provided by UASG 014 (Quick Guide to EAI) and UASG 012 (EAI: A Technical Overview).

A list of RFCs relevant to UA and EAI can be found in UASG 006 (Relevant RFCs to Universal Acceptance).

2 Evaluation Procedure

In its entirety, the EAI Evaluation project will measure the behavior of many different software applications and services. Here, we describe the procedure for analyzing a single one of these in terms of test data, test cases, and preparation of results.

2.1 Notification of Service Provider

If the owning entity or primary development group of the software or service under test can be determined, that party should be notified that an analysis of the component will be taking place. They should also be invited to participate to whatever degree possible, since expert knowledge and guidance regarding the software's intended behavior will likely make the analysis more accurate and efficient.

2.2 Test Data

Test data has been drawn from the set of Use Cases defined by UASG 004 (Use Cases for UA Readiness Evaluation), which provides domain names and email addresses for use in testing Universal Acceptance. The Use Cases have subsequently been organized into three types as denoted by the labels <Email>, <Domain>, and <Unicode>. Each of these contain examples of ASCII, Unicode, and Right-to-Left Unicode text. These values are to be used as inputs to the software under test, with the expected behavior documented within each test case.

2.3 Test Cases

The evaluation's test cases are detailed in the attached document. They are divided into categories as described by . Each of these categories encompasses a set of responsibilities relevant to that type of software. Within these categories, each individual test case consists of the following information:

1. A unique identifier for the test case.

[1] https://uasg.tech/wp-content/uploads/2018/09/UASG021A.xlsx

2. A summary of the behavior to be tested.
3. A description of the test, including any notable prerequisites or exclusions.
4. The action required to execute the test.
5. The expected result.
6. A classification of the tested behavior as “Required” or “Advisory”. Refer to .
7. An indication of the EAI Phase to which the tested behavior belongs. Refer to .
8. Any external references relevant to the test case.

Many of the test cases judge whether the component under test handles email addresses, domain names, or Unicode text in a particular way. For these, the test case has been listed once rather than repeated for each Use Case specified by UASG 004, and its action includes a reference to the type of test data for which the test should be repeated, one of <Email>, <Domain>, or <Unicode>. For such tests, a pass result indicates that *all* relevant Use Cases were handled correctly, while a fail result indicates that at least one Use Case was handled incorrectly. The detailed execution report should include a breakdown of results for each individual Use Case, as well as information about the cause of a fail result where applicable.

2.3.1 Test Case Classifications

Each test case includes an indication of whether the tested behavior is *Required* or *Advisory*. A component must pass all required test cases to be considered EAI-ready, while failing results for advisory test cases are permitted. Regardless of classification, all results should should be reported in the test execution report, with failures for required test cases indicated as errors and failures for advisory test cases indicated as warnings.

2.3.2 EAI Phases

There are in fact two phases of EAI Readiness. Specifically:

- Phase 1 EAI Ready software is capable of *sending to* and *receiving from* EAI addresses, while
- Phase 2 EAI Ready software is capable of *hosting* an EAI address.

Each test case indicates whether the tested behavior is more closely associated with Phase 1 or Phase 2 EAI Readiness. It’s expected that software will be Phase 1 Ready before it is Phase 2 Ready. All test cases should be executed during the evaluation regardless of phase, but the execution report should also indicate whether the component has passed all test cases for one phase or the other.

2.4 Assumptions

2.4.1 Software Versions

Some pieces of software included in the evaluation may have versioned releases, with multiple versions available for installation at once. In such cases, only the most recently-released stable version of the software will be analyzed. The version of the software under test will be indicated in the component's execution report. If no version can be determined, other information should be provided to describe the specifics of the software as it was tested.

2.4.2 Cross-Platform Software

In some cases, a single application will nominally run on multiple platforms. Mozilla Thunderbird, for example, is available for each of the Windows, Apple, and Linux operating systems, among others. In such cases, we expect that the evaluation will include a separate analysis of the component on each supported platform (within the scope of the evaluation) unless it can be determined that doing so is unnecessary. The reason for this seeming duplication of work is that software, when deployed to different platforms, often makes use of different underlying software libraries to provide its features. Again taking Mozilla Thunderbird for example, a different library is used to provide the user interface (UI) on each of its target platforms. Because the behavior of the UI is relevant to EAI, we must consequently treat Thunderbird as a different piece of software on each of these deployment targets. This applies to EAI-related behavior in general. The platform of the software under test will be indicated in the component's execution report.

2.4.3 Functional Testing

The analysis consists primarily of functional tests, which is to say that they compare the result of an action with an expected behavior but do not concern themselves with implementation details or intermediate results. Tests that measure precise technical compliance to the relevant specification have only been included where a failing result would make further analysis useless, for example because other correctly-implemented EAI-ready software would treat the component under test as not being EAI-ready as a result of the failing behavior. Moreover, the tests are positive in nature, meaning that they present the software with valid inputs and user actions only. Defensive tests, such as those for malformed or malicious inputs, are not included. As such, this analysis is not a replacement for a more robust test suite to ensure software safety and correctness.

2.4.4 Test Data Adjustments

In some situations, test data may need to be adjusted in order for an analysis to be possible. If a component has requirements or limitations unrelated to EAI that conflict with the test data defined by this document, the test data should be adjusted to work around these factors. An example of such a situation might be a minumum length requirement on text for which only data under the required length has been provided. When adjusting test data, it is important that the primary property of each datum (as indicated by its description) be preserved.

2.4.5 Test Case Terminology

For brevity's sake, some terms are used throughout the attached Test Cases as shorthand for more involved concepts. Refer to for a glossary of terms.

3 Email Software and Service Ecosystem

Since this project's goal is to measure the EAI Readiness of the email software ecosystem as a whole, care must be taken to avoid any blind spots when selecting components for evaluation.

Although popularity is an obvious and important metric to consider, we cannot simply evaluate the most popular components in absolute terms or we may introduce a bias toward one geographical region, to the exclusion of others. Similarly, we cannot select too heavily from one category of software or we risk missing important EAI “bottlenecks” in the ecosystem. An EAI Ready mail client is of no use if there are no mail servers capable of transporting the messages it sends, for example.

3.1 Selection Criteria

With considerations such as these in mind, we have applied six primary criteria when selecting components for inclusion in the study.

3.1.1 Popularity

The most obvious metric to consider is popularity. To be included, a component should be widely used. This criterion ensures that each analyzed component contributes significant information to our understanding of the *general behavior* of the email ecosystem. Niche software should not be included unless doing so adds substantially to that understanding.

3.1.2 Language Community Diversity

Whether a component contributes diversity, defined broadly as expanding the geographical and language community scope of the evaluation, should also be considered. This criterion ensures that the project produces useful information for as many stakeholders as possible, regardless of language or location.

3.1.3 Technical Diversity

Because an email message is processed by many distinct components on its journey from sender to recipient, a broad set of software types must also be considered. Multiple applications should be selected from each of the categories defined by UASG 012 (cf. *Parts of the mail ecosystem*). This criterion ensures that both end-user client applications such as Mail User Agents as well as internal systems with no user-facing component such as Mail Submission Agents, Mail Transmission Agents, and Mail Delivery Agents are included. Refer to for more information about this project's evaluation categories.

3.1.4 Platform Diversity

To account for the variety of different software platforms underlying the email ecosystem, components from multiple operating systems should be included. In particular, each of the Windows, Apple, and Linux operating systems should be well represented in the selected set of software. The selection should also include software targeting both “traditional” computers (such as laptops and servers) as well as mobile devices (such as phones and tablets). This criterion ensures that the evaluation does not focus too heavily on software from any one host platform.

Note that this criterion does not extend to other types of hardware. The set of devices required by the evaluation will be limited to common computer, tablet, and phone models, and will include only commodity hardware.

3.1.5 Importance

In some cases, a component may be of critical importance to a given language community but not necessarily well known or widely deployed. As such, the relative significance of a component to its primary user base should also be considered. This criterion ensures that the study does not overlook users who rely disproportionately on a single piece of software or service provider.

3.1.6 Feasibility

Finally, it must in fact be possible to execute an analysis of a given component. Prohibitive license terms may exclude a given piece of software from the evaluation, for example. Security or anti-spam features of web-based services may also make analysis impractical. As such, this criterion can be thought of as ruling components *out* rather than in, and, while reasonable efforts will be made to evaluate the ideal selection of components, if an execution proves unfeasible this will be documented and the project will proceed without it.

3.2 Email Software Categories

Naturally, different types of software have different responsibilities, and so must be evaluated according to different criteria when measuring EAI Readiness.

In user-facing software, visual representation of addresses is a primary concern. For a mobile email client to be considered EAI Ready, for example, it must not only communicate with other software in a way that preserves EAI correctness but it must also accept and display valid email addresses using Unicode (as U-Labels), rather than using their ASCII equivalents (A-Labels). For a mail server, on the other hand, the exact representation of data is less important than whether it stores and transmits messages in an EAI-compatible way.

We have adopted the categories described in UASG 012 for use in this project. These categories partition email-related applications into Mail User Agents, Mail Submission Agents, Mail Transmission Agents, Mail Delivery Agents, and Webmail programs. To these, we add Mail Service Providers as defined by RFC 5598 to describe systems that provide email addresses and mailboxes to consumers or client companies. This yields the following six types of email software:

- **Mail User Agent (MUA):** A client program that provides a user interface for sending, receiving, and managing mail.
- **Mail Submission Agent (MSA):** A server program that receives mail from an MUA and prepares it for transmission and delivery.
- **Mail transmission agent (MTA):** 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.

- **Mail delivery agent (MDA):** A server program that handles incoming mail and typically stores it in a mailbox or folder, to be accessed by an MUA. For the purposes of this project, an MDA may also provide access to stored mail through a protocol such as IMAP or POP.

- **Mail Service Provider (MSP):** A system that provides email services and mailbox hosting for other organizations or users, on their behalf. An MSP typically provides some or all of the software types described above, and may also support the creation of new email addresses and mailboxes.

- **Webmail:** A mail client program created using a combination of the user's web browser and web servers that the browser connects to. A webmail application typically provides an MUA, and may also provide some or all of the software types described above.

We have found that, while software seldom fits nicely into a single one of these categories in practice, their names are nonetheless useful for describing the roles and responsibilities of email software components. An application that is nominally a "mail server" may behave as all of MSA, MTA, and MDA, for example, while webmail platforms often behave as both an MUA and MSP at once. As such, components will be evaluated on the basis of these categories as *roles,* with each subject to the test procedures for any role that it intends to fill. Where a given test case cannot be executed because the component does not fill the role under test, this will be indicated as "Not Applicable". Note also that the "Webmail" category is provided for informational purposes only and does not correspond directly to any set of tests.

3.2.1 Out-of-Scope Email Software Categories

The software categories discussed so far are by no means exhaustive, and there is almost certainly useful information to be gained by evaluating a more expansive set of software types. In particular, the behaviors of security and anti-spam software are likely to be relevant to EAI. These and other types of software such as mailing lists and email automation services have been omitted in order to limit the scope of the evaluation to a manageable size. However, we anticipate that the test procedures developed within the scope of this project will be adaptable to future evaluations that may include more expansive software categories.

3.3 Email Service and Software Selections

The set of service and software components to be evaluated is listed below.

Where possible, the ICANN Geographic Region with which each component is most closely associated is indicated under the **Region** column. This is generally the region in which the component's owning entity or primary development group is located. In situations where no region can be determined, the table has been left blank.

Whether a given component includes any behaviors relevant to each of the software types defined in is indicated by an **X** in the corresponding column. This is an indication that the component will be subject to that category's test procedures.

3.3.1 Table of Email Software and Service Selections

Software	MUA	MSA	MTA	MDA	MSP	Webmail	Region
XgenPlus Email Server	X	X	X	X	X	X	AP
Axigen Mail Server	X	X	X	X		X	EUR
MDaemon Email Server	X	X	X	X		X	NA
Oracle Beehive	X	X	X	X		X	NA
Zimbra	X	X	X	X		X	NA
Apple iCloud	X				X	X	NA
Coremail	X				X	X	AP
FastMail	X				X	X	AP
Gmail	X				X	X	NA
Mail.ru	X				X	X	EUR
Microsoft Outlook.com	X				X	X	NA
NetEase 163.com	X				X	X	AP
Oath Mail	X				X	X	NA
Rediffmail	X				X	X	AP
Sina	X				X	X	AP
Sohu	X				X	X	AP
Tencent QQ	X				X	X	AP
Yandex Mail	X				X	X	EUR
IBM Notes	X					X	NA
Roundcube	X					X	
Apple Mail	X						NA
Microsoft Outlook	X						NA
Microsoft Windows Mail	X						NA
Mozilla Thunderbird	X						NA
Courier		X	X	X			
IBM Domino		X	X	X			NA

Software	MUA	MSA	MTA	MDA	MSP	Webmail	Region
James Enterprise Mail Server		X	X	X			NA
Microsoft Exchange Server		X	X	X			NA
Oracle Communications Messaging Server		X	X	X			NA
Exim		X	X				
Halon		X	X				EUR
OpenSMTPD		X	X				NA
Postfix		X	X				
Sendmail		X	X				NA
Dovecot				X			
Fetchmail				X			
Procmail				X			

4 Results

4.1 Execution Summary

The results of the analysis of each individual piece of software or service should be reported in a document containing at least the following information:

1. Details about the software or service under test, including precisely which version of the component was analyzed, as well as links to its homepage and associated documentation. In situations where a single version number cannot be determined (for example, when the system under test is a web-based application hosted on an organization's own servers), the date and time of the test execution must be included, as well as any other information that may help to identify the software as analyzed.

2. A pass/fail indication for each test case.

3. A summary of the failing behavior for each failing test case. For test cases with multiple Use Case inputs, an enumeration of the failing Use Cases should also be included.

4. A summary of the overall results, including total pass/fail counts, and whether the software passed all of either the Phase 1 or Phase 2 test cases.

4.2 Test Outputs

Any modifications to the test data per should be recorded, and the modified data should be included with the execution report. Similarly, if a component is found to exhibit failing behavior when given specific inputs besides those listed in the attached Use Cases, for example on email messages of a

certain form, then an example of such an input should be included with the report, to make it possible for another party to reproduce, verify, and troubleshoot failing results.

4.3 Test Automation

It is expected that many of the project's test cases, particularly those analyzing the behaviors of MSA, MTA, and MDA components, will be automatable. If test executors implement any software in support of the evaluation, they are encouraged to include these programs with the test results, or to make them available online and include some information about how they can be accessed, so that others may benefit from them during future evaluations.

5 Proof of Concept

The next phase of the project will carry out a proof of concept evaluation. This "pilot program" will analyze a small but representative set of software components, and will help to shape the tools and methodologies to be applied in phase three.

We suggest that the pilot include **Gmail** (as an MUA), **Courier** (as an MSA, MTA, and MDA), and XgenPlus (as an MSP). These three components span each of the five test categories, and, due to their large feature sets, will exercise a majority of the test cases defined by this report. The inclusion of Gmail also provides the opportunity to test both a Webmail application (gmail.com) and a native mobile application (the Gmail application for Android). This should provide experience and uncover issues applicable to most future evaluations. Additionally, all applications are either free to use or offer a trial period, which will help to reduce the cost of the pilot.

Taking into account the software to be analyzed, the size of the test suite, the repetition involved in executing many of the test cases, and the overhead introduced by administrative tasks such as configuring software and creating user accounts, we estimate that this proof of concept evaluation will take 120 hours to execute:

1. **Administrative tasks:** 12 hours
2. **Gmail (Android) – MUA behaviors:** 24 hours
3. **Gmail (Webmail) – MUA behaviors:** 24 hours
4. **Courier – MSA behaviors:** 16 hours
5. **Courier – MTA behaviors:** 16 hours
6. **Courier – MDA behaviors:** 24 hours
7. **XgenPlus – MSP behaviors:** 4 hours

These estimates assume ready access to all necessary software and hardware, such as a Linux server on which to install Courier and run other software in support of the tests and an Android device or emulator with which to test the Gmail application. A mechanism for inspecting network traffic sent between pieces of software is also required by many of the test cases.

6 Document History

Version	Author	Date	Description
0.1	Evan Hanson	19 June 2018	Preliminary selection criteria and component selection.
0.2	Evan Hanson	29 June 2018	Reformatting, clarifications, and additional components.
0.3	Evan Hanson	3 July 2018	Clarification of MDA category and software table corrections.
0.4	Evan Hanson	12 July 2018	Software table updates, and criteria, and test procedures.
0.5	Evan Hanson	29 July 2018	Software table updates and test cases.
0.6	Evan Hanson	31 July 2018	Test case updates and terminology.
0.7	Evan Hanson	6 August 2018	Updates per community feedback.
0.8	Evan Hanson	8 August 2018	Add and .
0.9	Evan Hanson	29 August 2018	Update test cases per community feedback.
1.0	Evan Hanson	14 September 2018	Change Proof of Concept to include Phase 2 MSP

7 Appendices

7.1 Appendix 7.1: Test Cases

The test cases are defined within UASG021A Test Cases.

7.2 Appendix A: Terminology

- **Address:** An email address, typically but not necessarily in reference to an <Email> Use Case. A **non-EAI address** is one comprised entirely of ASCII characters, while an **EAI address** is one that may contain UTF-8 text.
- **Assigned address:** An email address that is generated by a Mail Service Provider and allocated to a user when they register an account.
- **Destination address:** The value of a header indicating a message recipient, one of “To”, “Cc”, or “Bcc”, as defined by RFC 5322.
- **Destination server:** The Mail Transfer Agent to which an MSA or MTA is transmitting a message, as distinct from the software under test itself.

- **Message:** An email message, including its envelope where applicable. A **non-EAI message** is one comprised entirely of ASCII characters, while an **EAI message** is one that may contain UTF-8 text in its envelope or headers.
- **Originator:** The value of a header indicating a message sender, either “From” or “Reply-to”, as defined by RFC 5322.
- **Traffic:** Messages that are transmitted between programs at the protocol level, for example an EHLO message in the SMTP protocol or a LIST message in POP. This term is used to distinguish a Layer 7 transmission from a “message”, which is the term used to refer to email messages specifically.