Universal Acceptance community and work

Mark W. Datysgeld – ICANN GNSO Councilor

2022

Universal Acceptance

The Universal Acceptance Steering Group (UASG)

Procedures

- Supported by ICANN, but fundamentally independent.
- Priorities are set by the community based on feedback and real world experience.
- Projects and research proposals are developed by Working Groups in a collaborative manner.
- Working Groups meet once every one or two weeks, each discussing their own area of focus, but also taking into consideration broader UA goals.

UASG in practice

Working Groups

- **UA-Communications**
  - Responsible for communication strategy and engagement with both the ICANN community and the general public.
- **UA-Technology**
  - Responsible maintaining relationships with organizations and structures outside of ICANN (such as the IETF), bridging the gap between technical and social concerns. It both takes information out of the UASG and brings it in.
- **UA-Measurement**
  - Responsible for the coordination, analysis, and dissemination of statistics related to Universal Acceptance. Much of the data presented in this course was organized by this group.
- **UA-EAI (Email Address Internationalization)**
  - Responsible for studying the technical aspects of Universal Acceptance, understanding the interaction between email systems, servers, programming languages and international standards.

How to participate effectively?

- We are very welcoming of new volunteers, and the process of joining a Working Group is simple. Just find an area you care about and start following its mailing list. When you have the opportunity, enter one of our regular calls and introduce yourself.

- Don't be afraid of not understanding some or most aspects of the work, and be willing to ask questions. There are many different topics involved in UA, and not even our most experienced members understand them all!

- If you don't want to participate directly in the UASG but would still like to develop your own research or make a system UA-Ready, we're always happy to help. We maintain an archive of success stories just waiting to have your name in it.

UASG Ambassadors

- Some community members take further responsibility by becoming **Ambassadors**, seeking to:
  - Spread the awareness about UA in your region, using any languages you know.
  - Learn and find ways to explain the group's research and findings to varied audiences, whether technical or the general public.
  - Represent the UASG at academic, professional, or other educational events that advance the group's goals.
- **Requirements**
- Fluency in English is required.
- An already existing, or a great desire to acquire, intermediate knowledge of email systems and their workings (MX, MDA, MUA, etc.), as well as basic knowledge of the DNS.
- Availability to make presentations about UA for different audiences.

Examples of finished UASG research

- UASG 028: *Considerations for Naming Internationalized Email Mailboxes*

https://uasg.tech/wp-content/uploads/documents/UASG028-en-digital.pdf

Developed by the EAI-WG, this document makes several considerations about naming internationalized mailboxes. A good example of UA knowledge applied to the real world.

- UASG 032: *UA of Content Management Systems (CMS) Phase 1 WordPress*

https://uasg.tech/wp-content/uploads/documents/UASG032-en-digital.pdf

Developed by external contractors, it is a recent example of how extensive UA testing is performed, taking into account the multiple possible use cases, such as different combinations of plugins.

Our recently finished research

UASG 033: *UA Readiness of Open Source Code Pilot: Java and Python on Github*

- **Objective**: Evaluation of strategies to discover and improve UA-related components in open source software.

- **Methodology**: A crawler was coded to find "dependency files" of all Java and Python projects on Github, containing a list of all components needed to compile or run a software.

- With this information, it was possible to understand the popularity of dependencies associated with UA, as well as how they are being used. This now allows us to prioritize partnerships and remediation efforts.

Findings: Java

UA-related Libraries		
Library	**Occurrence in projects**	**Status (Source)**
icu4j	886	IDNA2008 (UASG018A)
libidn	29	IDNA2003, deprecated and ported to the Java language as "*java.net.IDN*". (Documentation)
commons-validator	4906	Relies on a static list of TLDs from 2017. (UASG018A)
validation-api	25190	IDNA2003 implied, RegEx via annotations. (Documentation)
springfox-bean-validators	12501	IDNA2003 implied, RegEx via annotations; SpringFox implementation of *validation-api*. (Documentation)
hibernate-validator	62963	IDNA2003 implied, RegEx via annotations; Hibernate implementation of *validation-api*. (Documentation)
***Occurrence in entire dataset**: 70,182; approx. 6%.*		

RegEx via annotations seems to be a popular method of performing validation in Java, which is unfavorable to the UASG's interests. validation-api ranks 55th overall in terms of usage, and its derivative hibernate-validator places even higher at 21st. springfox-bean-validators also ranks quite high at 79th.

Findings: Java

#	Dependecy
1	junit
2	mysql-connector-java
3	spring-boot-starter (family)
4	lombok
5	spring (family)
6	commons-lang (2 and 3)
7	jstl
8	slf4j
9	log4j
10	javax.servlet-api
11	commons-io
12	jackson-databind
13	h2
14	fastjson
15	gson
16	hibernate-core
17	spring-tx
18	commons-fileupload
19	servlet-api
20	mybatis

#	Dependecy
21	hibernate-validator
22	guava
23	springfox-swagger2
24	mybatis-spring
25	logback-classic
26	postgresql
27	jsp-api
28	jackson-core
29	selenium-java
30	mybatis-spring-boot-starter
31	gson
32	testng
33	httpclient
34	hibernate-entitymanager
35	druid
36	springfox-swagger-ui
37	aspectjweaver
38	commons-codec
39	httpclient
40	commons-dbcp

#	Dependecy
41	poi
42	jcl-over-slf4j
43	poi-ooxml
44	aspectjrt
45	assertj-core
46	jackson-annotations
47	commons-logging
48	jackson-mapper-asl
49	mockito-all
50	jedis
51	javax.inject
52	c3p0
53	json-path
54	validation-api
55	commons-collections
56	jjwt
57	mail
58	standard
59	tomcat-embed-jasper
60	javaee-web-api

Findings: Python

UA-related Libraries		
Library	**Occurrence in projects**	**Status (Source)**
idna	70789	IDNA2008 (UASG018A)
pyicu	243	IDNA2008 (Documentation)
idna_ssl	10	IDNA2008 (Documentation)
email_validator	1178	IDNA2008 (UASG018A)
validators	1660	Email validation based on Django validator, Not compliant; URL validation based on *regex-weburl.js*, which is a RegEx. (Code analysis)
***Occurrence in entire dataset**: 70,813; approx. 37%.*		

Out of the entire Python dataset, the "idna module" ranks 6th overall in terms of usage, which can be seen as a favorable result to the UASG's interests. It can also be a key argument in engaging with the Python language developers to port that module to the language's core, replacing the default IDNA2003 implementation.

Findings: Python

#	Dependecy
1	six
2	requests
3	pytz
4	certifi
5	urllib3
6	idna
7	chardet
8	Jinja2
9	MarkupSafe
10	python-dateutil
11	numpy
12	click
13	pyparsing
14	Werkzeug
15	Flask
16	attrs
17	jupyter
18	Pygments
19	wcwidth
20	pandas

#	Dependecy
21	decorator
22	itsdangerous
23	matplotlib
24	PyYAML
25	scipy
26	traitlets
27	pickleshare
28	ipython-genutils
29	prompt-toolkit
30	mccabe
31	wrapt
32	tornado
33	gunicorn
34	jsonschema
35	cffi
36	entrypoints
37	pycparser
38	pexpect
39	packaging
40	cycler

#	Dependecy
41	webencodings
42	bleach
43	ptyprocess
44	sqlparse
45	backcall
46	isort
47	colorama
48	pylint
49	nbformat
50	defusedxml
51	mistune
52	nbconvert
53	lazy-object-proxy
54	astroid
55	notebook
56	kiwisolver
57	py
58	zipp
59	terminado
60	pandocfilters

Upcoming research (1)

UA Readiness Evaluation of Standards and Best Practices

- **Objective**: Gap analysis of standards bodies, seeking avenues for the promotion of UA within them.

- **Examples:**
- **W3C Internationalization Working Group (i18n)**: one of the groups to work the closest with matters related to UA outside of the UASG itself. The lack of communication between both groups may be leading to missed opportunities. This WG is very active and reviews a diversity of questions related to i18n, providing recommendations and fixes. Closer cooperation could lead towards gains for both parties.

- **WHATWG HTML Living Standard:** As indicated in "UASG 025: Global Evaluation of Websites for Acceptance of E-mail Addresses in 2019", the latest iteration of HTML5 only partially supports UA, a situation that has been subject to discussion by multiple interested parties. A SOW should be made for a contractor to develop appropriate code and, with the assistance of the UASG as a whole, navigate the community's processes in order to get this change approved.

Upcoming research (2)

- **IRTF Hrpc:** Human Rights Protocol Considerations Research Group: the innovative nature of the HTPC's approach center on the Human Rights dimension of protocols and standards makes it a suitable home for UA-related discussions. The group is relatively recent and its focus on matters of access and free communication are a close match to the UASG's goals. Interaction with the group could help the UASG reach out to several other IETF groups, given the Hrpc's transversal nature in terms of recommendations.

- **WHATWG HTML Living Standard:** As indicated in "UASG 025: Global Evaluation of Websites for Acceptance of E-mail Addresses in 2019", the latest iteration of HTML5 only partially supports UA, a situation that has been subject to discussion by multiple interested parties. A SOW should be made for a contractor to develop appropriate code and, with the assistance of the UASG as a whole, navigate the community's processes in order to get this change approved.

- **Python Software Foundation PEPs:** "UASG 033: UA-Readiness of Open Source Code Pilot" and the core PyPI repository both show that the idna module, which upgrades Python's core implementation of IDNA2003 to IDNA2008, finds very significant adoption among open-source developers. A proposal should be made to integrate the module into the core of the language, making it UA-Ready at its foundation. This can be achieved by means of Python Enhancement Proposals (PEPs), which should be steered by a qualified contractor via a SOW.

Join us!