UA EAI WG Meeting

25th Oct 2022

Attendees

Mark Svancarek
Abdalmonem Galila
Jim DeLaHunt - Canada
Harsha Wijayawardhana
Sushanta Sinha
Jules Nizeyimana
Jessica Dadzie - Ghana
Amina Ramallan - Nigeria
Yin May Oo
Seda Akbulut

Agenda

1. Welcome and roll call
2. Drafting a flowchart of the action items from the FY23 Action Plan
    1. Developing a roadmap for implementing the EAI self-certification guide
    2. Statement of Work (SOW) for E1.1 and E1.2
        1. E1.1 Building a self-certification tool to generate EAI readiness score.
        2. E1.2 Helping early EAI providers perform self-certification using the guide.

a. User acceptance tests

b. A quick guide for IT and procurement managers.

2. Other action items in the FY23 Action Plan to be included in the flowchart

. E2 Make it easier to experiment with a self-hosted working EAI system

a. E3.1 and E3.2 that can be done in parallel

E3.1 Identify reference customers to showcase adoption of globally inclusive email, and document the experience (customer studies)
E3.2 Identify reference mail service providers to showcase for adoption of globally inclusive email, and document the experience (provider stories)

0. Shifting from 14:30 to 15:30/16:00 UTC for meetings between 6 Nov 2022 and 12 Mar 2023? https://www.timeanddate.com/time/change/usa?year=2023
0. AOB

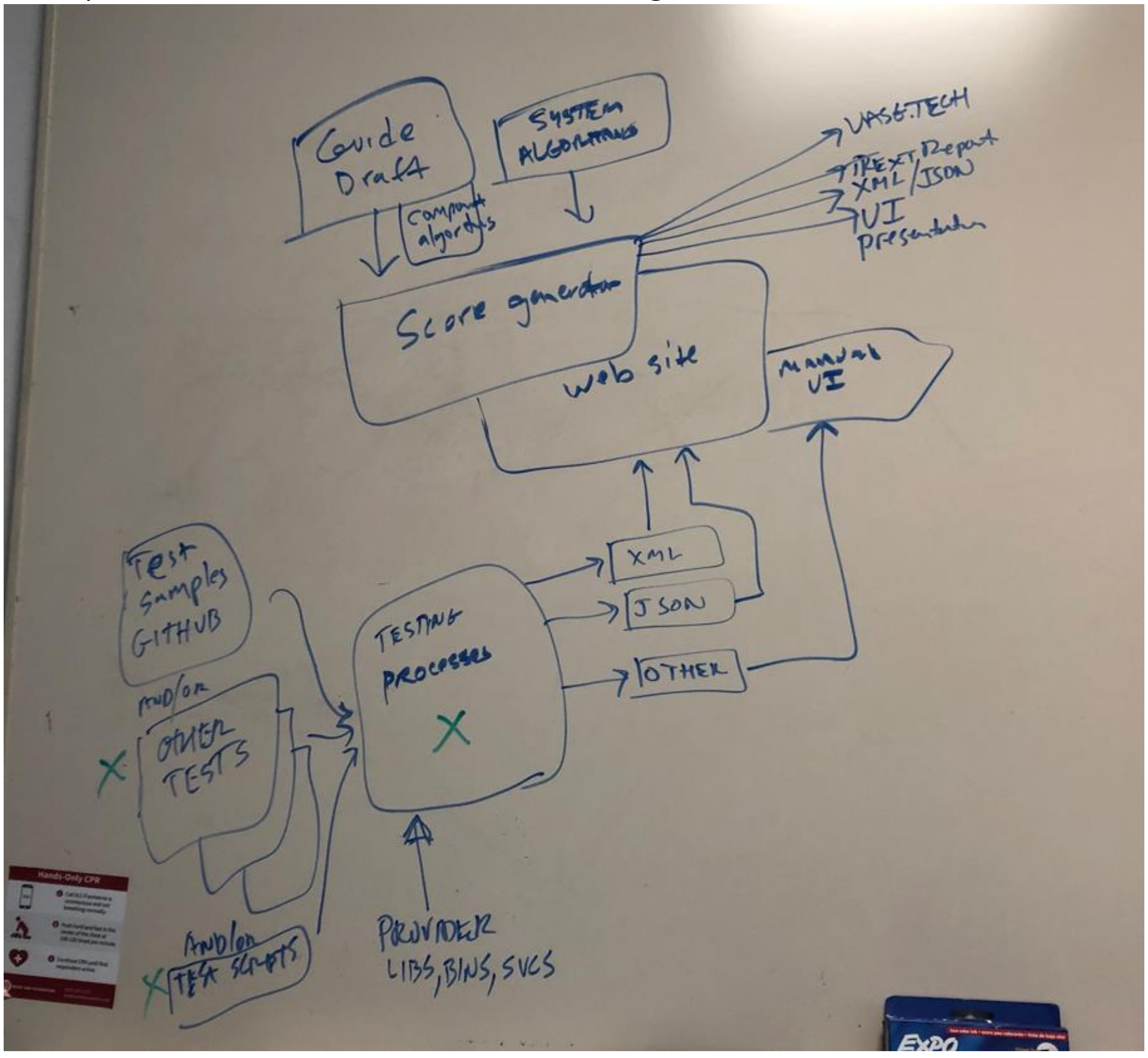
Meeting recording:
https://icann.zoom.us/rec/share/zmclsUa7gidUz8jCoh62HRIqFTVNyBKuoyvi-9qAA8P476HB1rjn-2FqDUVHLwBE.rNKpOQBKJwYfQRXz Passcode: r#nFfeK#68

Meeting Notes

Developing Roadmap for implementing EAI readiness score and self-certification

Mark said that our main work item is to develop a self-certification guide and we are at a point where it is good enough for someone to try implementing it, which seems more than 90% ready. And the statement of work item E1.1 is building a self-certification tool to generate the EAI-readiness score, and E1.2 is helping early adopters to perform self-certification using the guide. There would be a user-acceptance test, quick guide, etc. to be included in the flow chart. E2 is to make it easier for experimenting and there are E3.1 and E3.2 as part of the agenda.

Mark presented the reference architecture diagram:



Jim transcribed the the words on Mark's diagram:

> Testing process (lower part of diagram) --> XML | JSON | Manual output --> Evaluator (upper part of diagram)
>
> Testing process (lower part of diagram)
> Test samples (GitHub)
> and/or Other tests
> and/or Test Scripts
> --> Testing processes
> Provider provides libs [libraries], bins [binary code], svcs [services].
> --> XML -->
> --> JSON -->
> --> Other -->
>
> Evaluator (upper part of diagram)
> --> web site
> -->/
> --> Manual UI
> Score generation
> Guide draft
> Component Algorithm
> System Algorithm
> --> UASG.Text
> --> Text Report
> --> XML/JSON
> --> UI Presentation

Along the top is the site, and along the side are the tasks. There are test samples and sample codes on Github, made by John Levine (https://github.com/jrlevine/eaitesttools) using python two years ago. We can assume there is some test in public repositories. There is also the test script and they are on the left side of the diagram. There're the testing processes that someone has to put together. And if somebody's providing libraries and other services, all this stuff is being put together so that you can test your offering. And then this is going to export something. We haven't decided what to export. It should be XML or JSON.

Any programming language can be used for the user interface, so the people could use and test. Above is what the vendor will be creating. At the top there is a draft Guide and component algorithm. The idea is based on the guide, looking at the scores coming up, the algorithm will generate an output and issue your scores. The score is to be generated on a component by component or requirement by requirement basis.

There are algorithms for generating scores and components which have algorithms for generating the score of a system. These all will be part of a score generator component, and the website is in front of all of these that will allow you to submit the output in a decided format.

There will also be manual input User-Interface(UI) to allow manually adding scores through the site. That's where we start and then the score generator can be exported to uasg.tech in desired format, so that you can see how it renders to the pages. Mark questioned what the vendor would work with. Mark would like to make this as simple as possible for someone to understand easily, and to put it in contract simply. This was the first pass by Mark. He asked if this is enough to make sense.

Jim expressed that it was too simple. He asked if this is the minimum viable product (MVP) desired? Mark questioned what we should take out for the very first version.

Mark said, referring to the diagram, everything below the three up arrows is the provider. The vendor would only create the items above those arrows. In order for this to work, test results from people are needed. Mark said that it would be good to **draw a dotted line there**, everything below where it says, website and manual UI, that's up to the person doing the testing; everything above those three arrows, that's what we want the vendor to do.

Sushanta asked what refers to input and output.

The Inputs to the system are the test scores from the guide for each component. So the guide says you might have ten things for an MUA and it might have thirty things for an MSA. Each of those tests is a pass/fail. And then we could submit all those pass/fails. As we are testing an entire system, end collection of pass/fails will be the part of the output. (For example, when you submit a pass/fails to system, it will produce a result: your MUA score is gold, but your system is Silver.) So that's what those upward arrows are for. According to the guide's suggestion, the output will interpret and rank the system and produce a trusted certificate.

Abdalmonem said there is a need to create test cases for different languages and different email addresses. Rebuild the test cases for LTR scripts, RTL scripts and font rendering etc. Abdalmonem suggested the diagram to come up with front-end vs back-end, whether the testing is on EAI-readiness (Back-end) or display of labels/URLs on browser (Front-end) for both LTR and RTL scripts. He also shared the following: - It's API should be tested. Is the API compatible? We need test cases to test the system. Which programming language is targeted to be used for implementing the system? JavaScript?

Mark agreed that there are no test cases yet. He said that we need to note this feedback for the SOW. (Action Item)

Jim agreed with the diagram and where the product of this system is. He suggested naming the lower part of the diagram as "testing process", and the upper part as "evaluation process". Then, each can be broken up into front-end and back-end.

For the vendor created website, the xml or json format input file would be the simplest to work on. The work has two parts, one is to build the software part of the evaluation architecture, and also build the interface module for the testing process of the architecture. Jim suggested the minimal version of this would be for humans to run the tests on what they've got, which may not be enough, and generate a spreadsheet with score results. Humans can manually create a spreadsheet with scores and check how the evaluation algorithm works.

Before we can issue a statement of work, we may need to run this manually for the first a few times. Mark agrees with the idea of using a spreadsheet as the input of the evaluation system. Per Jim, in the initial version: humans open the spreadsheet for evaluation and teach us about the algorithm to show us how it works before we run into the whole system. Spreadsheet is kind of a simple way of software development.

Mark clarified that E1.1 is the top of the picture. E1.2 is the bottom of the picture.

Harsha said the evaluation system is not totally automated and there are places where humans need to manually check if the displayed text is readable and so on. Mark said in the initial stage, there must be a way for humans to manually check and mark the pass/fail scores, and then eventually build towards automated testing.

Abdalmonem said the testing structure may be different for many different application types. Some of them may accept only one URL or email address while some may accept LTR scripts only and so on. We need to develop the minimum test cases to handle Universal Acceptance. There should be a database with different types of email addresses and domain names to test applications and see which crashes their process. Mark said he thinks only glyphs-checking is required in manual form and the rest could be automated. Harsha said things could be automated at the Unicode level like an API. Mark shared that he thinks that people in the community will create more test cases.

Mark asked and Harsha answered about who could do the testing process API, the vendor could do it, and the respective community's input would be integrated with it.

Mark suggested breaking the diagram into three parts:

1) First, the top part of the diagram, which is the evaluator
2) The second is the middle part where there is data to be submitted to the evaluator. (.csv spreadsheet) Mark likes this idea as spreadsheet is the simplest form.
3) The third is the lower part of the diagram, which is the testing process. More complete and separate design is required for the testing process.

Jim said, for the data to be submitted to the evaluation part, someone has to write software to perform the tests according to the samples by John Levine. And another simple software is needed to send emails and look at the pass/fail results. Before specifying the diagram and ask someone to build a software, we need manual evaluation that brings us to E3.1 and E3.2 to perform some evaluation with customers and providers.

Mark said the reference diagram needs to be fixed and the bottom part of the testing process should be more detailed, however, it will prioritize the score spreadsheet and evaluation algorithm.

Jim suggested it would be better to plan how to write the test cases and example results, so people can refer to this as Abdalmonem suggested as well. Jim suggested writing down all the ideas and he would be happy to review them. He said manual evaluation will be attempted to Microsoft 365 email functions and external data mail.

Mark concludes the meeting with a thank you message and assigned action items for the next meeting (see the action items list below this document). All agreed.

Shifting meeting time

Jim suggested shifting the meeting time as it will become too early with the Daylight Saving adjustment. It was decided to change the meeting time from 14:30 to 15:30 UTC for meetings between 6 Nov 2022 and 12 Mar 2023. https://www.timeanddate.com/time/change/usa?year=2023

Next Meeting: Tuesday 1st Nov 2022 at 14:30 UTC

Action items

No.	Action Item	Owner
1	Fix the diagram's top part based on the discussions on this meeting	Mark
2	Make a simple spreadsheet (csv file)	Mark
3	UAT for the self certification guide by Microsoft India Team with students	Mark
4	Do the bottom-left of the diagram by the next week (Draw a flow chart or a reference architecture, or even just a bunch of words would suffice) (Components required for the testing part)	Abdalmonem and Jim
5	Share the picture via email	Seda
6	Add test cases information to SOW as per Abdalmonem's feedback above	Mark and Abdalmonem
7	Build up the test cases and examples	Abdalmonem