EPUB 3.1 : Server/Browser Friendly Manifestations

What would EPUB without ZIP look like? OCF 2.0.1 had a file system container that was removed in OCF 3.0. Readium “explodes” EPUB files and rewrites some of the XML as JSON (does anyone have more details on this?). In this subgroup we’ll explore how to make EPUB more server- and browser-friendly.

Participants

Dave Cramer (Lead) (Hachette Book Group)
Tzviya Siegman (Wiley) (cheerleader)
Ivan Herman (W3C)
Matt Garrish
Ori Idan
Daniel Weck (DAISY / Readium)
Hadrien Gardeur (Feedbooks)
Patrick Keating (Bluefire)
Leonard Rosenthol (Adobe)

Brady Duga (Google)
Tom Cleghorn (Cambridge University Press)
Markus Gylling (IDPF)
Bill McCoy (IDPF)
Jean Kaplansky (Safari)
Eunmi Jung (KERIS)
Yong-Sang Cho (KERIS)
Makoto Murata (JEPA)
Phil Burrows
Sanders Kleinfeld (O'Reilly)

Calls

Telecon April 18, 2016 at 19 UTC

Participants

Dave Cramer (Hachette Book Group)
Markus Gylling (DAISY & IDPF)
Tzviya Siegman (Wiley)
Sanders Kleinfeld (O'Reilly)

Agenda

- Detachment from EPUB 3.1.
    - does the mission remains the same? For example are we still tied to 3.1 and full round-tripability?
    - could we have a clear idea of how our first "evergreen" spec will be developed and maintained (vs the current charter based revision model)?
    - could we keep in 3.1 a way to link back to BFF (link@rel="alternate")?
    - could we make sure that the registries that we're currently using can be shared for both 3.1 and BFF?
    - Should we move the stuff to IDPF github.
- Data model
    - The links array vs the link object
    - Resources vs external resources
    - Collections vs renditions, collection roles vs package types, etc.
- https://github.com/dauwhe/epub31-bff/wiki/EPUB-3.1-F2F-Meeting-Detailed-Agenda

Discussion

Dave: first major purpose of this meeting is to sort out issues surrounding detachment from 3.1.

Does the mission remain the same?
Dave: my instinct, yes remains the same. Garth: agree we should keep roundtripping constraint.
Dave: we need to maintain a strong relationship with classic EPUB. Roundtripping in both directions should remain a goal.
Markus: +1.
Markus: are there other aspects of the mission that we should reconsider? E.g. scope as described by use cases...
Dave: don't think it changes too much. We need to be mindful of other projects such as PWP but we have our EPUB scope.
Hadrien: does roundtrippability include satellite specs? E.g. indexes, AHL
Markus: suggest we keep the scope to 3.1 only for now at least.

- Agreed: keep mission/scope and core 3.1 roundtrippability constraint

Could we have a clear idea of how our first "evergreen" spec will be developed
Dave: since we didn't have a clear idea before...

- Decision: we move the stuff to IDPF github.

Sanders: moving issue tracker might involve using the github API.
ACTION Dave: I'll look into what would be involved. Sanders helps out too. (Potentially use this to migrate issues: https://github.com/google/github-issue-mover)

Markus: re development and maintenance: either you target an IDPF Informational Document as the first step, or you just stay on GitHub publishing HTML and IDPF helps alert community when you want to make calls for implementation etc.

Tzviya: if we dont publish noone will ever look at it. We need a timeline. We can maintain at a different pace.

Dave: concerned about informational document route because with FXL we were following existing implementations, in this we are much more experimental. We would target FPWD this summer. And then from there, maybe it ends up being publishing as Informational Document.

agreed: use idpf.org to publish editors drafts, public drafts, and eventually (probably) Informational Document. Put administrative documents (timeline etc) on github using e.g. the readme file.

Hadrien: do we want calls on a regular basis? Dave: should we try every other week? Think once a week is too much right now.

ACTION Dave put together doodle poll for regular call time.

could we keep in 3.1 a way to link back to BFF (link@rel="alternate")?

Hadrien: make sure the rel=alternate
ACTION Matt: make sure the detached rel registry contains the value alternate with the same definition as HTML rel. (See also https://github.com/w3c/html/issues/160)

could we make sure that the registries that we're currently using can be shared for both 3.1 and BFF?
Hadrien: semantics for collection roles etc.
Agreed: share registries at all times. If some values are BFF-specific, they are simply noted as such as in the registry.

Tzviya: should we not use microformats registry instead?
Hadrien: depending on which link element you are using the extension mechanism can be different. In IETF specs, you need an RFC for a new term, or use a URI, that's different from e.g. rel in HTML.

Dave: another example is manifest item properties, which are not in a separate registry but in the core spec.
Hadrien: the idea is to have a unique set of term for item properties, and you can use them everywhere.

Hadrien: what if we want to do things that are not allowed in classic EPUB, e.g. images in spine.
Dave: hard to have a general policy on these things. I'd say we try to approach each one on its own merits.
Hadrien: just want to make sure we don't close this door completely.

Dave: if we are writing this up as a semi-formal spec, there's the matter of the editors. I'm happy to be one, Hadrien as well. Maybe Ivan will be interested too, I will contact him.

Tzviya: I can be a peer-review editor.

Data model

- The links array vs the link object
  Dave: object with reqd href and type key and optional keys like rel, title, properties, etc
  Think we are happy with this conceptual building block. So for example the spine could be an array of these link objects. Resources array which is all the other components which are not part of the linear reading order. Together these two arrays along with metadata can describe what we have in a regular single-rend epub.
  Hadrien: link object is the same thing as link and collection but slightly more powerful. What resources in spine are is actually a collection, we havent changed the model, what we've discussed though is to have a more compact way for the syntax to retain that model, and for certain collection roles what seems to possible is to only have an array of link objects, whereas others need to have subcollections. Without changing the model it

is possible to lean more towards compactness. I think the model remains the same, we are just finding new ways to make the syntax more compaxt.
Tzviya: collection has its own meaning in JSON.
Hadrien: I think we use our word in the same way as the collection+json spec.
Tzviya: if Ivan had trouble understanding something, we might have trouble with a less educated audience.
Dave: to me the collection as defined in EPUB is not a natural starting point. The way its described now is really odd.
Hadrien: the problem is that there is no standard model for EPUB. The goal of collection was to have something in between: to talk about groups of items.
Dave: now we seem to be crossing some boundaries between structural components and metadata. Is the spine a collection that doesnt happen to need metadata?
Hadrien: either you follow EPUB's route, define elements that you need, or you have a data model. This is flexible, easy to evolve over time. I strongly prefer having constraints and a model.
Tzviya: OPFs collection has been used in many many ways. Whatever we do for BFF, Dave's point is that we need to make sure that we are starting from a point that is understandable both for users and browsers.
Dave: its possible that we can agree on the syntax and explain it different ways.

- Resources vs external resources
    -
- Collections vs renditions, collection roles vs package types, etc.
    -

Telecon March 10, 2016 at 15 UTC

Participants

Dave Cramer (Hachette Book Group)
Tzviya Siegman (Wiley)
Ivan Herman (W3C)
Ric Wright (Readium)
Hadrien Gardeur (Feedbooks)
MURATA (JEPA)
Bill McCoy (IDPF)

Agenda

Updated single proposal: https://github.com/dauwhe/epub31-bff
Issues inline in that document or in github itself

Discussion

Dave: Merged everything into a single document, differences highlighted as inlined issues Broad agreement on structure, use of JSON and metadata

Murata: In disagreement, thinks that explosion and serialization are two issues and therefore steps. Should we stick to explosion for 3.1? We can already allow content documents to be shared between publications.

Hadrien: We will still have structure, which is complicated. But, we will not improve experience or move toward OWP.

Tzviya: Is it an issue if we do any of that?

Bill: Let's focus on technical issues, no known issue at a board level from Japanese publishers about our goals here.

Dave: Useful, need to be clearer about things. Place to discuss how, not why. Would like to have more feedback.

Tzviya: Need a date and a timeline.

Dave: Will collect and divide issues.

Dave: Hadrien's proposal turns rendition into a proper concept and piece of EPUB's structure. Can we simplify though for people who only need one?

Issue 1: Renditions. Given that most books contain a single rendition, is there a way to avoid having a rendition object in single-rendition books?

Ivan: With a single rendition, sounds logical to drop the renditions object.

Hadrien: Including rendition metadata for one or many renditions is good practice because it means the processor behaves the same way whether or not there are multiple renditions. Also provides a clear separation between publication and rendition metadata.

Ivan: It might take the burden off the processor, but it adds burden to the author. It's usually better to chose the author over the processor.

Hadrien: It is a minimal burden

Tzviya: Rendition only required when it's not default.

Hadrien: This enables separating rendition (display) metadata and other metadata. This also allows is an opportunity to refine and think about what we are doing. This allows for, not only reflowable but all types of publications.

Makoto: Are you re-engineering EPUB or making it available on the Web?

Hadrien: We don't have many opportunity to "think" what EPUB is. The move to a different serialization and access is a unique opportunity.

Dave: Is the idea of multiple renditions a core value of EPUB ? Not a very "webby" thing.

Ivan: Agree with Hadrien that if it's a central concept, the structure is OK. This is valid only if it's a central concept.

Tzviya: Let's make sure that we have a proposal to put forward to the group, reflecting information about explosion and serialization.

Hadrien: We have agreed on working in JSON-LD and creating a context. We need to agree on exact details.

Dave: Working on it. It has proven to be a little more difficult because of schema:Book, and I would like to skip layers

Hadrien: One thing to focus on is how the metadata would look in schema.org. We want something that works for those who are familiar with JSON-LD and for those who have never heard of it.

… (sorry but had a hard time moving between the call and the notes)

ACTION: Dave and others to add issues to https://github.com/dauwhe/epub31-bff/issues to formalize discussion so that we have a better understanding of discussion and issues around it
ACTION: Makoto to provide information about exploded distribution and/or alternation

Telecon February 16, 2016 at 16 UTC

Participants:

Dave Cramer (Hachette Book Group)
Markus Gylling (IDPF & DAISY)
Romain Deltour (DAISY)
Matt Garrish
Tzviya Siegman (Wiley)

Yong-Sang Cho (KERIS)
Hadrien Gardeur (Feedbooks)
Garth Conboy (Google)
Brady Duga (Google)

See comments on
https://github.com/dauwhe/epub31-bff/commit/cafee9bdf4d58e7eba12872354ea813464709ff4

Agenda:

1. The data model. Does it need to support linear=no? does it need to support everything in the extension specs? Do we still think some of those things are a good idea? (cough, index-group, cough).
2. Relationship to SEO, schema.org, and linked data
3. Single-page documents
4. How does the nav doc fit in?
5. If this is such a good idea, should we consider a new XML serialization?
6. How might these proposals allow us to do cool things we can't do in 3.0.1?
7. Relationship to Web application manifest
8. Relationship to OPDS
9. Relationship to PWP
10. How long can we pretend this isn't version 4.0?
11. Do we have enough implementation experience (in the Extensible Web Manifesto [3] sense) to make these kinds of decisions now?

Discussion

Dave: wrote up these questions based on discussion on list and on repo. Hadrien, where are you right now?
Hadrien: One thing we discussed a lot was JSON vs HTML, agreed that its easier to build something based on JSON. Second thing: if we adopt json, how do we actually included it? Linked to or embedded. Majority of tracker discussions are about that. On that point, we did touch upon different use cases on how the manifest is used (by browsers, apps, opds clients etc). What's not in the agenda is the extensibility model for the current proposals. Points listed on mailing list:

- collections roles
- metadata for collections
- link relationships
- link properties

Dave: brings up a meta question of our relationship to HTML in general. I get the sense that what you are trying to do is build this model for a digital publication that hardly touches on HTML

at all. In some general sense I get this idea that it's sort of not taking advantage of what we have in HTML right now.

Hadrien: we do have a different perspective. Garth summarized well: is the heart of EPUB the OPF or the NavDoc? Rather than saying I don't want to tie it to HTML, I'm saying it shouldn't be limited to HTML and browsers, including support for native apps etc.

Tzviya: I think we all love that idea, but its a huge change for a dot release.
Hadrien: conceptual model the same, serialization different.

Hadrien: adopting a conceptual model that enables us to better summarize and express what an EPUB is, with syntax pretty much aligned with the web (webapps manifest, etc). When I think about the web, I dont limited that to HTML and CSS; I include http, linking, etc.

Dave: one concern is, especially if all info is in the JSON, you still need server config for a browser to do anything at all. Hadrien: not true, you can add a link to the html. We should also be able to discover a publication via the manifest. Dave: but how is google going to find it?
Hadrien: you can ask the same thing about the webapp manifest.
Bill McCoy: but the question is not needed for the NavDoc. I dont see the benefit of not using HTML for aggregation. Hadrien: the navdoc is only a toc, not a spine, manifest etc.

Bill: webapps are made of javascript, not html and css. Tzviya: isnt the point of the webapp primarly for launching not discoverability? Hadrien: its a mix.

Dave: my q is: if you create a publication for reasons you dont control, do you have an obligation to create a toc?

Hadrien: other question first: embedding: whether its a different file or embedded. Dave worries about discoverability. I feel that in EPUB 3.0.1, metadata expression is confused. With 3.1 we're in a good direction, we are separating where metadata is expressed, document-level or collection. Resource-specific metadata expressed in HTML. Any content document saying that one is a "home-page" is inaccurate, and saying that a content document is the place where you express things about the publication as a whole is confusing things again. We need to retain clear distinction about the three things.

Hadrien: one reason why EPUB exists is to say, all these resources together make up a publication, and the container. We want to still want to do the first thing in BFF.

Dave: good summary of the problem: the web hasn't considered how to talk about collections of documents.

Tzviya: we're vigorously agreeing. We're getting into details at this point, need to come to some baseline decisions.

Dave: how to make information available to vanilla browser. My ACME-publishing example, I have the JSON manifest embedded in the HTML file.
Hadrien: one problem: include publication information and collection metadata in one specific file. Dave: we can set typeof to Book though.
H: second: frankly you are not going to have control over index.html all the time. Dont think its any easy to use, naming and location is an old-school think to use, better to use a link with link relationships.
Markus: ok to link from content document to json? Hadrien, yes think thats right. In OPDS we have an acquisition link, we rely on two things, the rel itself, and then on the media type. If I am an RS and I have links for EPUB and PDF, I can grab the one I want. For BFF we want an unambiguous media type too. BRings is back to what I was initially saying, I am not [...] by using link instead of embedding the json that works everywhere.

Question of the day: Will schema.org compatible crawlers follow a HTML head link to a JSON file, ever? (JSON or json-ld?) Need answers for both.

Hadrien: ideally you want all of your HTML files to be indexed. You want the information to not just be about the publication, but specifics on the page.

Tzviya: publishers want biblio meta and keywords to be indexed, but not the content itself.

Hadrien: if you embed the manifest in a webpage, the vast majority will be ignored by web engines, the rest will be ignored.

Dave: just want to make sure that publication metadata is available to the web at large. Hadrien: if you want to make the whole book crawlable, then repeating the metadata in each spine file is needed. Second case where you dont want indexing, then whats likely to happen is that the home page of the publication will not contain any content, but reviews and buy buttons.

Dave: then another question; we talked separately about a document identifying itself as being part of the publication, is that not mandatory? Hadrien, it would be better, but not mandatory.
Dave: linking to a publication manifest from a document thats not part of the publication? Will there be a way to distinguish which of the two you mean? Hadrien: not sure its necessary.

Dave: if the answer to the question of the day is yes, then I'm ok to proceed with a separate manifest.

Hadrien: having all the stuff in the same document is a problem for others too [link rel manifest]

Dave: going down this JSON-LD path, how LD'ish do we go? I made a really awkward example. How much do we mix schema.org, how far down do we go?

Hadrien: expressing everything in JSON-LD may not make sense. It less daunting to limit it to the metadata object. Do we need it for everything or just metadata? JSON-LD can provide a context and work along simple JSON. I'm tempted to go down that road, keep a clear easy syntax to write, but have a context.

Dave: alignment with/preference for schema.org might be good in general, given it wide use.
Hadrien: true, but both are possible.

What if the answer is no? Hadrien: the important elements for the manifest from an SEO perspective is limited to 5 elements. In a worst case scenario, duplicate those in the content docs. So frankly the benefit is very small.

Telecon October 20, 2015 at 20UTC

Participants

Dave Cramer (Hachette Book Group)
Matt Garrish
Markus Gylling
Romain Deltour (DAISY)
Daniel Weck (DAISY)
MURATA Makoto [only for 45 minutes] (JEPA)
Hadrien Gardeur (Feedbooks)
Sanders Kleinfeld (O'Reilly)
Fred Chasen (O'Reilly)

Agenda

1. Quick overview of F2F discussion
2. I would like every participant in the group to offer a brief summary of their hopes, dreams, and concerns for the subgroup. Of course, this is voluntary, but I would love to hear from everyone.
3. What does it mean to be web- and browser-friendly? Is it enough to say that EPUB content should be available via HTTP URLs? Are we talking about a file format, an API, or a URL scheme?
4. What might be some possible solutions? I would prefer not to close off any potential areas of exploration at this early stage, as we've already come up with some interesting ideas.
5. What next?

Discussion

Dave: we may do another doodle poll after the time change in November to try and encourage participation from more people
Dave: there was broad consensus at the f2f to continue this work but less consensus on what exactly it should be: web manifestation, api, something else - would like to hear from everyone about what they would like to achieve moving forward
Dave: I would like books to be a first-class citizen of the web - where book is shorthand for all types of publications - want it easier to read books in browsers, make reading systems - concerned about the complexities of epub vis a vis the web
Markus: two pronged:

1) With IDPF hat: What is the best way for EPUB to support the IDPF members who wish to publish in a mode where EPUBs are available both online and in portable form, and allowing their users to read through both RS and browser
2) With DAISY hat: I would like to see epub support print-disabled users to make the choice whether to read in a dedicated reading system or a browser - the browser is often more familiar territory and oftentimes provides a more accessible experience

Makoto: would like to make epub components first class citizens on the web - has proposed a uri scheme - does not want to endanger the ecosystem for epub authoring - has seen a lot of problems in epub authoring and any attempt to change the format may further endanger the ecosystem - has explained his other points in the email chain on the list
Markus: valid point that there is a risk of hurting things we don't want to hurt
Dave: has added salient points from the mailing list discussions below for consideration - please feel free to edit
BillM: from a readium perspective, we need to handle exploded epub files - a zipped version doesn't work well because it requires download - readium already handles exploding but is looking for an interoperable contract between a person putting content on a web server and a reading system consuming it - from an idpf perspective the goal is to support what the membership needs - epub makes portable documents out of web stuff which is what distinguishes it - need to simplify this message as the zipping complicates it - will help people understand and utilize epub more - want epub to be accessible by design instead of an optional layer you may or may not choose to support
Daniel: has a wishlist of items: 1) simplification of metadata and other data structures (levels of indirection, multiple files); 2) no xml - either html5 or json; 3) some kind of object model or data api - what does it mean to access core components - spine and navigation document - fixed layout v. reflowable 4) URI conventions (path referencing publication assets), scheme to address individual resources 5) WYSIWYG authoring, contenteditable (no XHTML5!); 6) core reading experience functionality handled by vanilla web browsers (e.g. document navigation, scrolling, maybe pagination), advanced publication features handled by as-thin-as-possible reading system polyfill (Javascript, CSS, etc.), e.g. Media Overlays

Fred: would like to see the complexity in the tools reduced - zip is an issue - some xml baggage makes working with web tools difficult - but would still like to retain the concept of the complete packaged document
Hadrien: I would like us to move in a direction where we have less complexity - easier to use and more elegant - not a fan of ocf and opf - have been using for years but an opportunity to make better - we could design a very simple document that still retains all the features we have - collection element could be a key to that - we could use collections to do it all - would like a single package document that handles it all - need more powerful linking - we have been underusing linking and hypermedia - by making the link element better we could do a lot more in the future
Matt: I think simplicity is the key; things are very complex now. I want epub to be friendlier on the web; a11y is important… there are lots of opportunities to simplify our structures when we're not bound to legacy. There's ways to make things easier.
Romain: 1) like others, getting closer to the dream of EPUB (or "books" in general) being 1st-class citizen on the open web 2) with browser-friendly EPUB, we'll hopefully see a lot of interesting developments/experiments from the web community, which is to this day largely untapped 3) easier to follow more recent web developments (including a11y) 4) opportunity to clarify some current EPUB constructs, dust-off the 3.0 spec
Sanders: already had a good summary of technical considerations - on a high level looking to make books in browsers a reality - want anything a publisher has in their archive to be published on the web - no biases in serializations or apis - more interested in making the vision a reality - need to make choices that are geared at getting publisher and browser adoption

Dave: question now is where to we go from here - what do we really mean by making a representation of epub web friendly - is that a document, an api a uri scheme
Makoto: does not want to commit the word uri scheme - could be fragment, conventions, etc. - a uri "something"
Dave: agree, we'll be agnostic
BillM: like the idea of a concrete proposal - may be helpful to show the options available - maybe subgroups could develop other proposals and sample content to help get a taste and feel for the alternatives
Dave: the work that Hadrien has done has been really interesting and unearthed questions more rapidly than they might otherwise have been - would like to see at least a couple of fleshed out alternatives
BillM: would like to see a full sample publication and something that could consume it - would help expose if if the JS is trivial
Dave: I can look at a sample of Moby Dick - the epub "hello world"
BillM: Bill and Dave epub zero and Makoto's uri one should also be elaborated on
Markus: did Daniel Glazman have another option in pure html?
BillM: not sure - sounded like an orthogonal issue
Dave: he's proposing a variety of things - a content model api is one he will be presenting at TPAC - also looking at having packaged information in html5 so close to epub zero
BillM: we can't do anything in 3.1 that means waiting on browsers to change

TASK: Dave will try to find out more from Daniel and see if there is a fourth alternative
Markus: how do we ensure that whatever we come up with is something that publishers will embrace - at least those interested in putting things online - how do we avoid being a bunch of techie guys that build in a vacuum - how do we ensure this is easy to create and deploy
Sanders: if the roundtripping is easy - can take any publication in their corpus - that would be huge
Dave: would like a simple program that could automatically transform content
BillM: Tzviya has mentioned that wiley is using epub for journals and handing off to their web people to put on the web - may decide that the packaged content is good enough and their own model already works well enough - but not a failure if people who are already doing their own thing don't change so long as it encourages others to also move forward down this path
Makoto: having two options is a bad thing - even if there isn't loss of information
Markus: even if you can choose to use?
Makoto: Japanese publishers do not want any major changes
Markus: we're not going to change the current zip ecosystem - we also face a split if we do not embrace online publishing - other solutions are going to appear
BillM: the web-friendly serialization is also authoring-friendly - add the right stuff like the json to an arbitrary web site it can be turned into an epub - right now it is a complicated endeavour because of a lack of metadata about the content - could help pull in more content - can't see it as bad for japanese publishers if we make it easier to author content
Makoto: do we have to call it epub 3.1 or another name?
Markus: let's worry about that later - maybe 3.1 is an understatement if we develop something wonderful
BillM: gap between windows 3.0 and 3.1 was huge so there is precedent for small numbering change with big impact
Markus: accessibility guarantee might not be realizable in a vision of epub that could bundle any web content
BillM: right, but if we want epub and html hand-in-hand for future compliance it is necessary
Markus: Daniel G. expressed that he wanted to see everything in html instead of html and json because of unfriendly aspects of json - hand codability, i18n issues - is the trend toward capturing data in html now?
Dave: allows for easy link checking, etc - not sure if there are similar tools for json or if they are widely deployed
BillM: does not agree Daniel's use cases are what should drive epub - html is a better way to represent user-readable strings - onside with json when the data is not meant for the user
Hadrien: everything is based on json these days - not necessarily a big fan of it, but it is the clear winner - the link element in html is not that good - is fairly limited in that it doesn't handle uri templates - does not handle multiple rel values - we could define a much better one - if you look at collection models in json they all define new links elements that are better than what is in html - can discover where and how to interact with a service - need to look into this work to see what they're doing
Markus: doesn't see anything in Hadrien's proposal that requires json for user-facing information

BillM: the fact there are competing ways to represent information in json is problematic - if you hand someone an html file with links it is clear what their purpose is - if it is just data then perhaps json is better but a lot of the data will have user-readable strings on it
Hadrien: for data representations I think html is terrible - would like to see a full opf replacement in html as it is not powerful enough - need to differentiate between what is for user and what is for reading system - are you going to style metadata like the title and author
BillM: you might need ruby in the title
Markus: sure, but only the link to the metadata would be in the json and the metadata in an html content document
BillM: you could link to the metadata straight from the html
Hadrien: even right now we're not using html, we're using xml - if you look at the design of any sdk they use plain text to send that information to the reading system - only the table of contents gets in the way of data/user information, but don't see the navigation document in the spine in the real world - it's not succeeding at being both
BillM: would like the proposal more if the table of contents was in the json - those strings need to be fragments of html in the json to retain i18n, accessibility, etc.
Hadrien: I have updated the sample - the spine does not exist as an element anymore - you can provide a title for it
Dave: the ncx was data presented to the user and it was frustrating because you could not style the titles - lack of italics
Markus: agrees that we can't skimp on i18n, a11y, styling so needs to be html
Hadrien: could be an option to include an html version
Dave: sounds like a progressive enhancement

Note: No call next week

Thoughts from the mailing list

Daniel Glazman:

1. URL scheme to address individual resources in a package
2. way to reach, through the above, the list of resources under a "directory" in a package in package's order
3. use only html5 resources in a package getting rid of proprietary XML dialects that make it so complex and expensive to deal with EPUB
4. Publications Object Model (POM)
5. OSS framework implementing POM and available to all websites and browser add-ons free of charge
6. being browser-friendly also means being editor-in-browser-friendly (think `contenteditable`) and that's not simple task. POM will help a lot here.

Tzviya Siegman

1. My biggest hope/dream for an outcome for this task force is to establish a method to open EPUB on browsers without middleware.

Jean Kaplansky

1. What does it mean to be web and browser friendly: It means my non-techie mother can open any EPUB (both reflowable text and fixed layout) in a web browser and start reading. This means it needs to be as easy to do as watching a video or reading an article on a website.
2. Invite browser engine developing community members (Blink, Webkit, Mozilla, Microsoft) to our conversations to be as inclusive as possible as early as possible.

Makoto Murata

In the F2F, I said that we need a scope rather than guiding principles.I would like to propose the scope of this subgroup. I also think that "manifestation" should be dropped from the subgroup name, since it sounds like one particular mechanism for the problem to me.

Scope

This subgroup aims to make EPUB components first-class citizens on the Web: they can be retrieved without retrieving entire EPUB publications. It thus becomes possible for thin layers on top of browser engines to retrieve EPUB components and provide rendering and interactions. This first-class citizenship of EPUB components is also beneficial to search engines, previewers, OPDS-based systems, and so forth.

Note: Representations of EPUB publications on the server side are outside the scope. We do not tie the hands of those who implement server-side programs.

Note: Authoring formats are outside the scope. We do not intend to change the authoring ecosystem of EPUB.

Telecon October 1, 2015 at 19UTC

Participants

Dave Cramer (Hachette)
Tzviya Siegman (Wiley)
Romain Deltour (DAISY)
Matt Garrish (Canada)

Sanders Kleinfeld (O'Reilly)
Hadrien Gardeur (Feedbooks)
Markus Gylling (IDPF & DAISY)

Agenda

1. Discuss background and scope
2. Use Cases
3. Basic Requirements
4. Ideas
5. What next?

Discussion

scribe: markus

Dave: EPUB is in many ways a bundle of web content, but if you try to open an epub in a web browser it doesnt do anything... there is interest in making EPUB more browser and server friendly. As you can see in the introduction, this was not only brought up in 3.1 feature requests, but was a deferred topic in 3.0 in 2011 (see introduction below).

Goals are to enable EPUB reading in the browser, while maintaining the benefits of portability. We have now written a first set of requirements (below). It does seem possible to come up with something like this. What do people have to say? We have had some proposals, experimenting with JSON approaches (below).

Markus: It w/b really nice if this group could feel happy about the use case and requirements before the f2f. There's a lot of people who don't understand why we want to do this at all. Agreeing on the high level scope would be good to avoid ending up in endless loops. I'd like use to focus on use cases, requirement and goals, then only move on to syntax proposals.
Your experiments (Dave, Hadrien) have proven this is technically feasible.
Looking at the requirements, does the IDPF membership agrees this is something to do?

Dave: yes, we can work on use cases and requirements
Markus: some the req + use case come from the w3c wiki.

Dave. Going through use cases (below).
On U1. Ties in to the whole idea of previews. Right now happens through RS themselves. Marketing may not happen in the confines of a bookstore. Here we have ebook content directly on the web. Tzviya: I added a link to the previews spec.
Romain: we are talking about EPUB without ZIP, is it in scope to talk about packaging? Markus, no not in this revision. Dave: for some of this we may want to keep way in the back of our minds that there may be another goal further on. The EPUB-WEB vision; this is just a first step for that larger vision. Tzviya: we will have a meeting with the TAG in end October. Romain: while talking

about scope: is it in scope to talk about an API to access all the files within the publication. Tzviya: Daniel Glazman is working on this. Dave: another thing we should be aware of. Markus: primary deliverable would be a document format, not an API.

On P1. From publishers perspective we want something that works in both senses, as a publication on the web and a publication in an EPUB RS.

On U2. Sanders: the URI itself is also pretty amazing from a publisher's perspective, ability to point users at the book. Tzviya: also from an identification perspective. Once we have a URI it might be a solved problem. Dave: would the RS create a zipped epub here? Markus, maybe, but not necessarily, the RS can choose how to cache etc.

Romain: That's why I asked about APIs, how to sync between to two versions. The physical storage of the content might just be an implementation detail. An API might be a facade to the implementation.

Markus: are we ok with the three first ones (some editing ensues).

U3-U7 Some of these are just basic requirements for portability.
U6. I would keep U6 here. I think that being able to progressively load the ebook has a lot do with this manifestation. The way we describe the ebook in a new format can be impacted by progressive/streaming.
U7. Whats new here? Dave: a webby way to do this. Markus: RS and server would have to use a public API instead of vendor-specific solution.

Romain. Added **U8**. Sanders: thats a requirement not a use case? Hadrien: what if as a user I could make a publication out of resources on the web, here you go. Thats the use case for users. For publishers, its to used shared resources across multiple epubs.

Introduction

Background

From EPUB 3.1 Feature requests (2015):

> **Consider reviving the OCF File System Container as a server-oriented manifestation**
> The OCF 2.0.1 File System Container was removed in OCF 3.0, with a promise to resume work on it at some future point.

> A revived version could be made into a browser- and server-friendly manifestation of EPUB (e.g. a set of content documents linked via a JSON-defined spine, etc). This new feature could be seen as a first baby-step towards realizing the Open Web Platform alignment vision as laid out in the EPUB-WEB white paper.

From EPUB 3 Changes Filesystem Container (2011):

> " [...] The Working Group understands that networked Publications will be increasingly important, and expects future work to include development of robust interoperable conformance definitions for distributed EPUB Publications based on emerging content publisher and Reading System requirements. [...]"

Basic use cases

U1. A User accesses a publisher's website (which may or may not be behind a paywall) and selects a book to read. The reading session initiates directly in the User's browser (now being consumed as an ebook using the new server/browser friendly manifestation). The User then elects to continue the reading in her favorite EPUB Reading System, and therefore uses the website interface to request a downloadable/portable version of the book (which is the classic zip manifestation).
Note connection to http://www.idpf.org/epub/previews/.

P1. A publisher wants to enable U1 above, without having to create and maintain two separate versions of the same publication.

U2. The publication has a URI. Same as U1, but instead of requesting a portable version for download from the website, the User points her favorite Reading System to the base URI of the online book, and the RS takes care of portabilizing the content. The publisher may also use the URI for purposes such as identification and marketing.

U6. A publication may be very large, e.g., due to a large amount of embedded data. The user does not want to wait for everything to be accessed before being able to consume the main content.

U7. Reader pauses reading on RS at the end of chapter 7. Reader resumes reading online and expects system to maintain position of reading when she opens EPUB in browser.

U8. A user wants to create a publication by aggregating public online resource available in the Web.

P8. A publisher wants to share resources across different publications.

Alternative Use Cases (phrased as user stories)

1. As a user, I want to find a book on the web and start reading it right away, in my browser. Later I may want to download it to my favorite EPUB reading system.
2. As a publisher, I want my books to easily discoverable and readable on the web.
3. As an application developer, I want to display ebook content in a browser without having to parse multiple custom XML vocabularies.

Requirements.
On 1. [discussion on scope]
On 4. Tzviya: navigation document, spine and manifest. Do we need manifest? For prefetching its interesting, and for validating content when putting back to classic.

Basic requirements

1. The new manifestation must be *consumable by end users in both vanilla browsers and EPUB Reading Systems*. The feature sets provided by a non-plugin-enhanced browser and a dedicated RS may not be exactly the same, but basic reading through the ability to access all the content of the publication (by navigation and spine traversal) must work in both.
2. The new manifestation should be servable by *web servers without any built-in need need for pre-manipulation* of the content (unpacking, transforming, etc).
3. *Roundtrippability by transformation* between the new manifestation and the classic EPUB ZIP manifestation must be possible. The first version of SBFM may not achieve this goal completely, but it remains the ultimate goal to have lossless roundtrippability.
4. The two manifestations must *share critical constructs and concepts* to the maximum extent possible (list to be produced). (Examples: both use the Navigation Document, both share the same requirements and terms/properties for metadata (but not necessarily the same syntax to express them), both has the concept of “spine”, etc.
5. Content should be storable(?) in distributed servers. Secured with CORS?

Scope

[add something about not necessarily translating all satellite features in this round]

Issues

- Compatibility and coexistence
    - alignment with other standards
- Relationship to EPUB+WEB

- Relationship to Web Application Manifest
- Container vs Package

Ideas

- Unzipped version of EPUB3; just a folder with required structure and files
- JSON versions of existing container.xml and package files
- Can we start to align with web application manifest spec?
- Are there opportunities to simplify the folder structure?
- zip/unzip API and local storage as part of OWP
- URIs for accessing resources in zipped EPUB publications

JSON Container

[probably not necessary]

JSON Package

[removed as Hadrien's proposal is much better]

Proposal (Hadrien)

First thoughts

First of all, here's a quick list of things I have in mind:

- there's no need for separate container and package files
- JSON serialization is probably preferred
- resources can be all tied together through a link to the new package document
- this is an opportunity to rethink what we need in the package

Regarding linking, this is very straightforward, any HTML document can link back to a package document in order to indicate that it's actually part of an EPUB document. This can be done in two different ways:

- a link in the HTML itself
- a link in the HTTP headers

In both cases the package document can be identified by:

- its media type (application/vnd.epub.package+json?)
- and a relationship (A new one? An existing one like "contents"?)

For the JSON document itself, based on the current state of packages in EPUB 3.1, we could probably boil it down to a very small list of elements:

- everything in the package should be a collection (rendition, manifest, package itself…)
- collections are made of metadata and links (as defined in 3.0.1)
- we'll just need to define a JSON element for "metadata" and another one for "link"

Regarding collections:

- there's no need to define a "collection" element
- use collection roles as the key in the JSON key/pair
- no need to define a "package" role, we'll assume that the document describes a "package" collection directly

Regarding links:

- we need to extend the current link elements to also support URI templates in addition to URIs

- we absolutely need media type and relationship (hypermedia)
- we should use links instead of specialized elements to list items in spine and manifest (already the case for other modules using collections)

Regarding metadata:

- do we need a generic element? Maybe.
- let's get rid of ID/IDref, doesn't make any sense in JSON (I would argue that they don't make sense in XML either)

Example

JSON Package

```
{
  "metadata": {
    "title": {
      "value": "The Matchmaker",
      "file-as": "Matchmaker, The"
    },
    "creator": {
      "value": "Elin Hilderbrand",
      "file-as": "Hilderbrand, Elin"
    },
    "identifier": {
      "type": "unique-identifier",
      "value": "9780316329026"
    },
    "modified": "2011-01-01T12:00:00Z",
    "language": ["en"],
    "link": {
      "rel": "record",
      "href": "publication.xml"
    }
  },

  "link": [
    {
      "rel": "search",
      "href": "/search?q={query}",
      "type": "text/html",
      "templated": "true"
    },
    {
```

```
      "rel": "alternate",
      "href": "9780316329026.epub",
      "type": "application/epub+zip"
    }
  ],

  "rendition": {
    "manifest": {
      "link": [
        {
          "rel": "cover",
          "href": "images/9780316329026.jpg",
          "type": "image/jpeg"
        },
        {
          "rel": "nav",
          "type": "text/html",
          "href": "toc.html"
        }
      ]
    },

    "link": [
      {
        "href": "cover.html",
        "type": "text/html",
        "title": "Cover Page",
        "properties": "rendition:layout-pre-paginated"
      },
      {
        "href": "http://example.org/chapter1.html",
        "type": "text/html",
        "title": "Chapter 1 - The Beginning"
      },
      {
        "href": "http://example.org/chapter2.html",
        "type": "text/html",
        "title": "Chapter 2 - The End"
      }
    ]
  },

  "preview": {
    "link": [ {"href": "cover.html"}, {"href":
"http://example.org/chapter1.html"}]
  },
}
```

Example of a link
In HTML:
<link href="package.json" rel="package" type="application/vnd.epub.package+json" />

In HTTP headers:
Link: <package.json>; rel="package"; type="application/vnd.epub.package+json"

The link element

First of all a few notes:

- both "item" and "itemref" in "classic OPF" have a bunch of attributes that do not currently exist for "link"
- some "properties" are better suited as "rel" (cover or navigation for instance) than "properties"
- we might still need "properties" for link, but let's figure out if it can replace more than one of the existing attributes
- I want to get rid of "linear" in spine, I think it's useless and both navigation documents and content documents should be allowed to link to documents both in and outside the scope of the package
- we probably want to extend "link" with media specific attributes, to be more future proof
- I'd also like clear rules on which "properties" and "rel" should be used in the "manifest" collection and which should be used in the "spine" collection (ideally, none for the "spine")

Here's a first list of key/pair that could be associated to the link object:

Name	**Value**	**Format/data type**	**Required?**
`href`	Link location	URI or URI Template	Yes
`title`	Title of the link	String	No
`type`	Expected MIME media type value for the external resources	MIME media type	No, but highly recommended
`rel`	Relationship	Array of relationships (point to registry) or URIs (extensions)	No
`templated`	Indicates that the href is a URI Template	Boolean, default value is "false"	No

`properties`	Properties associated to the linked resource	Array of properties (point to registry) or URIs (extensions)	No

Among the attributes included in "classic OPF" that are excluded for now (they should be discussed in comments):

- fallback
- media-overlay
- linear
- refines

For both fallback and media-overlay, the use of inlined links could be a very good option.

For instance, for fallbacks we could use a link@rel="alternate" link inlined in the manifest link for an image in spine:

```
"link": {
  "href": "page1.jpg",
  "type": "image/jpeg",
  "link": {"href": "page1.html", "type": "text/html", "rel": "alternate"}
}
```

On a side note, it's unclear to me whether it is best to provide fallbacks and media overlay in the manifest or in the "spine".

Among the media specific attributes that could also be useful on a link :

- length (in octets)
- height (in px)
- width (in px)
- duration (in seconds)
- bitrate
- samplingrate

Another direction worth considering is how powerful we want the link element to be?
Mike Admundsen's H-Factors (http://amundsen.com/hypermedia/hfactor/) are probably one of the best way to evaluate this current proposal.
The current element has the following properties:

- [LE] Embedding links
- [LO] Outbound links
- [LT] Templated queries

- [LN] Non-Idempotent updates
- [LI] Idempotent updates
- [CL] Control data for links

It makes it better than the equivalent in Atom due to its support for [LT] Templated queries, and also better at Control data for links [CL].
HTML as a whole supports more H-Factors but it needs many different elements to achieve the same goal.

Support for the remaining three H-Factors should also be in our list of considerations as we layer additional services on top of our package document:

- [CR] Control data for read requests
- [CU] Control data for update requests
- [CM] Control data for interface methods

The (non-existing) collection element

Since we assume that everything is a collection, the only real manifestation of a collection is through the use of roles.

Basically, if it's not "metadata" or "link", then you can assume that it's a collection and that the key is the role.

That said, the collection element in "classic OPF" had two attributes that might have to be moved elsewhere:

- xml:lang
- dir

Both attributes could probably be defined in the metadata element of a collection, although there's no real equivalent for xml:lang in JSON that I can think of.

More generally, I think this raises a few questions about two different things :

- metadata describing a publication or a group of resources
- indications to the reading system (rendition properties, dir, page-progression-direction)

Should those two be considered completely different with their own elements? Should we group the RS specific stuff somehow?

Some areas left to explore (in an update)

- for content documents, do we need a way to include HTML/CSS that should not be displayed if the resource is displayed as part of a publication (vs as a standalone resource)?

Bonus features

OPDS

OPDS (Open Publication Distribution System) is a protocol to browse, search and acquire publications, mostly used to distribute EPUB files (but neutral regarding content).
It is based on Atom, and relies mostly on three things:

- links (yeah, hypermedia!)
- a collection model (feed/entry in Atom)
- some basic metadata

The new proposed model (everything is a collection) is basically generic enough to be fully compatible with OPDS. We could imagine a future OPDS 2.0 revision that would define a JSON serialization fully compatible with this new JSON package document (including metadata).

It would be super easy to build RS capable of support both at the same time, with minimal code needed to handle OPDS on top of EPUB-(Whatever-we-call-it).

Example of a navigation feed

```
{
  "metadata": {
    "title": "eBook Store"
  },

  "link": {"rel": "http://opds-spec.org/shelf", "href": "/shelf"},

  "navigation": {
    "link": [
      {"title": "New Releases", "href": "/new"},
      {"title": "Best Selling", "href": "/top"},
      {"title": "Categories", "href": "/categories"}
    ]
  }
}
```

Example of an acquisition feed

```
{
  "metadata": {
    "title": "New Releases"
  },

  "link": {"rel": "next", "href": "/?page=2"},

  "acquisition": {
    "publication": [
      {
        "metadata": {
          "title": "The Girl in the Spider's Web",
          "author": "David Lagercrantz",
          "identifier": "urn:isbn:9780385354288",
          "category": {
            "label": "Thrillers / Technological",
            "code": "FIC036000",
            "scheme": "https://www.bisg.org/bisac"
          }
        },
        "link": [
          {
            "rel": "http://opds-spec.org/image",
            "href": "/cover.jpg",
            "type": "image/jpeg"
          },
          {
            "rel": "http://opds-spec.org/acquisition/buy",
            "href": "/buy",
            "type": "application/epub+zip",
            "opds:price": {
              "currency": "USD",
              "value": "13.99"
            }
          }
        ]
      },
      {
        …
      }
    ]
  }
}
```

App Manifest

I also believe that this would be a much more flexible model than the proposed W3C specification: http://www.w3.org/TR/appmanifest/

It's super easy to serialize the current W3C proposal into our proposed model (not true the other way around). To prove it, I've used the first example from the W3C spec and rewrote it using our model :

```
{
  "metadata": {
    "lang": "en",
    "name": "Super Racer 2000",
    "short_name": "Racer2K",
    "display": "fullscreen",
    "orientation": "landscape",
    "theme_color": "aliceblue",
    "background_color": "red"
  },


  "link": [
    { "rel": "scope", "href": "/racer/" },
    { "rel": "start", "href": "/racer/start.html" }
  ],

  "icon": {
    "link": [
      {
        "href": "icon/lowres",
        "type": "image/webp",
        "height": "64",
        "width": "64"
      },
      {
        "href": "icon/hd_hi",
        "height": "128",
        "width": "128"
      }
    ]
  },

  "splash": {
    "link": [
      {
        "href": "icon/hd_small",
        "height": "1334",
        "width": "750"
      },
      {
        "href": "icon/hd_hi",
        "height": "1920",
        "width": "1080"
```

```
        }
      ]
    }
}
```

Examples

[Rough sample file of Moby Dick]

JSON package with more stuff:

```
{
   "metadata": {
      "title": "Moby Dick",
      "identifier": {
         "type": "unique-identifier",
         "value": "9780000000000",
         "modified": "2015-09-29T17:00:00Z"
      },
      "type": "preview",
      "language": "en",
      "link": [
         {
            "rel": "schema.org-record",
            "href": "metadata.json"
         },
         {
            "rel": "acquire",
            "href": "http://example.org/book/9780000000000.atom",
            "media-type": "application/atom+xml;type=entry;profile=opds-catalog"
         },
         {
            "rel": "encrypt",
            "href": "encryption.xml",
            "media-type": "application/xenc+xml"
         },
         {
            "rel": "sign",
            "href": "signature.xml",
```

```
            "media-type": "application/xml"
        }
    ],
    "rendition": {
        "flow": "paginated",
        "layout": "reflowable",
        "spread": "auto",
        "orientation": "auto"
    }
},
"manifest": {
    "link": [
        {
            "type": "text/html",
            "href": "html/cover.xhtml"
        },
        {
            "rel": "cover",
            "href": "images/9780000000000.jpg",
            "type": "image/jpeg"
        },
        {
            "rel": "nav",
            "type": "text/html",
            "href": "html/toc.xhtml"
        },
        {
            "type": "text/html",
            "href": "html/title-page.html"
        },
        {
            "type": "text/html",
            "href": "html/copyright.html"
        },
        {
            "type": "text/html",
            "href": "html/introduction.html"
        },
        {
            "type": "text/html",
            "href": "html/epigraph.html"
        },
        {
```

```
            "type": "text/html",
            "href": "html/c001.html"
        },
        {
            "type": "text/html",
            "href": "html/c002.html"
        },
        {
            "type": "text/css",
            "href": "css/mobydick.css"
        }
    ]
},
"spine": {
    "link": [
        {
            "href": "cover.xhtml"
        },
        {
            "href": "html/title-page.html"
        },
        {
            "href": "html/copyright.html"
        },
        {
            "href": "html/introduction.html"
        },
        {
            "href": "html/epigraph.html"
        },
        {
            "href": "html/c001.html"
        },
        {
            "href": "html/c002.html"
        }
    ]
},
"rendition": {
    "label": "Go with the Flow",
    "media": "",
    "identifier": {
        "type": "unique-identifier",
```

```
            "value": "9780000000000",
            "modified": "2015-09-30T17:00:00Z"
        },
        "link": [
            {
                "type": "text/html",
                "href": "html/chapter001.xhtml"
            },
            {
                "type": "text/html",
                "href": "html/chapter002.xhtml"
            },
        ]
    },

    "rendition": {
        "label": "Fixed Layout is For the Birds",
        "media": "",
        "identifier": {
            "type": "unique-identifier",
            "value": "9780000000000",
            "modified": "2015-09-30T19:00:00Z"
        },

        "link": [
            {
                "type": "text/html",
                "href": "html/page001.xhtml"
            },
            {
                "type": "text/html",
                "href": "html/page002.xhtml"
            },
        ]
    },

}
```

Using renditions array:


```
{
   "title": "Moby Dick",
   "identifier": {
      "type": "unique-identifier",
      "value": "9780000000001",
      "modified": "2015-09-29T17:00:00Z"
   },
   "rendition": [
      {
         "metadata": {
            "identifier": {
               "type": "unique-identifier",
               "value": "9780000000002",
               "modified": "2015-09-30T17:00:00Z"
            },
            "rendition": {
               "layout": "reflowable"
            },
            "selection": {
               "label": "Go with the Flow",
               "layout": "reflowable"
            }
         },
         "manifest": {
            "link": [
               {
                  "type": "text/html",
                  "href": "html/cover.xhtml"
               },
               {
                  "rel": "cover",
                  "href": "images/9780000000000.jpg",
                  "type": "image/jpeg"
               }
            ]
         },
         "link": [
            {
               "type": "text/html",
```

```
                "href": "html/chapter001.xhtml"
            },
            {
                "type": "text/html",
                "href": "html/chapter002.xhtml"
            }
        ]
    },

    {
        "metadata": {
            ...
        },
        "manifest": {
            "link": [
                ...
            ]
        },
        "link": [
            ...
        ]
    }
  ]
}
```