Shorenstein Center on Media, Politics and Public Policy

Discussion Paper Series

#D-95, July 2015

The Rise, Fall, and Possible Rise of Open News Platforms

The Twisty Path towards a Net Ecosystem That Makes News More Discoverable, Reusable, and Relevant

by David Weinberger
Joan Shorenstein Fellow, Spring 2015
Senior Researcher at Harvard's Berkman Center

HARVARD Kennedy School
SHORENSTEIN CENTER
on Media, Politics and Public Policy

Table of Contents

A Failed Vision in Common

“It was a success in every dimension except the one we thought it would be.”

That’s Daniel Jacobson’s tweet-length summary of his experience opening up National Public Radio’s vast resources to any software developer anywhere in the world who had a good – or bad – idea for an app that could use it. It’s a tweet that could have been sent by other pioneers of open news platforms, including *The New York Times* and *The Guardian*, with the only difference that the *Times*' expectations were lower.

The vision was idealistic. Imagine any Internet site, service, or application that had even a passing interest in saying something reliable about what’s happening easily getting the news it needed. Imagine a world in which news has an ever wider impact as clever software developers put it to work in ways the news organization itself would not have come up with, and could not have afforded to implement. The vision was humble as well, for it assumed that some of the best ideas for putting a news organization’s stories to use would not come from that news organization itself.

So, in July 2008, NPR opened up its resources to external developers, followed by *The New York Times* in October of that year, and *The Guardian* in March 2009.[1] They launched software platforms that used the most up-to-date industry standards. The platforms were constructed to be robust and reliable. Documentation and tools introduced external developers to the bounty. When the platforms were ready, press releases were emitted. The occasional “hackathon” was held at which developers were given the space, time, and pizza required to slap together apps over the course of a weekend or even a single day. There was a flurry of enthusiastic articles in the technical press and conference presentations. These platforms embodied the Internet's best values.

And then the air went out of the vision. The Renaissance of News Apps became just another failed dream of cyber utopians.

Yet, the platform architects at NPR, *The Guardian* and *The New York Times* still use words like “transformative” and “revolutionary.” But it’s not the world that’s been transformed. It's the news organizations themselves.

If these major news media outlets went down the platform path in large part to enable an Internet vision of an open information ecosystem, only to discover that these platforms are justified by straightforward business benefits, are there less utopian ways of achieving the utopian dream? Might there be ways of creating a far more open news environment on the Net by following business imperatives?

The Triumph of Abstraction

There is no single reason why 2008 was the year that three of the most important news organizations developed open platforms.

Facebook had done something similar in August 2006[2] when it permitted external developers to build applications that integrated with Facebook’s data and services, creating everything from games that run within Facebook to apps that track users’ vitamin intake.[3] Facebook's success was influential.

Or perhaps having to bring news to the Internet in all the ways the Net demands simply broke the back of the news organizations’ existing systems.

It was a long time coming. As early as 2001, NPR’s Dan Jacobson had headed down the platform path, prompted by thinking, “How many more times are we going to redesign the site?”

By then, NPR.org was no longer just an adjunct to NPR’s on-air programming. It was crucial that the site keep up with the cascade of new content types and services users demanded. For example, RSS[4] was becoming the accepted standard for open content syndication, but RSS was undergoing rapid evolution, and soon spawned Atom, a competing standard. Then podcast versions of on-air stories become de rigueur,[5] requiring new extensions to NPR program pages. In just a few years, the rise of smartphones would necessitate producing and syncing parallel sites where users expected to swipe, tap, pinch, and type with their

thumbs on tiny fake keyboards displayed on small screens that can rotate from landscape to portrait.

The content management systems (CMS) used by all three news organizations were beginning to fray under the pressure. They were designed to manage the workflow of the production of content, and then to output it in a few relatively stable formats: a print version, an audio file, a website. But the Web ecosystem never slept and was placing a new premium on flexible and rapidly changing outputs. That required an approach that Jacobson and his colleague Rob Holt early on expressed as the COPE approach: Create Once, Publish Everywhere.[6]

Graham Tackley, a leader of the technical team at *The Guardian*, understates it when he says that by 2007, "We were beginning to face a number of scalability issues." The demands were increasing, and the riskiness of experimenting with solutions was rising: the team at *The Guardian* was worried that if something they tried went terribly wrong, it might break the CMS and slow down or even disable the organization's existing sites. Reaction to that would be swift and chilling.

Concerns like these came to a head at NPR in 2007 when the network wanted to launch a cross-program site dedicated to music to be called, reasonably, NPR Music.[7] "It was a very different user interface, different presentation, different assets associated with each piece," says Jacobson. It would have been relatively easy to extend the CMS to accommodate the new needs, but how about for the next new site? And the next? And the next? "It would be better if we could create an abstraction layer that the site could draw from."

An abstraction layer hides the details of how a task is accomplished. Evan Sandhaus – director for search, archives, and semantics for the *Times* – explains it using a restaurant metaphor. A customer who wants a dish doesn't have to go into the kitchen and tell the chef to grate an ounce of cheddar cheese, beat two eggs with a tablespoon of milk, heat up a 9 inch pan, etc. Instead, the customer can simply ask for the cheese omelet that's on the menu. The menu provides a layer of abstraction.

This makes life easier for the customer, but the systems benefit is at least as important. The restaurant can hire a new cook, replace the stove, switch food

suppliers, or tinker with the recipe, and the customer can continue ordering the omelet without knowing or caring about what exactly is going on in the kitchen. The whole place runs more smoothly.

Rajiv Pant, the *New York Times* chief technology officer, gives a concrete example. Subscribers have to register to make it through the *Times*' pay wall, so the *Times* built a behind-the-scenes registration system that serves all the for-pay services the *Times* offers, on all the devices those services run on. Creating a new user registration system for each new service would be a waste of resources, and having to use a new system for each new service would frustrate subscribers.

So in 2007 both NPR and the *Times* built APIs — application programming interfaces — to serve as their abstraction layers.[8] The API stands between an application and the information that application needs. A command like "Check the password the user just entered" is like "omelet" on a menu. The developer doesn't care how the user registration system actually does that check. The people maintaining the registration system could change which database they're using — which in fact was one of the considerations that drove NPR to put an abstraction layer in place — and the app developer wouldn't have to change a single line of code. The "Check this password" command to the API will continue to work without a hiccup.

Matt McAlister, who was in charge of *The Guardian*'s API development, points to exactly the same benefits of putting an API on top of their CMS. Around 2007 their CMS was "very, very big" and complicated. His team wanted to ensure that the burgeoning applications and services would be stable, robust and very fast. They needed an API.

APIs have been around for decades,[9]but modern APIs tend to have two distinctive features.

First, they don't talk directly to the CMS (or other database application) but to a full text index of the content of the CMS. A full text index serves the same purpose as the index at the back of a book: rather than requiring the reader to skim through all the text every time she wants to find where the book talks about some topic, the book's index provides a "pre-computed" tool for rapidly looking

up words. An online full text index lets the API operate at Google-like speeds and adds some flexibility in the sorts of requests it can field.

But it is from the second distinctive characteristic of modern APIs that the vision of an open news ecosystem arose: The APIs adopted by NPR, the *Times* and *The Guardian*, as well as many other content-producing organizations, use the Web as the medium for sending requests for data and for receiving data back. This means that you can think of a typical modern API as a website designed to be accessed not by humans looking for pages, but by computer applications looking for data. So, without much work, the set of clever developers doing good things with news media content could be expanded to any developer with an Internet connection.

Why not?

The Open Impulse

NPR's Jacobson says "We built an API for the music site, and then thought about having all our news sites draw from the same API." That thought led immediately to another: "If we have this API, it's a relatively small step to get it to be public."

It would hardly be rocket science. They'd have to provide an automated way of issuing ID codes, called "keys," that developers can insert in their API requests so that there is a modicum of protection against miscreants. They might also want to set reasonable limits on how many requests per second and per day would be permitted. And they'd have to decide which information they would give access to. Licensing issues and concerns about constraining misuse can make content problematic, but information about the content — metadata — generally is not. Jacobson says that the biggest technical challenge was "making sure we had content tagged with correct rights information and then have the API be smart enough to prune or modify the content that hit certain rights issues."[10]

But why go to even that much trouble?

The three organizations had overlapping reasons for putting up public APIs.

"NPR has a public service mission," explains Jacobson. Graham Tackley, *The Guardian*'s director of architecture, also points to the corporate mission, saying that the creation of APIs in the first place was prompted by the organization's commitment to the sort of collaborative sharing the Web enables. "Our editor-in-chief Alan Rusbridger absolutely believes that…engaging with other people on the Web is an important part of being a news organization."

Rajiv Pant at the *Times* makes the same point: creating public APIs "lets others build and benefit from what we do," which is "consistent with the *New York Times* mission — it's good not only for our company but for society overall." But Pant's hopes were never high: "We did external APIs in part because it's the right thing to do," without expecting it would spur all that much development.

By 2008 the *Times* had been using APIs internally for over a year. But Tackley says "The [*Guardian*'s] initial creation of the content API was about enabling other people to access our content and do interesting things with it." Likewise at NPR, Daniel Jacobson recalls, "If you look at the article and interviews from the time, the quotes were about us being really excited about the magic that would be developed by people in their garages."

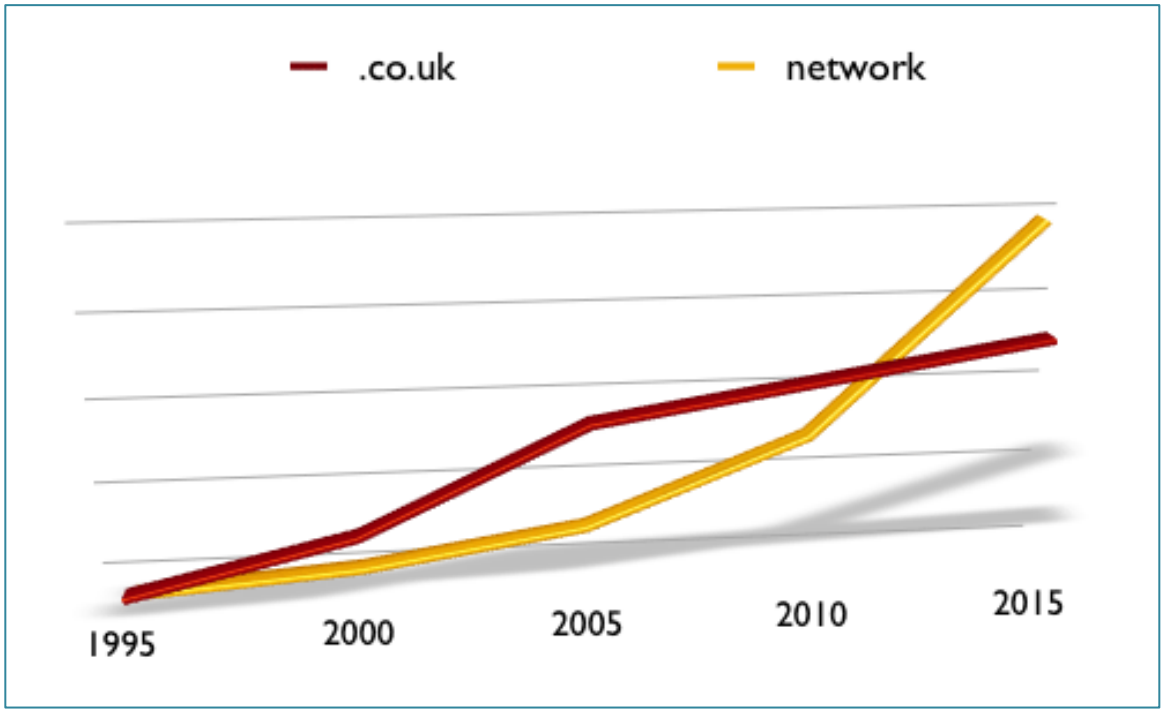
Behind this a set of cultural values was at work. In July 2008, Wired.com ran an interview with Derek Gottfrid of the *Times* and NPR's Jacobson in which Gottfrid makes this explicit:

> We're geared to whoever is going to find the content interesting. Anyone that's interested in it, we're interested in making it accessible and having them use it. This isn't something that's driven off of market research or anything like that. This is fulfilling a basic gut-level instinct that this is how the internet works.[11]

McAlister had come to *The Guardian* from Yahoo! and the Silicon Valley where building "open APIs seemed like the right thing to do." But he found that it wasn't as obvious to a news organization. *The Guardian* under Rusbridger's leadership was willing to listen, "but there needed to be at least an understandable benefit to

it. That took a lot of work," he says, "and a lot of slides showing the value of openness.

One slide McAlister used in many meetings laid out the most basic dynamic: "While we were in an unsustainable decline in print, there was strong growth in digital, but not enough to put *The Guardian* or any media organization back into a position as strong as before the decline of print."



McAlister's slide[12]

Therefore, McAlister urged, "media organizations needed to extend beyond their domain to grow, and to be present in all the places that readers are." Some of those extensions would be built by *The Guardian*, using their API as their way to connect their own apps to their own data. But some would be built by developers outside of *The Guardian*. "The open innovation pitch was an important part of this," he says, "but a little harder to justify."

As a result, all three organizations opened up public APIs within months of each other. Javaun Moradi, a product manager at NPR at the time, recalls: "The

2008 conventional wisdom on APIs was that it would let a thousand flowers bloom. People in our community would rise up and build mashups that we couldn't even imagine."

Sure enough, the community of developers responded. Wanly.

Moradi remembers a visual app that displayed a "globe that spun that popped up stories from around the world." People wrote some syndication apps that made it easier to distribute NPR stories. McAlister recalls that there were about fifty apps from independent developers in *The Guardian*'s gallery. For example, GoalRun.com integrated *Guardian* content into its niche football websites. There were some interesting political sites, too, he says, including one that used *Guardian* political data "to create an index to understand how powerful your vote is, given the strength of the political parties in your area." Joe Fiore, Manager of Technology at the *Times*, recalls an app a high school history teacher wrote that's like a fantasy football league except you draft countries, not players. "Any time one of your players is mentioned in the *New York Times*, you get a point."[13] The popular app IFTTT[14] ("If This, Then That") still uses the *Times*' public APIs to let users know if there's an article about a sports team or a new book by a favorite author. "People ask me why public APIs, and that's it right there: they didn't need permission."

It was thrilling when developers built permission-free apps. Moradi says, "We got a call saying there's an NPR app in the iTunes store that isn't ours. We used that gentleman's name for years as a shorthand for when someone disrupts you with the technology you invented." He adds, "It was meant as the highest possible praise."[15]

So, a lot of excitement and some excellent examples. But, after an initial spurt, growth flattened. Development of the public APIs and of the supporting documentation and tools slowed at all three organizations.

But that is far from the end of the story. Jacobson points to the iTunes app that Moradi referred to: "We learned about it because all of a sudden we saw a lot of traffic coming from a single app. You could navigate through our topics and programs to listen to our stories. I think the app was downloaded somewhere

around 500,000 times. It told us that the iPhone was a really important space. It gave us a kick in the pants" — a prod for NPR itself to develop for the iPhone.

"The public developers have been fine historically," Jacobson concludes, but "it wasn't transformative in the way we expected. It was hugely successful in letting us see the internal uses."

Why the Failure to Thrive?

The pioneers at these three news organizations point to a number of possible reasons why these APIs failed to spark a renaissance of apps and create an open ecosystem of news innovation.

McAlister at *The Guardian* attributes much of it to the failure of the news organizations to help partners and developers make money by using the open APIs, either directly, perhaps by syndicating ads through the platform, or indirectly by "giving partners greater exposure to users and other partners, etc."[16]

Although one of the great benefits, and even joys, of a public API is that developers don't have to ask permission to use it, it turns out that developers with no business relationship with the API's owner probably won't build anything except toys. Joe Fiore points out that for a developer to invest in building a serious app, she or he has to be confident not only in the API, but that the company providing the API is committed to maintaining it.

The Guardian's Tackley says that legal issues made their API "harder to work with than we'd hoped." The lack of legal clarity meant that *The Guardian* had to be overly cautious about what they made available publicly.

Even after that was cleared up sufficiently, "we had to have a robust process for removing content about which legal issues develop," Tackley continues. "If we've published something incorrect or libelous, we have an obligation to take control over it." That means that if you write an app that distributes content from *The Guardian,* developers have to "check every twenty-four hours to see whether

they're still able to use our content." Even if you automate that, it puts you in an uncomfortable position at the least.

Evan Sandhaus at *The New York Times* points to a related issue: ensuring proper attribution of protected content. Once an article leaves the creator's site, the person who wants to repost it can decide to strip out the author, the name of the publication, and the link back to the original site.

Javaun Moradi at NPR thinks that it was a case of thinking if you build it, they will come. "APIs are great for lightweight prototyping," he says, "but it takes more than technology to build a great user product. For anything more serious you need insight about your business, insight about your customers, and full-time work to get the experience right." In short, there's a long distance between the fun app you throw together at a hackathon and a sustainable, successful product.

Moradi suggests that this is not simply the external developers' problem. The creators of the API "shouldn't just release for the sake of releasing. You have to know your target audience" so you can provide them with the information and content they need. He sums it up in a way that goes against the utopian "build it and they will come … and they will then create wonders" philosophy: "If you don't know why something is valuable, then probably no one else will."

Then there are cultural issues. McAlister says, "A lot of institutions that had created open APIs became scared of what they'd unleashed." An open API cedes control to unknown developers. In the early 2000s, one of the arguments within the BBC against opening up its archives was that a photo or audio snippet might show up in a hate group's promotional material. [17]

McAlister also faults the fact that the news APIs were one-way: a partner could request information from the news organization but not contribute information back to it. In 2014 he, Tackley, and Tom Armitage got funding from the Knight Foundation to explore this possibility, with examples such as a "realtime clicker" by which an audience could react to debates, and a simple survey tool that a newspaper could easily mount.[18] Thinking of news as a one-way flow may be a residue of pre-Net culture.

One of the people interviewed asked not to be cited as the source for two other suggestions about why public APIs have failed to thrive. First, he thinks the cultural divide has driven a business divide within his organization: “The tech people have embraced it. The commercial people are worried about losing control of brand and unauthorized content.”

Second, perhaps the fact that many of the initial technology advocates were promoted out of their jobs took some of the wind out of the public API sails.

Or perhaps the utopians simply over-estimated the world’s eagerness for metadata about the news media’s content.

The Indoor API

If the external story is one of disappointment, the internal story is about exceeding expectations.

NPR and *The Guardian* tell matching accounts about the moment they understood the value that APIs could bring to their organizations.

On January 27, 2010, Steve Jobs introduced the first iPad at one of his trademark on-stage demos. According to Jacobson, three weeks earlier NPR had received a call from Apple inviting them to participate in the launch by showing off an NPR app for this first-of-its-kind piece of hardware.[19] McAlister says *The Guardian* got a similar call and was just as eager to share in the excitement.

Three weeks later, NPR had a native iPad app ready for the launch. In fact, in those three weeks it also produced an app for the Safari Web browser — "Different technologies leveraging the same API infrastructure," Jacobson says.[20] Five or six weeks later *The Guardian* had an app for the iPad.[21] Without an API, development time would have taken months. Moradi explains, “Having the API meant we already had all the technical problems about getting content solved. We spent the few weeks we had designing the user experience and coding the app.”

Typically when developing software of this sort, there will be a team working on the data the app will use (the back end) and another team working on how the

end user will interact with that data (the front end). Front end work is often much more demanding when developing for a new category of hardware, but that's also where the opportunity to excel is.

An API enables the front end developers to work independently of the back end team. They can try out an interface, tweak it, implement a new idea for a feature, drop it or change it, all without ever asking for anything from the back end folks.

"No way we could have gotten that done without an API," says Jacobson. McAlister agrees. "We worked hard to be in their inner circle as a launch partner," he says. "Because we had the API we knew we'd be able to build something amazing quickly." The app they built, EyeWitness,[22] provided access to the hundred most recent photos in their EyeWitness series, along with "pro tips" from *The Guardian*'s team. In the end, "Jobs hauled it on stage" at the launch.

In September 2014 *The Guardian* announced it was no longer supporting the EyeWitness app because that functionality had been rolled into a new suite of apps.[23] That's a good run for an app, especially one created so quickly. And it's an unqualified success story for the *Guardian* API, since the new suite that obviated the EyeWitness app uses that API as well..

The claims of the early adopters that their APIs were "transformative" and "revolutionary" are supported by the range of overlapping benefits they have brought to their organizations:

Driving Traffic

In September 2011, *The Guardian* decided to bring the news to social media, announcing on its blog that the *Guardian* Facebook app would enable "Facebook users to read and experience *Guardian* content without leaving Facebook." If you clicked on a link within the app, the Guardian's content would be shown on a Facebook page so you can see "what your friends are also reading from the Guardian…"[24]

In December of the following year, the success of the new direction motivated *The Guardian* to tack. As the announcement on the company's blog said:

> In the months following the launch, we saw tremendous volumes of traffic being generated by the app. Over 12 million Facebook users have authenticated the Guardian Facebook app since launch, and at its peak (April 2012) we were seeing 6 million active monthly users.[25]

With that many users, *The Guardian* decided they'd rather have clicks take users to the articles on *The Guardian* site, rather than defaulting to embedding the article into Facebook pages. "The Facebook app has given us access to a hard to reach audience and has helped us learn much more about our new and existing readership which, as a digital first organisation, is crucial," says the blog post. No mention is made of the increase in ad revenue the switch would bring.

Jacobson cities NPR's own evidence of the effect an API can have on traffic:

> Around 2009–2010 there was a one year stretch in which traffic to all of NPR's traffic increased 100%. Eighty-five percent was due to the platform development by the API. That tells a powerful story about the power of APIs.

Syndication

Syndicating content brings it to users rather than requiring them to go to the originator's site, lowering the hurdle — which is, granted, only a single click high — to getting that content in front of a user. This can bring revenue from embedded ads, increase the brand recognition of the source, and help fulfill the news media's mission of informing the public.

"Most of our syndication partners use our content API," reports *The Guardian*'s Tackley. That's for a good reason: with an API, the recipient of the content can decide what content is desired, and can use search engine techniques to select content. For example, a service could ask an API for headlines, links, and snippets of articles that contain the words "Paris" and "Texas" but not "France" in a headline and that have a related audio file.

The flexibility of APIs makes syndication more flexible and more appealing.

Opportunism

NPR's APIs gave the network "the ability to move quickly, innovate, and be ready for what comes next," says Moradi.

Building the iPad app under an absurdly short deadline has become the canonical example, but there are so many others that the organization's expectations have been altered. Projects that used to require a major commitment of time and resources have become relatively trivial. It's now easier for the organization to say yes to a new product or new integration with a strategic partner.

Lower Risk

When a change in the information ecosystem requires updating the back end, it can be accomplished without affecting the operation of the sites that depend on it.

Says NPR's Moradi: "It's as close as life ever gets to future-proof."

Spreading the Brand

"We don't have to be the ones to get our brand everywhere," says *The Guardian*'s McAlister. "At some point we have to let go of it."

Moradi says back in the heady days when the NPR API launched he used to tell "wild stories" to NPR folks to get them excited about the possibilities. Referring to the popular driving-based video game Grand Theft Auto, he would say that "If you want to turn on a car radio in GTA and have it play 'Morning Edition'," the API is the way to have that happen.

It didn't happen with GTA, but it did with Ford. According to Sarah Lumbard, who was responsible for business development for NPR Digital during her time there from 2010 to 2014, the API made it easy and straightforward to integrate NPR content into the entertainment and communications system that the automaker calls Ford Sync.

Likewise, when Spotify rolled out an open platform for people to be able to create their own channels, *The Guardian*'s top ten list became one of the apps Spotify featured in its launch, thanks to its API. *The Guardian* continues to supply Spotify users with playlists drawn from its music section, as well as from special features such as music local to areas covered by *The Guardian*'s travel section.[26]

The API not only makes it easy to do this sort of integration, it removes the need to ask permission from the news media itself. For example, in 2013, the Harvard Library Innovation Lab wanted StackLife, an experimental browser for the Harvard Library's nearly thirteen million items, to embed an audio player on any page about a book about which there was an NPR story; the player would let the user listen to the NPR story while remaining on the StackLife page.[27]

Paul Deschner, the Harvard developer who implemented the feature, reports that it took a few days. This not only puts NPR content just one click away from users of StackLife, it also positions NPR as a valued source for Harvard students and faculty.

More Platform Reach

In early 2014, the crossover happened: In the U.S., people spent more time on the Internet on their mobile phones than on their desktop and laptop computers.[28] APIs vastly eased the development of apps for this new environment. But mobile phones and tablets are certainly not the last new platform that will be introduced. For example, Tackley says *The Guardian* created an app for Google Glass, an environment with its own unique constraints and user interface. The API will enable *The Guardian* to get up and running on whatever is coming next with a minimum of effort.

NPR is relying upon its API to fulfill its strategy of becoming the "Pandora of news," as NPR's former CEO Gary Knell termed it.[29] The first step was to create the NPR One player that streams NPR content, personalizing the stream based upon each user's interaction with it.

But how to get this player to work on the ever-increasing number of devices? On May 6, 2015 the organization announced that authorized external developers

will be given access to an API so they can adapt the NPR One player to new devices.[30] Demian Perry, NPR's Director of Mobile, contrasts this with the early aims of providing an open API. "The first API, the Story API, was intended to inspire ideas and creativity with our content," he says. For the new program, "We're proposing a fairly refined use case": adapting the NPR streaming player to new platforms, with a degree of customization by the developer.

Perry says that they are requiring developers to apply for API access to make sure that they're "trustworthy" because data from users will be flowing back through what they develop. Also, NPR wants to avoid having too many redundant players on any one platform.

By allowing controlled access to their API, NPR is enabling itself to scale up across existing devices, and devices not yet invented.

Internal Customers

NPR is a member organization. Jacobson credits the API with substantially improving the relationship between the mother ship and the local stations.

Without an API, if a member station wanted to feature content from NPR on its Web site, it had to take the content that NPR offered. With the API, it became much easier for a member station to specify exactly what content it wants — perhaps movie reviews' headlines and ratings but not the summaries or list of actors.

This flexibility encouraged the use of more NPR content on by the member stations websites, and made them feel more in control of those sites. “It really improved our relationship with the member stations,” says Jacobson.

Better Content Management

All three of the news organizations credit their APIs with making their CMS more flexible and usable. “It’s completely revolutionized how we publish content” says Tackley.

“It used to take lots of tools and direct access to the database to get all the sites right,” McAlister of *The Guardian* says. Rajiv Pant of *The New York Times* agrees:

“If you have to change, say, the content management database to be faster or to handle more data, without an API all the apps that talk to it would have to change, which would be a huge effort.”

Because APIs typically talk to a full text index of the CMS, they are able to provide services that the CMS by itself would labor to produce. For one thing, the full text index software typically can be easily configured to return results that are better tuned to users’ needs. For example, results could be weighted so that the staff’s reviews count for more than user reviewers. In addition, index software makes it easy to enable users to “facet” on searches, i.e., narrow a set of results by date, media type, author, or section, as at *The New York Times*' site.[31]

Also, an index stays fast even when there are massive numbers of users using it.

Internal Web Awareness

The Guardian’s Tackley believes that lowering the hurdles to interacting with the latest innovations on the Web has helped the organization to stay up to date and on the leading edge.

Experimentation

APIs lower the cost and risk of experimentation: developers can change a site without having to ask for any changes from the back end. If the experiment goes wrong, the error is confined to that one site or service.

For example, *The Guardian* described its 2011 Facebook app as an “experimental social news experience” and as an experiment in providing a “reading experience” away from *The Guardian*’s main website. "No additional editorial effort was invested in any aspect of the app," the blog post said, demonstrating the power of Daniel Jacobson’s COPE principle.[32]

Reputation and Recruiting

All three organizations stress the way in which their early and very public adoption of APIs has enhanced their reputations.

McAlister: “The reason *The Guardian* has become known for innovation in the world is that it’s so easy for us to become a go-to partner.” He adds, “If a big brand comes along we’re able to just say yes. There’s no debate.”

Jacobson: “COPE and the API and all of the evangelism publicly that we did helped put NPR on the map as a technical organization. I think that had an impact on business development, but also on recruiting and raising the visibility of NPR as a technical place to work.”

Pant: “People see that *The New York Times* is also a software engineering shop in addition to being one of the world’s best news organizations.” In fact, since developers can use the public APIs, it gives them "a taste of what working at the *Times* is like.”

Paths Forward: Opening up APIs

It’s of course important to learn lessons from failures, but it's even more important not to learn the wrong lessons.

The obvious lesson of the failure of news APIs to spur the external development of apps and services is that there is insufficient demand for those apps and services. But the actual conclusion should be that there was insufficient demand at that time and under those circumstances.

Times and circumstances change faster than ever these days. There are perhaps now ways in which an open news environment might come to flourish.

Increasing Internal Rewards

NPR, *The New York Times*, and *The Guardian* agree that public APIs did not catch on at least in part because there weren't enough business reasons for them to do so. Matt McAlister puts it concisely: "Money talks, even in places where journalism calls the shots."[33]

As news media become increasingly economically straitened, the business benefits of creating public APIs may change. These include:

- Increased reach
- Brand awareness
- Building a reputation as a technologically advanced organization, aiding in the recruitment of technical staff
- The creation of apps and services that meet particular needs but that would be too resource-intensive for the news organization to provide
- Mission fulfillment

The organization could also create more business opportunities for itself. For example, McAlister argued at *The Guardian* "that we'd build an ad network through our partners, but for that you need a content network first." The public API would make it easy to get partners. Those partners then could accept ads provided by *The Guardian* and would share some the additional revenue from those ads.

Increase the External Incentive

Organizations can make it more attractive for developers to use the APIs that they have made openly available. For example, they could:

- Open the possibility of sharing in some ad revenue.
- Treat the developers as valued partners in real and symbolic ways: host events, provide special access to some news or features, tout them and their work.
- Feature and promote the apps and services that are built using the public APIs.
- Develop a community of developers that can support its members technically and socially.

Much of this could be achieved with a single resource who takes on community development as a part-time responsibility, as well as some time from

internal developers, funding for some events, etc. This is not a major expense, but a news organization would have to be convinced that it's worth the time and money.

Cultural Change

Many Web software developers default to publicness, including making their code open and reusable. In such an environment, a closed API looks like a lost opportunity to generate additional value from existing data. As the Web engineering culture continues to proliferate, public APIs may be perceived as generally more valuable than closed ones.

Open Access to Data as a Wedge

Providing access to the raw data behind stories is becoming more commonplace, as at ProPublica,[34] *The Guardian*,[35] and *The New York Times*.[36] It is a small step from providing open access to quantifiable data to providing open access to metadata about content.

Aggregator APIs

News organizations are not the only ones who have access to the metadata about their content, including headlines, snippets, summaries, and links to the original article. News aggregators do as well. Any one of them could create an open, public API.

This is indeed something that Jason Calacanis, the creator of Inside.com has given some thought to. Inside.com has a network of people around the world who summarize stories according to the site's carefully-designed structure: a summary sentence instead of a headline, another sentence or two of explanation, a visual element, a link, etc. In an interview, Calacanis said that the site plans to "have an API available to the public." He speculated about Inside.com's business model for this: "We can let you use it [the API] for free up to a point and then you pay for it. If you don't pay for our API, you have to take our ads."

News aggregators know less about the stories that they aggregate than do the source media. The source has access to the full CMS, whereas the aggregator generally only knows what it can glean from the metadata the source organization has decided to publish. Even so, developers might find it very attractive to have a public API put up by an aggregator who knows a little about a very wide range of news media stories.

Open Source API

All three news organizations have demonstrated solid business reasons for adopting APIs for internal use: lower cost, greater flexibility, future-proofing, greater reach, more page views, and higher impact.

So, assume that in the future more news media will add internal APIs to their CMS for purely self-interested reasons.

It does not take a huge leap to imagine that a developer somewhere might create open source API software designed for news organizations. It would run on top of one of the standard open source full text indexes such as Elastic Search or Solr (both of which are built on top of the indexing software Lucene). The index would itself run on top of the organization's CMS. Such an API would appeal to news organizations because the software is free, and because open source software can be extended or modified by anyone without asking permission, which lessens an organization's dependence on a commercial supplier's decisions about features.

As more and more news organizations use this open source API, a powerful additional motivation arises: as people within and outside of news organizations create new services that use that API, those services can be made available to all users of the API, no matter what CMS is underneath it. These services might be anything from a simple dashboard for tracking an article's progress, to complex analytics of its impact, to an entire ad-sharing network management system. Because an API provides an abstraction layer, a service that works for one organization well might work for another organization using the same API.

Now take one more speculative step. Suppose the developer of this open source API were to build in the ability to configure it so that it makes some set of the organization's content and metadata publicly accessible. Suppose it came with the mechanisms required to regulate usage of the API by external developers, such as providing authorized keys and limiting usage. It would, in short, be a turn-key system for opening public portals in an internal API.

This could have two effects.

First, it would lower the technical hurdles for a news organization considering opening up a public API. Once the organization made a business decision to do so and agreed on what content and metadata to make available, the public API could be created simply by rewriting some lines in the internal API's configuration files.

Second, it would then be straightforward to write applications that fetch data from multiple news organizations. Because the APIs are the same, they would constitute a networked, distributed news infrastructure that would enable applications and services that no single news organization could — or even would — provide.

- Critical mass might thereby be achieved, opening up some of the utopian possibilities that drove the initial excitement around open news APIs:An app or a site could show issues reflected through media diverse in their geography or political orientation.
- We could see which articles were the most read or most shared, clustered by their focus or demographics, and across media.
- Readers could see how an issue connects to others via a web of shared topics and cross-references.
- Researchers could more readily watch the movement of ideas through the Net.[37]

Then there are all the uses that cannot be anticipated, which is a more potent reason for adopting open platforms than any set of expected uses.

Thus in this speculative model we get to the utopian environment through the self-interest of news organizations…with a just a soupçon of mission-based, low-cost openness.

A Note on Costs

Matt McAlister in an email provided guidance on the cost of creating an API that runs on top of a news organization's existing CMS.[38]

Assuming that the API runs on top of a full text index of the organization's content, powerful open source indexing software is available for free. If the organization decides to host it in the cloud rather than maintaining it on its own servers, McAlister estimates the charges will be between $100 and $1,000 per month.

Writing API software that will output data in the desired formats will take three to six weeks of a developer's time, McAlister estimates. He points out that some search engines come with API functionality built in, which can lower these costs dramatically.

Given that the API also needs a web site, documentation, testing, etc., McAlister concludes: "[A] small project could happen in a month and cost about $1,000, and a more robust project could happen in 3 to 4 months and cost $50,000 or more."

This estimate puts an API on top of an existing CMS. Replacing the CMS, optimizing it for API access, and rewriting existing apps and services so that they use the new infrastructure can be a much larger project. "But," adds McAlister, "if you ever plan on redesigning your CMS, web site or apps, then you have to think about doing this. It will save you a ton of money and time, and it will future-proof you from many costs of doing business in a digital world."

Paths Forward: Interoperability and News You Can Reuse

"…we are falling behind in a … critical area: the art and science of getting our journalism to readers."[39]

Thus states the 2014 *New York Times* Innovation Report in its second paragraph.

The initial strategy news media adopted — trying to draw people to their websites — clearly has not sufficed. The *Times* report found that visits to its homepage dropped by about half between 2011 and 2013. Overall visits to the site have remained relatively steady, meaning that more and more people are getting to the content through indirect means, such as search results and links from social media.[40] Although a recent Pew study found that users who arrive at news sites directly, not by links, visit more pages on those sites and spend more time on each page, online news sites can only grow their audience by going where people are.[41]

One tactic for increasing the visibility of online journalism is to pander to social tastes with screaming clickbait headlines. But another approach hews closer to the social mission of news media: design news content for the network engine.

The Internet really isn't a series of tubes. Content moves through it because someone was interested enough to link to it, and someone else was enticed by that link to pull the content toward her or him. Interest is the motive force of the Net. When marketers forget that and try to push their content through the Net, we call it spam. News media who forget this think their content is so important that people will make a special trip to their site to find it.

The Guardian understood the challenge early on: "Our editor-in-chief Alan Rusbridger absolutely believes that being a news organization on the Web means being part of the Web," says Graham Tackley. Being *part of* the Web requires not just being *on* the Web. News media will have continued relevance only by lowering the barriers to finding, accessing, and re-transmitting their content so

that it can show up wherever users are and on whatever apps or devices they're using.

APIs are an important part of this strategy, but are not the entirety of it. News also needs to become more *interoperable*: capable of being usefully reused by multiple sites and services so that stories and their elements can be discovered, integrated into other sites and content, and redistributed by its users.

For example, appointments on your electronic calendar are interoperable if you can easily move them to a different calendaring system (say from Google Calendar to Outlook) and the new system can, for example, automatically tell the starting time from the ending time. A news story is only minimally interoperable if another site can post it. It becomes truly interoperable when a site can recognize which text is the headline, which is the byline, and even who is shown in the embedded photo. That type of discrimination enables other sites and services to reuse the elements of the story intelligently.

Interoperable news is easier for people to find, care about, re-express, and move through their network. People are more likely to move a chunk than an entire article, chunks are more likely to drive interest in the entire article, and chunks are easier to "mash up" with other chunks.

Here are steps that would make news stories more interoperable, and that treat the Net as an engine and not just a series of tubes.

Design for Dispersal

If the Net-based strategy is to get news media's content out to where people are, then articles will necessarily be seen out of the context of the source's site. They therefore need to carry more of their context with them.

It's probably a good practice in any case to assume that networked readers need the context that clarifies the news' significance; the Internet fragments much of what it touches. For example, the editor-in-chief of CNN Digital, Meredith Artley, reported at a 2015 talk at the Shorenstein Center that CNN.com has adopted the practice of leading with the significance rather than the breaking news in order to make stories more accessible and interesting to readers.[42] This

is even more helpful as the article (or its linked summary) starts showing up outside of CNN.com.

The ability to stash hidden metadata inside an article can enable a news site to have it both ways: the lead can remain terse and relatively context-free while the contextualized version is embedded in the metadata description field for any app that wants to expose it.

License for Reuse

Content won't be accessed if the licenses for using the metadata about that content are not clear and designed to encourage reuse.

The fewer and clearer the restrictions, the lower the barrier to reuse.[43] For example, the *Times* requires external apps to attribute the content to the *Times*, and forbids apps from modifying the content, distorting its meaning, or suggesting that the *Times* endorses the app's use of it.[44] This is expressed in clear language, although the application of prohibitions on modifying content or distorting meaning is of course subject to interpretation.

Be Generous with Metadata

The more metadata for each piece of content, the more apps can do with it, and the more reasons can be presented to a user for clicking on the link to encounter the content. Including a generous summary or description as open metadata can greatly increase the allure.

The *Times* is typical in that its Articles API provides much metadata but no actual content other than the headline, lead paragraph, a snippet, and an abstract.

Work with the Search Engines

To make it easier for its content to be discovered, used, integrated with other information, and redistributed, news media can help search engines better assess what its stories are about. The key is to supply explicit metadata about the article.

There are three basic ways to do this. The first two are quite common.

First, SEO (Search Engine Optimization) at its best helps search engines discern what the topics of an article are. For example, Susan Steade recommends a "mullet approach":

> "Business up front, party in the back," which simply means get the words people search for at the front of the headline…[45]

At its worst, SEO does what it can to trick the search engine into returning an irrelevant story. The euphemism for this form of dishonest practice is "gaming the system."

Second, the news source can use explicit metadata to flag what the topics of an article are. This metadata is hidden from the user but is visible to search engines and other applications that encounter the page. For example, there are standard metadata tags, very commonly used by sites, for labeling a brief description and keywords.

Third, the news source can insert metadata that identifies the structural elements of a story and clearly identifies the people, places, things, and events that the story refers to. There have been various attempts at standardizing how that should be done, but one is now gaining traction because the major search engines recognize it and use it in their decisions about how to rank results.

The effort is called Schema.org, and it is being supported by Google, Bing, Yahoo, and Yandex. It deserves its own subsection…

Schema.org

Schema.org consists of a set of terms that can be invisibly embedded in Web pages so that the meaning and role of constituent elements can be understood by search engines and other applications.

For example, imagine the number "1" is somewhere on a Web page. With Schema.org, the page creator could indicate through hidden metadata that "1" refers to how many hours it takes to prepare a recipe, the number of teaspoons of baking powder to be added, or the rating the recipe got from its users. Likewise,

with Schema.org a page creator could indicate that “Jane Austen” is the name of the author of a book while “Elinor Dashwood” is the name of a character in a book she wrote. This lets the search engines return more accurate responses to queries.

To use Schema.org, the publisher inserts hidden metadata that states what type of document the page is, choosing from the available categories. There are dozens of these categories so far, including recipes, movie reviews, and news articles. You then use the specified vocabulary to label the elements of the page. For example, the writer should be identified as “author,” and the date as “datePublished.” These are inserted into the HTML that the browser turns into a viewable page; browsers know not to display Schema.org metadata. There are a variety of tools that enable people with no expertise in HTML to add the Schema.org markup as part of the document workflow.

Times column: Reader's view

```
<meta property="article:author"
content="http://topics.nytimes.com/top/opinion/editorialsandoped/oped/colum
nists/maureendowd/index.html" />
<meta property="article:tag" content="Stroke" />
<meta name="des" content="Stroke" />
<meta property="article:tag" content="Brain" />
<meta name="des" content="Brain" />
<meta property="article:tag" content="Emergency Medical Treatment" />
<meta name="des" content="Emergency Medical Treatment" />
<meta name="keywords" content="Stroke,Brain,Emergency Medical Treatment" />
<meta name="news_keywords" content="Stroke;Brain;Emergency medicine" />
<meta property="article:modified" itemprop="dateModified" content="2015-05-
04" />
```

Times column: Schema.org metadata (snippet)

Schema.org is not only one of the few SEO tools sanctioned by the major search engines; it can also help news media anxious about the snippets of news stories published by search engines and other aggregators. Rather than relying on the search engines to take an algorithmic stab at what an article is about, a content producer can embed its own snippet as Schema.org metadata. It is thus not surprising that a spot check suggests that many major news media are already supporting Schema.org.

But there is another reason to use Schema.org. Evan Sandhaus at *The New York Times* has been following the development of Schema.org's vocabulary for news articles from before its inception, including the influence that the International Press Telecommunications Council has had on it. The Schema.org approach is, he says, "probably even a bigger deal in terms of how news organizations get their data out there than APIs." He explains that when a user posts a link at sites such as Facebook and Twitter, the sites automatically scan the linked pages for Schema.org markup. This lets them do things like format the headline appropriately for their site, display the date in a way consistent with their users' preferences, and embed topic tags to suggest related links. "The page itself becomes kind of like an API," Sandhaus says, in the sense that it can be queried for information about itself. "That's huge!"

Persistent Identities

An obvious strength of news reports is that they can talk about anything that happens. That is a huge problem for interoperability, however. There is always more than one way to refer to anything, which confuses computers, our great Engines of Literalism.

That's why there has been an enormous effort over the past couple of decades to come up with ways of enabling computers to identify the things in the world that pages refer to. This is a two-part problem.

First is the process of *entity extraction*: recognizing some strings of characters as referring to people, places, things, and events in the world. If a computer sees the phrase "London Fog" without further help it won't know if it's referring to fog

in London or to a brand of raincoat. If it's referring to a city, is it the London in England, Ontario, or Tierra del Fuego?

Second, once an entity has been unambiguously identified, how can it be referred to without reintroducing ambiguity? A best practice is emerging. Rather than identifying things by using words in some preferred language, instead use a link that points to some well-known, stable resource on the Web. For example, the metadata could identify London not with a phrase but with a URL such as http://en.wikipedia.org/wiki/London, which points to the Wikipedia page about London.[46] A computer that sees metadata on two separate Web pages that both link to the same Wikipedia page can be quite confident that they're talking about the same thing. Additional sources are emerging for linking to disambiguated people, places, and things.

The use of persistent, unique identifiers for the things that news articles talk about enables information from all around the Web to cluster. We can discover more and more about the things the news cares about. The contribution of news media to our knowledge of those things then becomes part of our web of knowledge, discussion, and understanding. And, not so incidentally, the news articles about those things are more discoverable, including by people who didn't know they were interested in them.

Conclusion: Utopia Redux?

With interoperability, we may get to something like the failed utopia but for pragmatic, non-altruistic reasons.

News media's internal APIs have succeeded beyond expectation because they make the content development and distribution processes far more resilient, lower the cost of building and maintaining sites, provide more granular access to content, enable experimentation, and are the backbone of powerful strategic partnerships.

With the rise of Schema.org, news organizations have a self-interested reason for making their news stories interoperable across all boundaries.

Interoperable content from multiple news media can be treated computationally as a single distributed database representing many views on the events of our times. It can be traversed by algorithms looking for latent connections. It can be made more traversable by incorporating the latent meanings discovered by human beings.

This brings us significantly closer to what in clickbait fashion we might call “The News Singularity” in which news is undammed and forms an ocean of information, ideas, and human expression ever richer with meaning. Such an ocean opens far more opportunities for synthesizing value than any one news source ever could, even three as august as NPR, *The New York Times*, and *The Guardian*.

The rise of internal APIs brings us one step closer. The experiences of the three major news organizations we've looked at proves the business benefits of APIs. Once in place, reconfiguring them for external access is not a significant problem. Making the business decision to do so is a far greater obstacle, since it involves a leap of faith (Will anyone develop anything worthwhile?), a risk (Will our content be ripped off or abused?), and a commitment to some level of support for developers (How much technical help and community development are we signing up for?).

It is probable, indeed likely, that news media will continue to see public APIs as an investment that brings them too little direct benefit. The industry's experience may convince the news media that there is no real demand for public APIs: they built them in 2008 and few came.

But the technical barriers are lower than ever and conceivably go as low as altering a few lines in a configuration file. The benefits might increase if enough news media open up public APIs creating a critical mass of content and metadata.

Perhaps most important, as the news media become ever more concerned that their "product" is under-valued on the Net — the kids don't read newspapers, Facebook is filtering out news stories — they could view the development of an open ecosystem of news as an environmental concern. By opening up portions of their internal APIs for public access and making their stories interoperable, news

media could create a deep, rich, and lively resource that would enable the engine of the Internet to engage with that content, making the news an irreplaceable component of the infrastructure of the Net.

Appendix 1: How an API Works

Let's say you're a software developer writing a music listening app and it occurs to you that your users might like to be able to see what *The Guardian* knows about the musicians they care about.

One of your users clicks on the Beatles, a popular band from the 1960s. To retrieve what *The Guardian* has published about the Beatles, your app will check with the *Guardian* API. To do this, your app has to know how to phrase the question — or *query* — in a way that the *Guardian* API understands.

The Guardian site for developers explains its query language in detail. It is quite typical. Here's the Beatles query:

q=Beatles&api-key=test

The ampersands mark the beginning of a new element of the query, so the above is equivalent to:

1. q=Beatles
2. api-key=test

For the Guardian API, "q" means that you want the site to search for mentions of "Beatles." The label "api-key" is the personal ID that the Guardian gives to developers for free. (*The Guardian* allows anyone to try out the API by using the key "test." A real key is a couple of dozen random letters and numbers.)

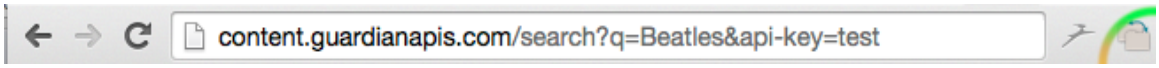
But how is your program going to send that query to the Guardian API? Here's where the system gets brilliant. APIs have Web addresses. *The Guardian*'s API's Web address is http://content.guardianapis.com/.[47]

The World Wide Web was designed to let a user click on a link that sends a message to a website anywhere in the world and get a Web page back. Modern APIs — technically called RESTful APIs[48] — use the Web to communicate data, not pages. RESTful APIs have become the norm.

The fully formed query that your app constructs when a user clicks on the Beatles is:

http://content.guardianapis.com/search?q= Beatles&api-key=test

This is a legitimate Web address. In fact, you can copy it into your favorite Web browser's address bar and click the "Go" button. In response, your browser will show you the raw data that is the answer to your query.



The Guardian's API now interprets the query you've sent it, translating it into the language that its data management system understands.

This might be its CMS, but more often these days it's a full text index of that CMS. Think of the content of a CMS as a giant book. It's got all the articles and other media published by *The Guardian* organized into well-thought-out chapters and sections such as News Articles, Music Reviews, and Sports Scores. If you wanted to find every reference to the Beatles, you'd have to skim through the entire book. A CMS can do that quite quickly, but it's still a lot of skimming, and when you have a popular website with lots of people looking for information, it can become laborious, especially for searches more complex than looking for a single word.

The solution for websites is much like the solution for printed books: create an index. The mechanics of a computer index are quite different from the alphabetical listing in the back of a book, but the principle is the same: Process all the words ahead of time so that they can be found without having to re-read the entire content for each query.

Full text indexes bring some other benefits as well. For example, an index can be easily configured to weigh some factors more heavily than others when deciding the order of results. For example, it could be adjusted to give more

weight to articles that have “Beatles” in the headline or to articles that are more recent.

Depending on whether the API is talking directly to the CMS or to the index, it will speak different languages. And each CMS and index has its own language or variant. So, the API developers will have to do some work to translate the queries it receives.

In our example, the query in the Web address was quite simple:

```
q=Beatles
```

The “q” in *The Guardian*'s API language means “Find everything that contains the word after the equal signs.” Many of the most popular index engines expect exactly that syntax for their queries, so the API doesn’t have to do any translation at all.

If a user wanted to find only videos pertaining to the “Beatles,” the query coming in to *The Guardian* API might look like this (minus the API key):

```
q=Beatles&type=video
```

This too would require little translation, perhaps into:

```
q=Beatles&type=true&type.field=video
```

It is not a coincidence that some of the most popular indexes and APIs speak such similar languages: people who write APIs make the translation task easier if the API syntax they specify closely matches the index’s.

The translation work done, the API has the index deliver up its results. It then provides this data in a form that will be convenient for the app developer. One very popular form is JSON (JavaScript Object Notation) because the Web page programming language JavaScript has built-in ways of working with it.

In our example, *The Guardian* will return a list of the ten most relevant items that contain the word “Beatles.” (It will also report that it found a total of 8,980 matching items.) Each item on this list contains seven pieces of information, including its title, date, Web address, and the section it appeared in. Each piece of information is expressed as a *key/value* pair. For example, the title of the first article looks like this in JSON:

“webTitle":"Music institute opens in Beatles' Abbey Road Studios"

“webTitle” is the key, and everything after the colon is the value. Here is the complete set of information about that first item:

{
"webTitle":"Music institute opens in Beatles' Abbey Road Studios",
"webPublicationDate": "2015-03-19T17:53:32Z",
"sectionId": "music","id": "music/2015/mar/19/music-institute-opens-in-beatles-abbey-road-studios",
"webUrl": "http://www.theguardian.com/music/2015/mar/19/music-institute-opens-in-beatles-abbey-road-studios",
"apiUrl": "http://content.guardianapis.com/music/2015/mar/19/music-institute-opens-in-beatles-abbey-road-studios",
"sectionName": "Music"
}

The app now receives this information. The app has been programmed to turn it into Web content that the user can see and interact with.

The Guardian API waits for the next request from some app somewhere in the world.

Appendix 2: Creating a Public API

If a news organization already has an API for its internal use, opening it up to external developers requires little additional technical work. For many of these steps there are robust open source solutions.

- Make sure that your API supports current Web standards, including being a RESTful API and producing output in JSON, XML or other expected formats. (See Appendix 1: How an API Works.)
- Create a system that issues "keys" (passwords) to registered developers, and configure your API so that it only responds to requests for information that include a registered key. That lets you know which apps are using the API, which is helpful for analytics. It also lets you shut off any apps that are abusing the API. Also, the process of requesting a key lets the news organization know what developers are using it, which facilitates community building.
- Establish and implement usage guidelines. For example, it is common to limit the number of requests per second and per day.
- Consider creating tiers so that developers who want to make more use or commercial use of the APIs are able to do so for a fee without requiring your organization to absorb the associated costs (bandwidth, hosting, etc.).
- Provide documentation and tools to lower the hurdle for developers. If an organization is using internal APIs, it very likely can repurpose what it already has.
- Lower the hurdle further by providing easy-to-use tools, tutorials, and pre-made widgets to make it easier for sites to incorporate your content and metadata without having to do hard-core coding.
- Devote some organizational time to building and maintaining a community of developers. They will make one another more effective, efficient, and enthusiastic.

NPR's Daniel Jacobson is emphatic, however, that while the technical costs may be relatively small, "there is a range of non-technical issues that become much more challenging with the public API," including issues around rights, nurturing the community and policing and enforcing the use of trademarks, attribution, content, etc.[49] "Those are the big costs (not the technical ones) that, when compared to the small business value, make it not worth doing, he concludes."

Appendix 3: Supplementary Services for Readers

1. I have created a small online service for checking whether a particular news site supports Schema.org: http://hyperorg.com/programs/schemaorgSearcher/schemaOrgSearcher.html
2. With the help of my researcher Ria Mukherji, and at the suggestion of John Wihbey, we have started a page at Wikipedia listing news media with open APIs. You are invited to extend and correct that list: https://en.wikipedia.org/wiki/List_of_news_media_APIs
3. For an extended hands-on API tutorial designed for beginners, with library metadata as its example, please see: http://hyperorg.com/misc/myFirstApiScript/myFirstAPIScript-HarvardLibCloud/myFirstApiScript_article.html

Acknowledgements

Thank you to the Harvard Shorenstein Center on Media, Politics and Public Policy for their support throughout this project. The chance to work with and be edited by this group of superb journalists, teachers, mentors, colleagues and friends has been a once-in-a-lifetime opportunity. This is true at every level: fellows, staff, faculty, and students.

Special thanks to my cohort of Fellows: Bill Buzenberg, Jackie Calmes, Michele Norris, and Nick Sinai.

John Wihbey at the Shorenstein Center was helpful beyond any requirement or expectation and consistently insightful.

My research assistant, Ria Mukherji, was not only helpful and resourceful, but a delight to work with.

A special thanks is due to the people who consented to be interviewed. They were to a person exceptionally generous with their time and patient in their explanations. Thank you!

Endnotes

[1] Daniel Jacobson, in an email, April 9, 2015.

[2] Dave Fetterman, “Facebook Development Platform Launches…,” August 16, 2006. https://www.facebook.com/notes/facebook/facebook-development-platform-launches/2207512130

[3] Debrah Donston-Miller, “Use Facebook Apps To Woo Customers: 6 Examples,” *InformationWeek*, February, 8, 2012. http://www.informationweek.com/use-facebook-apps-to-woo-customers-6-examples/d/d-id/1102727

[4] “RSS” stands for “Really Simple Syndication” or “Rich Site Summary.” Both are attempts to preserve the initials of its source, “RDF Site Summary,” after the standard stopped using RDF. “RDF” stands for “Resource Description Framework.”

[5] Interestingly, the same person, Dave Winer, was a pioneer of both the RSS standard and podcasts. This was not entirely coincidental: podcasts were first enabled by a clever extension of RSS.

[6] “COPE: Create Once, Publish Everywhere,” Programmable Web. http://www.programmableweb.com/news/cope-create-once-publish-everywhere/2009/10/13

[7] http://www.npr.org/music

[8] Brad Stenger, “New York Times’ Derek Gottfrid and NPR’s Dan Jacobson Discuss APIs,” Wired.com, July 15, 2008. http://www.wired.com/2008/07/oscon_new_york_times_and_npr_discuss_apis/

[9] For an example of an innovative news API from 1992, see Gene Miller, Greg Baber, and Mark Gilliland, “News On-Demand for Multimedia Networks,” '93 Proceedings of the first ACM international conference on Multimedia, NY: ACM, pp. 383-392. doi: http://dl.acm.org/citation.cfm?id=168432

[10] Daniel Jacobson, email April 9, 2015. On April 29, 2015, NPR announced that it had made 800,000 audio files available for easy embedding in anyone's Web page. It exempted streams and some NPR Music exclusives. Patrick Cooper, " There Are Now 800,000 Reasons To Share NPR Audio On Your Site," *NPR About*, April 29, 2015. http://www.npr.org/blogs/thisisnpr/2015/04/29/401552958/there-are-now-800-000-reasons-to-share-npr-audio-on-your-site

[11] Brad Stenger, “New York Times’ Derek Gottfrid and NPR’s Dan Jacobson Discuss APIs,” Wired.com, July 15, 2008. http://www.wired.com/2008/07/oscon_new_york_times_and_npr_discuss_apis/

[12] Provided by Matt McAlister. Note that the gray lines are ornamental shadows.

[13] http://www.fantasygeopolitics.com/

[14] http://www.ifttt.com

[15] Moradi, email April 20, 2015.

[16] Matt McAlister, email April 13, 2015.

[17] Based on interviews I did at the time for a *Wired* article: "The BBC: A Case Study in Going Digital," *Wired*, September 2005. http://archive.wired.com/wired/archive/13.09/bbc.html

[18] The project is called Swarmize. http://www.swarmize.com/

[19] "NPR iPad App Now Available on the app Store," April 2, 2010. http://www.npr.org/about/press/2010/040210.iPad.html

[20] Daniel Jacobson, email, April 9, 2015.

[21] Jonathan Moore, "Introducing the Guardian Eyewitness app for iPad," April 6, 2010. http://www.theguardian.com/help/insideguardian/2010/apr/06/theguardian-eyewitness-app-ipad

[22] Tom Grinsted, "Eyewitness is moving from a stand-alone app to being integrated into our main offering," September 9, 2014. http://www.theguardian.com/info/developer-blog/2014/sep/09/eyewitness-is-moving-from-a-stand-alone-app-to-being-integrated-into-our-main-offering

[23] Meg Packard, "Introducing the Guardian's new Facebook app," Sept. 22, 2011. http://www.theguardian.com/help/insideguardian/2011/sep/22/the-guardian-on-facebook-app

[24] Ibid.

[25] Ibid.

[26] Anthony Sullivan, "The Guardian and Facebook - a more social experience," December 13, 2012.
https://partners.spotify.com/case-studies/case-study-guardian-newspaper/
http://www.theguardian.com/help/insideguardian/2012/dec/12/guardian-facebook-app

[27] I was co-director of the Lab at this time.

[28] Rebecca Murtagh, "Mobile Now Exceeds PC: The Biggest Shift Since the Internet Began," *Search Engine Watch*, July 8, 2014.http://searchenginewatch.com/sew/opinion/2353616/mobile-now-exceeds-pc-the-biggest-shift-since-the-internet-began

[29]Melissa Korn, "CEO Re-Imagines NPR as a Pandora of News," *Wall Street Journal*, July 9, 2013. http://www.wsj.com/articles/SB10001424127887324867904578595730483213490

[30]Demian Perry, "The NPR One Developer Center Is Now Live", *This is NPR*, May 5, 2015. http://www.npr.org/blogs/thisisnpr/2015/05/04/404212255/the-npr-one-developer-center-is-now-live

[31] For example, here is a search at NYTimes.com: http://query.nytimes.com/search/sitesearch/?action=click&contentCollection=Politics&region=TopBar&WT.nav=searchWidget&module=SearchSubmit&pgtype=article#/israel/since1851/allresults/1/allauthors/relevance/Opinion/

[32] Anthony Sullivan, "The Guardian and Facebook - a more social experience," December 13, 2012. http://www.theguardian.com/help/insideguardian/2012/dec/12/guardian-facebook-app

[33] Matt McAlister, email, April 13, 2015.

[34] ProPublica Data Store: https://projects.propublica.org/data-store/

[35]For example, Ami Sedghi and Simon Rogers, "Guardian/ICM Polls: every one since 1984", *The Guardian*. April 20, 2015.http://www.theguardian.com/news/datablog/2009/oct/21/icm-poll-data-labour-conservatives#data

[36] For example, the *Times* posted the data behind Matt Apuzzo's "War Gear Flows to Police Departments," June 8, 2014. http://www.nytimes.com/2014/06/09/us/war-gear-flows-to-police-departments.html The data are here: https://github.com/TheUpshot/Military-Surplus-Gear (Thanks to Alex Howard for this example.)

[37] Berkman/MIT's Media Cloud (http://mediacloud.org) enables this already, but it could do so with far more precision and completeness if a standard way of marking up the structural elements were adopted.

[38] Matt McAlister, email, June 10, 2015.

[39] The Innovation Report was published by Mashable.com: http://mashable.com/2014/05/16/full-new-york-times-innovation-report/. See also Sam Kirkland, "3 takeaways from the 'death of the homepage' and The New York Times innovation report," Poynter, May 19, 2014.http://www.poynter.org/news/media-innovation/252632/3-takeaways-from-the-death-of-the-homepage-and-the-new-york-times-innovation-report/

[40] Amy Mitchell, Mark Jurkowitz and Kenneth Olmsted, "Social, Search and Direct: Pathways to Digital News," Pew Research Center for Journalism & Media, March 13, 2014. http://www.journalism.org/2014/03/13/social-search-direct/

[41] Ibid.

[42] I liveblogged the talk here: http://www.hyperorg.com/blogger/2015/02/24/shorenstein-new-rules-for-modern-journalists/

[43] "Article Search API v2," http://developer.nytimes.com/docs/read/article_search_api_v2

[44] http://developer.nytimes.com/attribution

[45] Steve Buttry, "Susan Steade's SEO headline tips: Business up front, party in the back," *The Buttry Diary*, November 1, 2013. https://stevebuttry.wordpress.com/2013/11/01/susan-steades-seo-headline-tips-business-up-front-party-in-the-back/

[46] Technical people often prefer to talk about URIs (Uniform Resource Identifier) in this context. An URL links to a page. A URI links to information that may or may not be formatted as a page.

[47] Technically, *The Guardian* has multiple APIs. For example, one lets a program discover what tags have been assigned to Guardian topic.

[48] "REST" stands for Representational State Transfer, but the acronym is so common that the full name usually isn't even explained on first use.) See Wikipedia:

http://en.wikipedia.org/w/index.php?title=Representational_state_transfer&oldid=653506153

[49] Daniel Jacobson, email, April 9, 2015.