<meta name="viewport" content="width=device-width; initial-scale=1.0; "\>Event ID: 2028065

Event Started: 10/25/2012 3:00:00 PM

10/25/12 [Captioner present. Please stand by for real time.]

>> We'll be getting started in two minutes.

>> Thank you for joining us this morning. This is a webinar from GSA's digital service center. Covering APIs advanced technical overview. There are many technical aspects of any digital program including APIs. But five areas which we're going to focus today include formats, API keys, analytics, issues of scaleability and documentation. This is the 5th in an 8 part webinar series that seeks to provide an overview of everything from the basic fundamentals of providing APIs to your customers to more advanced technical aspects as well.

>>> We're going to be including experts including presidential innovation and census bureau that have a number of examples to show and context to provide. As we get started, I wanted to step back though because I know one of the more difficult areas can always be the question of what is an API. To that end, basically just how two computers, two web sites, two devices talk to each other without needing human being. By providing an API to your consumers, you are allowing third party developers to build an app off of your data and information without having to necessarily interact with you personally. The agenda today will go through these five topic areas providing overall context as well as examples. And then at the end have time for questions and answers. Please feel free to be submitting questions during the actual event and then at the end we'll take all the questions people have sent in and go through them. Thank you, though, for being here and we're excited to be talking to you about this information.

>>> To begin, we're actually going to start with an overview of formats that are commonly used in APIs. To that end, we'll start off with the census bureau.

>> Thanks for joining us on this API webinar. I'm Lisa from the census bureau's communications and I have with me some of the people who have worked hard to develop our API. I'll introduce some and they'll introduce themselves as they start to talk about the implementation details. We have bill RANKIN, Dave and riche from our research and methodology. Our API, just want to give a really brief overview. Our API we launched in July and offers access to two of our primary data sets. The American community survey and the 2010 census. You can see information about the API at census.GOV/developers.

>>> So we're going to talk first about formats and turn it over to Bill to discuss that.

>> Hi, this is Bill. Basically the -- we decided we wanted to go to a get rest full request instead of using soap or trying to do something with languages.

>>> Tried to simplify keep the API as simple as possible. We have basically four sets of parameters. One is the key which we'll talk about in another five minutes or so. The next is the data set variables that you want. Things like total population or the number of people of a certain age group. Limit for specific geographies and we also allow a contract of being able to do within parent geography. So for example, you could get some information for like all places in the country or you could get it for all counties within a specific state or group of states. The example URL there shows how to get the total pop for all 50 states and the district of Columbia I think is in there as well.

>>> When we talked about the output format, we decided that there's two major things that typically are utilized. One is XML the other is JSON. We decided to go with JSON primarily because it was a smaller footprint which was required basically for removal applications. If you are just doing -- if most of your clients were on web apps, this wouldn't be as important. One of the things we do is list the column headers once at the top. A lot of the JSON things they like to have the column headers repeat within within a list. But we decided to try to remove as much of that as possible. We make this so it fits like base storage on the client. And this tends to be faster and uses less memory than the object based storage that would be used for the XML.

>> Back to you.

>> For a bit more context on JSON you see the most because it's most popular.

>> This is Ben. Just for a little bit of background, there's a couple different formats for how the actual API, what does it look like and how is it consumed. There used to be a between between soap and rest. Rest is really the way we're going. At a high level all rest says, rest is an acronym. All that says is that every object -- so if you are a kennel and you have dogs. Each dog you have its own URL. Each page would have its own URL. This is called rest full rather than having question mark API equals this and having arguments. Within rest the data can be presented as XML and JSON. These are different ways to display the data. Kind of like a word document or PDF fail. XM L is great because it keeps the fidelity of the data. It's more heavy weight. So if you are going within a data center or from server to server, you might want to look more towards XML. A lot of web developers are looking towards JSON. Less data there that is needed to describe the data. So when you are building a mobile app or something like that, a lot easier to use JSON. Job description, every language used is needed JSON capabilities. You might see JSONP, it's just a form of JSON that makes it more secure to use from web site to web site. Those are some of the terms you might hear. Developers tend to be agnostic as to which one of those rest full APIs to use. I prefer JSON. That's the direction things are heading. But as long as you put the data out there, people know how to use it.

>> Going to talk about API keys. It's actually there are several different options. You can use keys and not use keys. Really, we want you to have an overall understanding of why the discussion includes that subject matter. If we could kick off to you, Mike.

>> Sure thing.

>> All right. I'm going to give a little overview on API keys.

>>> A couple things on API keys. There's really four reasons you might want to consider using API keys. And those are authentication, authorization, rate limiting and analytics tracking. Do you want to protect the information and limit it only to people who you've approved who can use it? Do you have information you want publically or do you want to limit who can access the APIs to certain set of people. Related concept is authorization and that is do you have multiple APIs and want to have one account system but only give access to a certain set of users. The concept of rate limiting is related but it's a little different. If you want to provide access to everybody but some protection on usage. May have limited set of servers or something where you need to limit how many calls the API people can make. And last is analytics tracking. You just want to know at a more granU lumbar level -- GRANULAR level. You want to have a sense how people are using your API over multiple calls. These are kind of the four reasons that you may want to think about an API key. I think taking a step back, the real question I think in the government space is is there a really strong business case for APIs. One of the challenges is API keys can make it challenging depending how complicated you go. At a simple level, API keys can be a straight forward process. But fundamentally if you want API to be used, you need to have an easy way for developers to have access. Make it publically available without any sort of limits on it. But then understanding that may not be the case. Thinking through the different approaches to security and analytics is going to give you variations of API keys on the spectrum. I will talk a little about security. A couple things that I like thinking about is the first when you are requiring developers to sign up for API key, consider using third party user management. Getting them to sign up with credentials they already have. You've all seen that third party. That's a great way to get the information you need or get a unique ID for each user you can offer an API key for. But it keeps you out of the user management world which carries its own set of challenges. One of the other things I'd recommend around API keys only rely on OAUTH. You need to do some OAUTH authentication and authorization. It can be a great solution when you've got data from users you need to protect but for this read only APIs, really overkill and can be a really big barrier we'll all like to see. Neat and interesting things and OAUTH can be a big pain to implement. And the challenge with this all is balancing the barrier to entry for developers using your API with whatever security needs existed by the organization and security needs are something we need to recognize and brace. However, it's important to understand they do pose this barrier of entry. There's a balance between the two and without diving into it much longer, these are kind of the general approaches we use for API keys.

>>> I will pause.

>> We'll add that we did choose to use a key for many of those reasons. Not really for authorization. Cause it is all publically available data. But for analytics and for the ability to throttle or restrict access if there were any security reasons to do that.

>> The only thing I would add on is don't make the assumption that you need API keys. All the arguments for them is spot on. In government, the problem we have is getting people to consume our concept. Not people over consuming our concept. One model might be to wait before you decide that you need API keys. Or if you do, the one model I really like is having a tiered access level so even if I don't authenticate, I can just go to my URL, open up my browser and type in a URL and makes it easy. I can get back a list of all the tweets or something like that. And if I want to step up my usage I I can provide API key. There's lower barrier entry as possible from a developers stand point and kind of restriction and various entry as needed in order to ensure service.

>> As always be collecting metrics.

>> This is Lisa again at census. Just to talk a little about what we have available. Metrics over various times. We can look at metrics by the hour if we wanted to. We look at the number of keys requested and the number of keys that have been validated. We use an e-mail validation system pretty much like things you've seen in other places that way we can track what % of requested keys are not just validated for one reason or another. We can look at the calls to the API and see how to use that as a measure of volume to the API and we also have a forum that we make available through idea scale for developers using the API where they can get support, report bugs, make future requests. They can share with us their applications that they've developed off the API. And from that, we can get the members on that forum, the traffic, the posts, et cetera on that. So all of those together give us a pretty good picture of the interest and usage in the API.

>> Mike, from your perspective over doing a lot of content via API at the federal communication's commission, you know, what role were you able to play in analytics in that perspective?

>> There's interesting approaches. One of the cool things I've seen is you can actually push events back to Google analytics for an API. And download blank on a web site, sort of pushing it back. Even though Google is not tracking it directly. There's a number of different ways you can approach analytics and there's obviously a number of great providers out there who are starting to dive into the different ways but what ways are they using it and it's easy. Hopefully the goal is easy to crosswalk data. User engagement on a web site carry over to an API. If you are doing it well and successfully, your users aren't going to be making one call and dropping off. They'll be making one call that leads to another call. And a well designed system would give you an insight into usage patterns that does well to both the example of the recent roll outs from census and sometimes what happens when your API has become popular. Can you kick us off on that front?

>> Sure. This is Bill again. Basically, we had to think a lot about the scaleability of this in terms of managing for growth. When we had -- we already had infrastructure in place that did certain

things and the idea was that we might have somewhere in the neighborhood of maybe 50 concurrent users at one time for the systems. When we were talking about APIs, we were talking more in the terms of growing for 1,000 concurrent users. So things that we had to do, we ended up doing things where we did he coupled -- DEcoupled a lot of the pieces. We had something that dealt with the API things, the services themselves where you are doing key validation and queery validation. We'd have separate databases. And in the middle of all this was having a load balancer that it was doing things like the throttling that we were talking about earlier and also this allowed us to put more servers with the same services on it where we determined that we needed just more growth or we just -- if we determined we had too much of a load, then we could add another server on there that would be able to be balanced either on the API or data service side.

>>> This is a diagram of our current set up basically. The client comes in and talks right away. There's throttling done by key and IP address. So it's basically to prevent service attacks. Goes to API servers on the left. And you can think of it as [Inaudible]. The low balancer does something where it's more smart about it. Actually checks to see whether the server is responding to make the determination. The request goes over there and gets returned back with a data request. Going basically from the left-hand side of the server over to the data servers. If goes through the low balancer so that it can go to one of several of the data servers. The data server then can talk to one of multiple database servers and then returns back to the low balancer which goes back to the API server which reformats which goes back up to the client.

>> Very helpful thinking about the fact there's two sides to this coin. Building for scaleability but also not preventing projects from taking place by being overly cautious. One of the advice we give people is prototype and allow yourself to fail. Fail fast. Because at the end of the day, you know, too much invocation too fast is usually not the problem we're facing in government. Problems to experience. It's worth starting with early active prototypes and then -- but it is important to always see scaling issues come up. In the long run, that's something we all shoot for.

>> One way to possibly quickly prototype, depending on the data set. A lot of data sets aren't updated as a Facebook update. Updated weekly or monthly or daily. Depending on the structure of the data set, might be possible into multiple text files. Multiple XML files so you mimic the effect and the public might think there's a database driving it but it's posted on a static file server. This is different than putting out a single file. But mimicking what would be the output of an API calls by creating 10, 50, 100 files might be a way to high scaleability and low cost and see if it's being used and how it's being used.

>> That's a very good point and it was eye opening on my part to understand that with issues of scaling, there's really a binary difference between if there's a processer that's having to run in order to deliver the product back to the consumer. If you have your demands for the system processing things in order to deliver the page, that puts scaleability front and for most. But when you can have flat files serving just like a hosted text file or small code file those can scale better.

Data that doesn't update but once a month or once a year. All of these things are choices. Do you go with XML? Or JSON you need to choose do you start with API key or not?

>> There's a very important component to a developer that comes along. One of the biggest points of failure in dealing with an API is they have to work to figure it out. In order to make yours a success, you want to eliminate that work. That gets to the point of documentation.

>> This is Mike again. I'm going to do a couple things here. Give an overview of what we see as good documentation and I'll walk through a couple examples that I really like. So I think with any set of good documentation, it falls on the spectrum. From very high levels to GRANULAR. And why I put this at the top of the list, people forget the high level overview of the API, what the purpose is, how it's architected and the create TER visioned it being integrated is a critically important part of any documentation. So giving that high level overview at what is the API system and what's it to be used for, gives the context to developers to use it correctly.

>>> I think it's worth spending time and putting this together. Sometimes it's as simple as architectural overview. Other times you might want to think about particular use cases to highlight so that people have a sense of how they should be thinking about using the API and all the other information that's included in the documentation.

>> The second piece that's important is if you have authentication or an API key. Maybe a clear and present on almost every page what the authentication scene is and what the developers need to do to get access to it. The important fact of this is you need to talk about what might they run into when authentication goes wrong. Everything works great, here's what you can do with our API. But if there's rate limits, how are they going to know? Every API developer is trying to work with Twitter has run into the really annoying lack of response initially when you would hit rate limit. So making clear the expected returns in all of these situations is really important.

>>> This is for more GRANULAR, you are going to want to talk about input formats and parameters. What are you expecting and this goes for whether you take updates approach API. You need to tell your developmenters what your input is. And same on update. What are the options, what's the developer going to see? And how are they supposed to interpret the return in various situations. The final thing again that's really important and sometimes gets overlooked is this is air conditions and return. When something goes wrong, there's been a server error. What is going to be returned? Using HTTP status code? Actually return an error pay load? I think the best practice on errors and APIs is be as descriptive as possible. Give them as much information they need to DEBUG. And straight forwardly document what the air conditioned and returns are. There's three examples on the the scale of enterprise to not. Enterprise that I like walking through. I think they do a good different job of API documentation.

>>> The first one, Google does an excellent job of documenting all of their individual APIs as well as talking and telling a story about how the APIs work across the services. There's a good narrative flow to the consistent ways you are going to interact. And what this is nice for especially for developers is you learn one and you know how all of the Google APIs work. And all follows the same format. Very simply, this is the kind of overview for the Google drive API. They give you the end points and tell you what you need to do and link through to what the operations are. Comprehensive but also pretty clear. One of the other things just as a ground rule, the more examples you include in your API documentation, the easier it's going to be for developers to. So not just generic example of how to do the documentation or how to use the service, but they actually give you in different languages the different example codes. This is a really powerful way to kind of ensure your developers get a jump start and when you can include it in the documentation pages. It's even more helpful than giving a reference implementation or a package of a sample client.

>> The next one is Twitter's documentation. They do a good job in a little different way of Google of breaking down. It's a different types of API where Twitter takes much more of an approach. Splitting out what are each of those parameters and what does it do and what kind of format do you need to get by. So this one is straight forward. This works or you can fill it up. At the low end the light way. The application [Inaudible] they do a good job of a pretty simple API that's easy to use. Easy to consume. And pretty straight forward. There's not a lot of complexity here. There's really only a couple calls you can make with API. But they take a rest full approach and broken it out to talk about account info. How to define the different parts of the API. And then each of these you can click through and you actually get example requests, example response and the same kind of per amateur options. So these are the top three. I think we're at the for front of and there's a couple different approaches is how to do automatic API generation. API documentation. There's a couple technologyies out there that are starting to emerge. One of them that is an idea for having your documentation auto generated and actually immaterial bedding a client in the page for your API. And I think that's the next wave of tools. Now, we've got code examples but really neat if you could actually just do an API career builder. In browser enabled API builder. You can see the output to return and you can see the expected parameters to play around with the API without having codes.

>> Great. Thanks. You definitely included a large amount of documentation when you are rolling out APIs, what are some of the things from your perspective?

>> Thanks. Yeah, the challenge we had was not as much documenting how the APIs worked. They pretty much all worked the exact same way. The challenge we had was documenting the kinds of calls you had to make to get the data out. The community API has 45,000 -- 42,000 different variables that you can get out of it. Total population. People you commute. All different sorts of combinations by age, sex. So getting people to be able to pass those variables in to get the information they wanted including margin of errors and other things. So we had to piggy back off a lot of the existing data that was out there. The other challenge that we found from listening to people on the forum and from some other e-mail feedback was having people understand how the geography works along with the variables that some geography was available. So we tried to focus our effort on data documentation and providing additional

information about census geography. We've not gotten a lot of feedback from people who have had difficulty making the API calls themselves. But more just working with the data sets in general.

>> This is David. Talked about auto generating documentation. We do a subset of that and we generate a piece of our documentation from the CONNFIG files and makes sure it mirrors.

>> I won't add too much. It's not just about the code and the API calls. Forums and list serves. And engaging people as human beings and not as API consumers. And two to make it sexy. Use it as a marketing web site. Not simply you are talking to engineers. If you are lacking for a great example of this. Developer.ESPN.com. It's totally done by their marketing if I recall. And -- marketing firm and really well done. And using it as a marketing stage rather than just a shuttle.

>> Great. Thank you. Going back to some of the examples of the work done by the federal communications commission, I would stress to people, a good thing to do is go to the web site but look on the right sidebar. What you'll see is a list of the developer hubs that are growing out at other agencies. And I really recommend taking those as good precedence to follow. And click through them and you'll see the same things. You'll see people working on the same opportunities. Basically, I would try to assure you that this kind of rounds out the picture of what technical aspects there are. And it is a lot more accessible than people realize. One of my favorite examples is Department of Labor's hub. They actually touch on each of these aspects. I know that behind labor's efforts but also others. There's definitely a willingness. Reach out, e-mail the list serve and we can get you in touch with the people doing this and they are happy to help you do the same. Each of these areas, formats, analytics, API keys. All of them have very ready solutions. That's why we're at the point we're at. There will be a number of releases coming out. Very relevant to you and to this conversation. OMB is releasing guidance that will actually or tick late best practices -- articulate best practices and some of the norms. There's plenty of good ways making APIs available. If you can get past some of the questions, settle on something to begin and start prototyping, it will become a lot easier to figure out what works best for you. The other part of it though is GSA will be releasing a number of tools, resources and actual like software to facilitate your creating APIs. It will be rolling out throughout November but cull Monday ating -- CULMONATING. Each executive agency to actually make two APIs by this coming may. I want to stress that for some agencies we're already on track to do that. For a lot of other agencies, there were barriers to getting started on that. Not necessarily real world barriers but just an uncertainty to all what is entailed and how to get started. What we tried to show today was an overview of pretty much the largest and most common technical issues people address. And what we really wanted to convey is the solutions are there. But, feel free, please, we'll have contact info up. I really hope you will reach out and let us support you in the creation of APIs at your agency. Because it really is worthwhile.

>>> So we're going to switch to the question and answer session. Some of you have put in questions throughout the event. I will get to those and if other people have questions, please send them in and I'll be happy to answer them.

>>> The first is for the census bureau, what percentage of your keys are issued after authentication? Rather, is it representing a barrier? Do you have any impression as to API key requirements are lowering the number of users using API?

>> Yeah. This is Lisa. I don't think it's a huge barrier. I think we're -- I should have looked today anticipating that question. But I think out of 1600 questions, 1500 were validated. I think it runs very close. So I don't think it's really a barrier. And we also do use that opportunity as an engagement and tell people in the key validation how to get information and how to follow us and what not. So I don't think it's a barrier. I'll ask the guys.

>> I don't think so either. This is Dave. One other thing I'll throw out there [Inaudible]. And there are plenty of services out there that allow you to basically have a temporary e-mail address. And anybody who doesn't want to give us the information will use the services. We haven't seen any problem.

>> That was actually also one of the questions that we asked out on our forum is how was the key registration process for you. And we've really not gotten anything too negative except a couple times if the service is down for some reason. People let us know. But for other than that, there hasn't been any negative feedback on the process.

>> Great. Mike, I'll throw this one at you. How have you used the API metrics that you've received with analytics to change design decisions or other aspects of your services?

>> Sure. I think the important thing to note there is metrics are a starting place, not an end. Analytics are anything that rolls up gives you a direction to start looking. But just like when you are developing a web site, when you are developing API, the most important thing is to talk to your users. We've talked today a bit about having forums and ways for users to contact you and provide feedback. I actually like the process of using your analytics to understand where is it successful and where are you seeing things that tend to -- people tend to drop off or aren't successful. A lot of errors or requests or something like that. And then actually go find users to talk to and get some feedback on what you can do to improve.

>> Census, what about you?

>> For our analytics, we've had our API out. We did a closed beta test in April. And then we did an open beta test. We launched at the end of July. So for right now we're using our analytics primarily to try to learn how people are using the system, if there's volume we can handle. How we have to plan. Things like that. We also can look at which data set is more heavily used than the other. Right now we're primarily using the analytics to speed into planning for the next wave.

>>> When you were showing the load balancers during the scaleability question, the question was if this was hosted in the cloud or locally?

>> It was hosted locally.

>> We heard his favorite documentation section was ESPN. And Mike showed some of his examples in there. When you are looking at documentation, where did you take cues or who do you annotate?

>> Yeah, this is Lisa. I don't know that we specifically imitated anybody. I knew we would have the challenge of getting people to data set information so that they could learn about what was available through the API. The data sets that we have up there are very large and offer so many different variables. So we were looking for a way to segment the documentation between here's how to use the API but even more importantly, here's what you can get out of the API. And how to try to find out what that appropriate variable is for your need. But we did look at labor and FCC as two examples that we thought did a really good job at documenting their APIs.

>> This is Ben. The one thing I have to tip my hat to Mike to is there is a handful [Inaudible] Google just opened doors there. API documentation. But that would allow to query the server. So I can dynamically just from the web site make a query in the browser and see the response. And allows me to get a feel and tryout the APIs without even leaving the web site.

>> So that actually touches on one of the more metaaspects here. And that is when trying to decide all what to include in API roll out as far as features or documentation or making decisions, single biggest thing you can include is [Inaudible]. We didn't include that here. Not so much of a technical functionality. But having right next to your APIs as part of your developer hub a means for the public to give feedback or ask for help or communicate with you is definitely an appreciable aspect. One of the best overviews I've come across is to get a chance, I've been reading a lot lately APIEVANGELIST.com. One of the best overviews I've seen to explain what APIs are and give a brief overview of the technical background. So if you want to follow up on this conversation and kind of wrap your head around it a bit more, just read this page and it should put you in the right place to at least begin having the conversation within your agency.

>>> Are there any other questions coming in?

>> No, there are not.

>> All right. So a couple more aspects that I guess I am curious about and that is Mike, some of the work you've done touches on the role of content, web content. Like the actual text of a page as an API as opposed to more data heavy or numbers heavy APIs. Are there any differences you found in any of these arenas, keys, analytics, et cetera that vary when you are dealing with content?

>> That's a great question. I think the short answer is not really. It's important to recognize [Inaudible] it's just another type of data. It's text instead of numbers. The same tools and techniques you use with APIs we can use with constant APIs. You think about scaleability with HTTP based APIs where you don't need an API key, you can use the same techniques for an API as well. Service will handle an eye call. We were able to play around with the tricks. The other thing that was important with content was provide a good way to get to content. On an API you want to be able to go to a single piece of content to a list of content that's of a related type.

So within the API and take a look at Google. They do a good job of this and provide end point URLs. So when you list a set of tags. Part of the return for the tags is the URL for going and grabbing the content list.

>> That appears to be all the questions we've had come in all this time. I want to thank everyone forgetting on today and provide context. There's some URLs and e-mail addresses here. I want to encourage you if there's something you need or want, if you need help talking with other people, let us know. If you are in a position of managing a project or having some connections with those, it's the right time to be asking yourself what's it take to get started. Each of these five arenas will come up in the process. Basically the answers are at land. There's a lot more information that's coming up. So check back on that. But otherwise, the private sector does a great job of showing us what the answers are. Developers are clearly around and that's worth imitating. It's a refreshing time in government to be able to actually follow the lead and act as the private sector acts. Cause it just works well. Thank you all for attending. And the video of this is going to be posted afterwards. There will be project series coming up in mid November for the webinar. But also I want to put in a quick plug, next week on Thursday, there is an in person conference we are hosting at GSA around APIs that will be much more in depth. Please go to howto.GOV/training and you should see an event that will have a registration form. We'd love to have you if you are in DC. And if there are other people at your agency who you want to show and introduce them or even technical people and you want to connect them with other agencies, please join in. Welcome everyone from agency next Thursday afternoon. Thank you all forgetting on today. And have a good afternoon. getting on today. And have a good afternoon.

>> [Event Concluded]