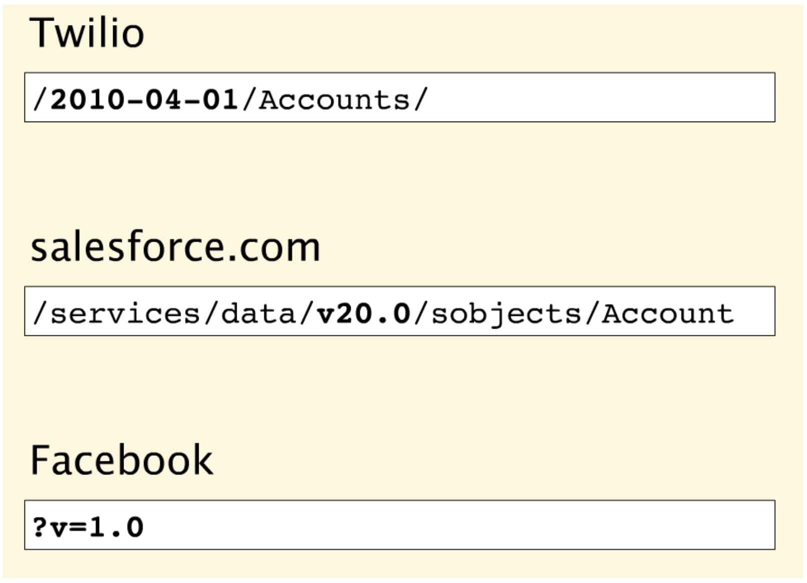
In the last post in this series about Pragmatic REST API design, I talked about designing error conditions in your APIs. Check out the full series here (/taglist/RESTful).

This time - **versioning** - one of the most important considerations when designing your pragmatic API.

Never release an API without a version and make the version mandatory.

Let's see how three top API providers handle versioning.



Twilio

```
/2010-04-01/Accounts/
```

salesforce.com

```
/services/data/v20.0/sobjects/Account
```

Facebook

```
?v=1.0
```

Twilio uses a timestamp in the URL (note the European format).

At compilation time, the developer includes the timestamp of the application when the code was compiled. That timestamp goes in all the HTTP requests.

When the request comes into Twilio, they do a look up. Based on the timestamp they identify the API that was valid when this code was created and route accordingly.

It's a very clever and interesting approach, although I think it is a bit complex. It can be confusing to understand whether the timestamp is compilation time or the timestamp when the API was released, for example.

Salesforce.com uses v20.0, placed somewhere in the middle of the URL.

I like the use of the **v.** notation. However, I don't like using the **.0** because it implies that the interface might be changing more frequently than it should. The logic behind an interface can change rapidly but the interface itself shouldn't change frequently.

Facebook also uses the v. notation but makes the version an optional parameter.

This is problematic because as soon as Facebook forced the API up to the next version, all the apps that didn't include the version number broke and had to be pulled back and version number added.

How to think about version numbers in a pragmatic way with REST?

> Specify the version with a **'v' prefix**. Move it all the way to the left in the URL so that it has the highest scope (e.g. **/v1/dogs**).
>
> **Use a simple ordinal number** - v1, v2, and so on. Don't use the dot notation like v1.2 because it implies a granularity of versioning that doesn't work well with APIs--it's an interface not an implementation.
>
> Make the **version mandatory**.
>
> How many versions to maintain? (http://blog.apigee.com/detail/restful_api_design_how_many_versions/)

Next time -Giving developers just the information they need - pagination and partial response (http://blog.apigee.com/detail/restful_api_design_can_your_api_give_developers_just_the_information/).

Please join the API Craft (https://groups.google.com/group/api-craft) conversation on Google groups.