ARIN's RESTful Provisioning Interface

Tim Christensen

Pull up a chair, let's REST a while

- What is REST?
- What's ARIN's RESTful interface?
- What does a RESTful interface buy me?
- How does REST improve automation?
- Where can I learn more about ARIN's RESTful interface?
- Can I do that now?

What is REST?

- Representational State Transfer (fancy talk for an architectural style of designing software systems that use Web technology to make defining, addressing, and exchanging data easy)
- Commonly implemented by using a URL along with XML-formatted data

ARIN's RESTful Interface

- Uses a URL containing 3 parts:
  - **The address (you're used to this one)**
  - **The method (a basic instruction to the server that tells it what you want to do)**
  - **A resource identifier (basic information about the record (resource) you want to affect**
- But wait… there's more!

ARIN's RESTful Interface

- Also requires (in most cases) a payload
  - **Contains the data you want to send**
  - **Data format we're using is XML**
  - **Format and structure is defined by a schema**
  - **Schema is published by ARIN for each type of record (resource) in ARIN's system: think template, only more strictly defined**

What does REST buy me?

- As a human, not much
  - **Can be difficult to read, write, and interpret (it isn't intended for humans, anyway)**
  - **Relatively unforgiving and unfriendly**
- As a computer, plenty
  - **Predictable input and response mechanism**
  - **Discrete set of responses**
  - **Instant gratification**
  - **Secure, authentic communication**

REST Improves Automation

- Schema provides a precise definition of expectations
- Methods and payloads define the contract for interaction
- Direct connection to the RESTful service provides immediate response
- Enables more efficient and more complex interaction with ARIN systems

What is REST not so good for?

- One-off activities
- Activities best supported by guides, wizards, examples, or explanations
- Interactions requiring a lot of back-and-forth human communication
- Interactions requiring out-of-band confirmation

…but REST is great for…

- Repetitive,
- Mundane,
- High-volume activities
- Interactions requiring no human communication

…..like SWIP

Why SWIP is a good fit for REST

- Discrete format definition thru schemas
- Concise and immediate response
- Stateless (all-in-one) communication
- Scalable
- Secure (via HTTPS and API Keys)

Where do I learn more?

- ARIN's website:
  - https://www.arin.net/resources/restful-interfaces.html
  - https://www.arin.net/resources/restful-methods.pdf
  - https://www.arin.net/resources/restful-payloads.pdf
- ARIN Technical Discussion Mailing List
  - **arin-tech-discuss@arin.net**
  - https://www.arin.net/participate/mailing_lists/index.html#tech

Where do I learn yet more?

- OT&E (Op Test & Eval) Environment
  - https://rest-beta.arin.net
  - **Not email-enabled**
- Experiment using tools for interacting with ARIN's RESTful interface
  - **wget, curl, and xmllint are sufficient**
  - **Google Chrome extension**
  - **I like WizTools.org's REST Client for Mac**
  - **Learn thru one-off experimentation before writing programmatic interactions**

Today's Tutorial

- A tool for accessing ARIN's RESTful interface that is palatable to humans
- Basic requirements:
  - **ARIN Online Account with an API Key**
  - **The usual authorization to act for an Org**
  - **The Methods document**
  - **The Payloads document**
  - **A little attention to detail and persistence**

Methods and Payloads

- Basic tips for methods: follow URL rules precisely
  - **CAsE MatTers!**
  - **The space within becomes the reality of a failed interaction (apologies to F.L.Wright)**
  - **API Key? I don't need no stinkin' API Key! (um, yeah, you do. apikey=API-xxx……)**

Methods and Payloads

- Basic tips for payloads: the XML schema is your friend
  - **<tags> must be correct and present </tags>**
  - **The payload must be attached to the URL using "application/xml" content type**
  - **Best to use GETs first before performing other methods on existing objects**

Live Tutorial: Viewing data

- Seeing what resources are in ARIN's system: GET
  - **Requires a URL (and the GET method)**
  - **Payload must include identifiers**
  - **Authentication via apikey=API-xxxx…..**
  - **Follows ARIN's authorization scheme**
  - **Does not require any payload**
  - **Returns a payload containing the resource**

Live Tutorial: Updating data

- Updating resources: PUT
  - **Requires a URL (and the PUT method)**
  - **Requires identifier and apikey= like a GET**
  - **Requires a payload (GET it first) and modify**
  - **Must be content type application/xml**
  - **Follows ARIN authorization scheme**
  - **Returns a payload containing the resource, complete with newly updated data**

Live Tutorial: Creating data

- Creating new resources: POST
  - **Requires a URL (and the POST method)**
  - **Requires NO identifier, but needs apikey=**
  - **Requires a payload (application/xml)**
    - All required elements
    - No system-generated elements
    - It is helpful to use another payload as a sample
  - **Returns payload containing new resource**

Live Tutorial: Removing data

- Deleting resources: DELETE
  - **Requires a URL (and the DELETE method)**
  - **Requires an identifier and apikey=**
  - **Does not require any payload**
  - **Follows ARIN authorization scheme**
  - **Returns payload containing the resource that was deleted**

Questions?