



Fall 2014

# Apigee Advanced Developer Training

Workshop: SOAP to REST

## Designing a RESTful Interface

**Goal:** Create a RESTful Interface to interact with a legacy SOAP Interface

**Actions:**

- Design REST Interface
  - Resources and how they relate to standard OOP
  - Wizard versus policy based implementation
  - Build Object Model

JSON

REST  
SOAP

XML

# Designing a RESTful Interface

## New API Proxy

### 1 Choose Your Starting Point

Extracting from API Proxy Bundle Completed

Starting Point Type\* ☐ Backend Service ☐ API Bundle ☒ WSDL ☐ No Target ☐ New Node.js ☐ Existing Node.js

WSDL Source\*  [Change WSDL Source](#)

API Proxy Type\* ☒ REST to SOAP to REST ☐ Pass-Through SOAP

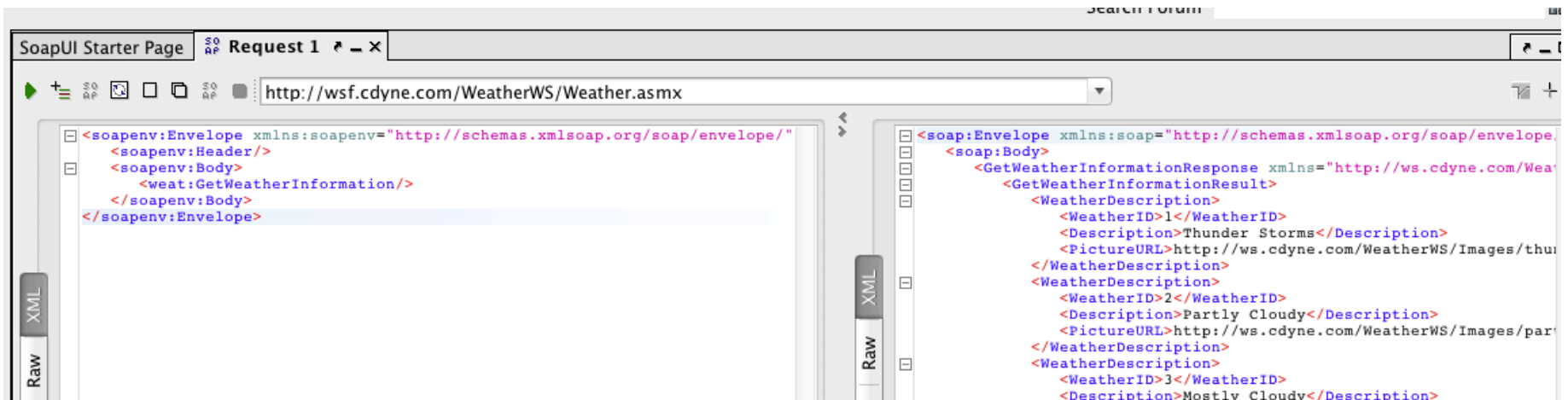
API Proxy  
Discovered from WSDL  
9 Operations

Include	WSDL Operation	Description	HTTP Method	REST
<input checked="" type="radio"/> Port Type: WeatherSoap				
<input checked="" type="checkbox"/>	GetWeatherInformation	Gets Information for each WeatherID	GET	weath
<input checked="" type="checkbox"/>	GetCityForecastByZIP	Allows you to get your City Forecast Over the Next 7 Days, which is updated hourly. U.S. Only	GET	cityfor
<input checked="" type="checkbox"/>	GetCityWeatherByZIP	Allows you to get your City's Weather, which is updated hourly. U.S. Only	GET	citywe
<input type="radio"/> Port Type: WeatherHttpGet				

## Resources

Name	Proxy Endpoint	Method	Path	URL
▶ GetWeatherInformation	default	GET	/weatherinformation	.../weather//weather
▶ GetCityForecastByZIP	default	GET	/cityforecastbyzip	.../weather//cityforec
▶ GetCityWeatherByZIP	default	GET	/cityweatherbyzip	.../weather//cityweat

# Designing a RESTful Interface



# Designing a RESTful Interface

## GET /weather\_soap/weatherinformation HTTP/1.1

```
{
  "GetWeatherInformationResponse": {
    "GetWeatherInformationResult": {
      "WeatherDescription": [
        {
          "WeatherID": 1,
          "Description": "Thunder Storms",
          "PictureURL": "http://ws.cdyne.com/WeatherWS/Images/thunderstorms.gif"
        },
        {
          "WeatherID": 2,
          "Description": "Partly Cloudy",
          "PictureURL": "http://ws.cdyne.com/WeatherWS/Images/partlycloudy.gif"
        },
        {
          "WeatherID": 3,
          "Description": "Mostly Cloudy",
          "PictureURL": "http://ws.cdyne.com/WeatherWS/Images/mostlycloudy.gif"
        }
      ]
    }
  }
}
```



## Exercise 1: Create a Weather proxy using the wizard

### Instructions:

- Create the proxy using the sample weather WSDL
  - <http://wsf.cdyne.com/WeatherWS/Weather.asmx?WSDL>
- Explore the policies created
- Make a few requests into the created proxy to get familiar
- Inspect the responses

# Designing a RESTful Interface

## Weather WSDL

GetCityWeatherBy  
ZIP

GetCityForecastBy  
ZIP

GetWeatherInformation

Zip

Icons

Forecast

Weather

## Designing a RESTful Interface



A more RESTful design of the API would be:

**GET** /zipcodes/{zipcode}/forecast

**GET** /zipcodes/{zipcode}/weather

**GET** /icons

**GET** /icons/{iconid}



## Building a RESTful Interface



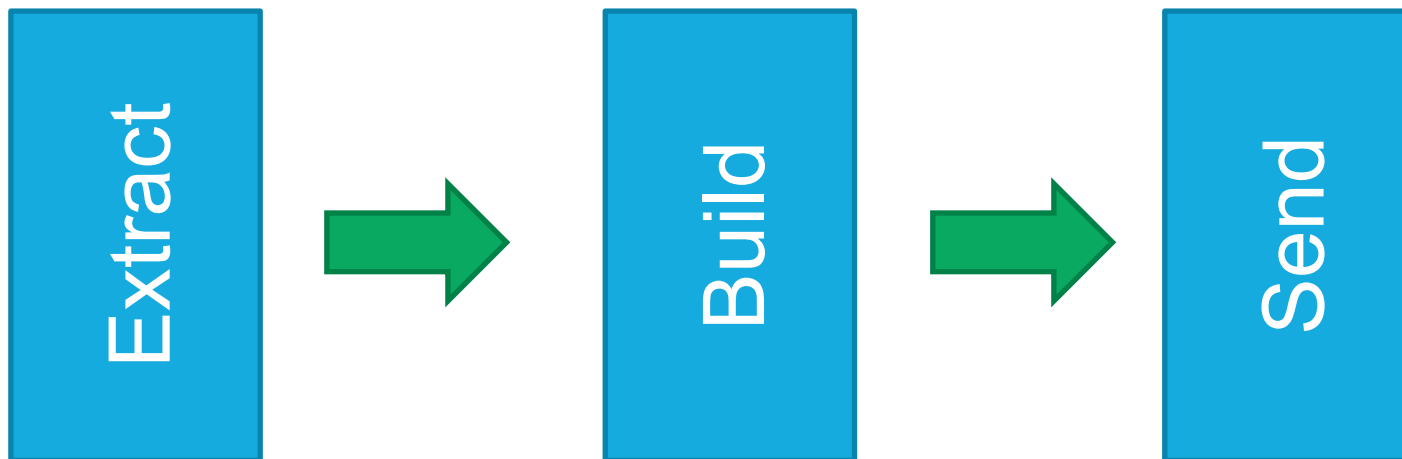
**Goal:** Take our RESTful design and build it out in the Apigee Platform

**Actions:**

- Create RESTful resources
- Extracting Data From the Request
- Build Assign Message Policy
- Copy soap request and parameterize the fields
- Test Requests

## Building a RESTful Interface (cont'd)

- Add Resources (conditional flows) using overview pane
- Extract incoming parameters using an extract variables policy
- Build requests to SOAP service using assign message policies



## Building a RESTful Interface (cont'd)

Extract {zipcode} and {iconid} into variables

Code: Extract-Zipcode

```
4
5     <FaultRules/>
6     <Properties/>
7     <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
8     <Source clearPayload="false">request</Source>
9
10    <URIPath>
11        <Pattern ignoreCase="true">/zipcodes/{zipcode}/*</Pattern>
12    </URIPath>
13
14 </ExtractVariables>
15
```

### Variables

proxy.pathsuffix		/zipcodes/12345/forecast
zipcode		= 12345

## Building a RESTful Interface (cont'd)

Build the SOAP message are built using the extracted variables

```
14      <soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/"
15        <soap:Body>
16          <GetCityForecastByZIP xmlns="http://ws.cdyne.com/WeatherWS/"
17            <!--Optional-->
18            <ZIP>{zipcode}</ZIP>
19          </GetCityForecastByZIP>
20        </soap:Body>
21      </soap:Envelope>
```

## Exercise 2: Creating Resources and Building Requests

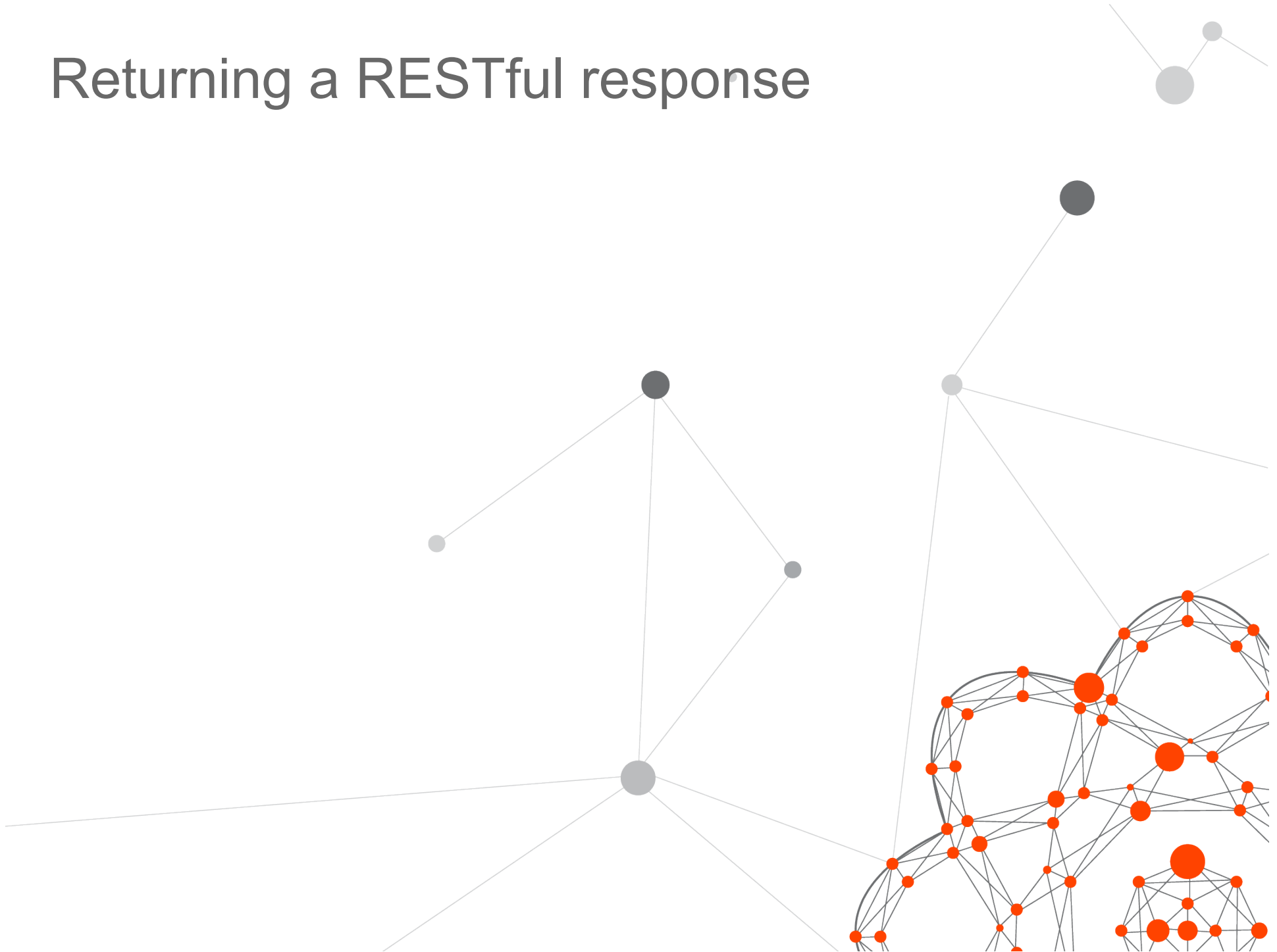
### Instructions

- Create a new revision of the proxy created in the previous exercise
- Modify existing flows to match the design, create new ones as needed
  - GetCityWeatherByZip -> weather
  - GetCityForecastByZip -> forecast
  - GetWeatherInformation -> icons
- Extract zipcode and iconid into variables
- Assign values to request going into backend

## Exercise 2: Creating Resources and Building Requests

Transactions					<<
▲	Status	Method	URI	Elapsed	
1	404	GET	/certification/v1/weather/somethingelse	6 ms	
2	200	GET	/certification/v1/weather/icons	31 ms	
3	200	GET	/certification/v1/weather/icons/4	35 ms	
4	404	GET	/certification/v1/weather/zipcodes	6 ms	
5	404	GET	/certification/v1/weather/zipcodes/95113	5 ms	
6	200	GET	/certification/v1/weather/zipcodes/95113/forecast	227 ms	
7	200	GET	/certification/v1/weather/zipcodes/95113/weather	49 ms	
8	404	GET	/certification/v1/weather/zipcodes/95113/lost	7 ms	

# Returning a RESTful response



## Returning a RESTful Response



### Goal:

Take a SOAP response and transform it into RESTful response.

### Actions:

- Discuss Extracting Data Methods
- Use Xpath to extract data
- Use XSLT to extract data
- Protocol transformation (JSON / XML) – Accept header



## Returning a RESTful Response

### FROM

```
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  <soap:Body>
    <GetCityWeatherByZIPResponse xmlns="http://ws.cdyne.com/WeatherWS/">
      <GetCityWeatherByZIPResult>
        <Success>true</Success>
        <ResponseText>City Found</ResponseText>
        <State>CA</State>
        <City>San Jose</City>
        <WeatherStationCity>Hayward</WeatherStationCity>
        <WeatherID>4</WeatherID>
        <Description>Sunny</Description>
        <Temperature>71</Temperature>
        <RelativeHumidity>62</RelativeHumidity>
      </GetCityWeatherByZIPResult>
    </GetCityWeatherByZIPResponse>
  </soap:Body>
</soap:Envelope>
```

### TO

```
{
  "Success": "true",
  "ResponseText": "City Found",
  "State": "CA", "City": "San
Jose", "WeatherStationCity": "Hayward",
  "WeatherID": 4, "Description": "Sunny", "Temperature": 71,
```



## Exercise 3: Build the Response

---

### Instructions

- Clean up responses from the SOAP services
- Conditionally return either JSON or XML based on Accept request header
- Requests to /icons returns the full collection, while /icons/{iconid} only returns the data for the requested iconid

## Exercise 3: Build the Response

### GET /certification/v1/weather/icons

```
{"WeatherDescription":[{"WeatherID":1,"Description":"Thunder Storms","PictureURL":"http:\\\\ws.cdyne.com\\WeatherWS\\Images\\thunderstorms.gif"}, {"WeatherID":2,"Description":"Partly Cloudy","PictureURL":"http:\\\\ws.cdyne.com\\WeatherWS\\Images\\partlycloudy.gif"}, ...
```

### GET /certification/v1/weather/icons/4

```
{"WeatherID":4,"Description":"Sunny","PictureURL":"http:\\\\ws.cdyne.com\\WeatherWS\\Images\\sunny.gif"}
```



# Thank you

An abstract geometric pattern consisting of white dots of varying sizes connected by thin white lines, set against a solid orange background. The pattern forms a series of interconnected triangles and polygons, with some dots acting as vertices and others as internal points or isolated nodes.

**apigee**

Fall 2014