apigee

Fall 2014

Apigee Advanced Developer Training

Workshop: SOAP to REST

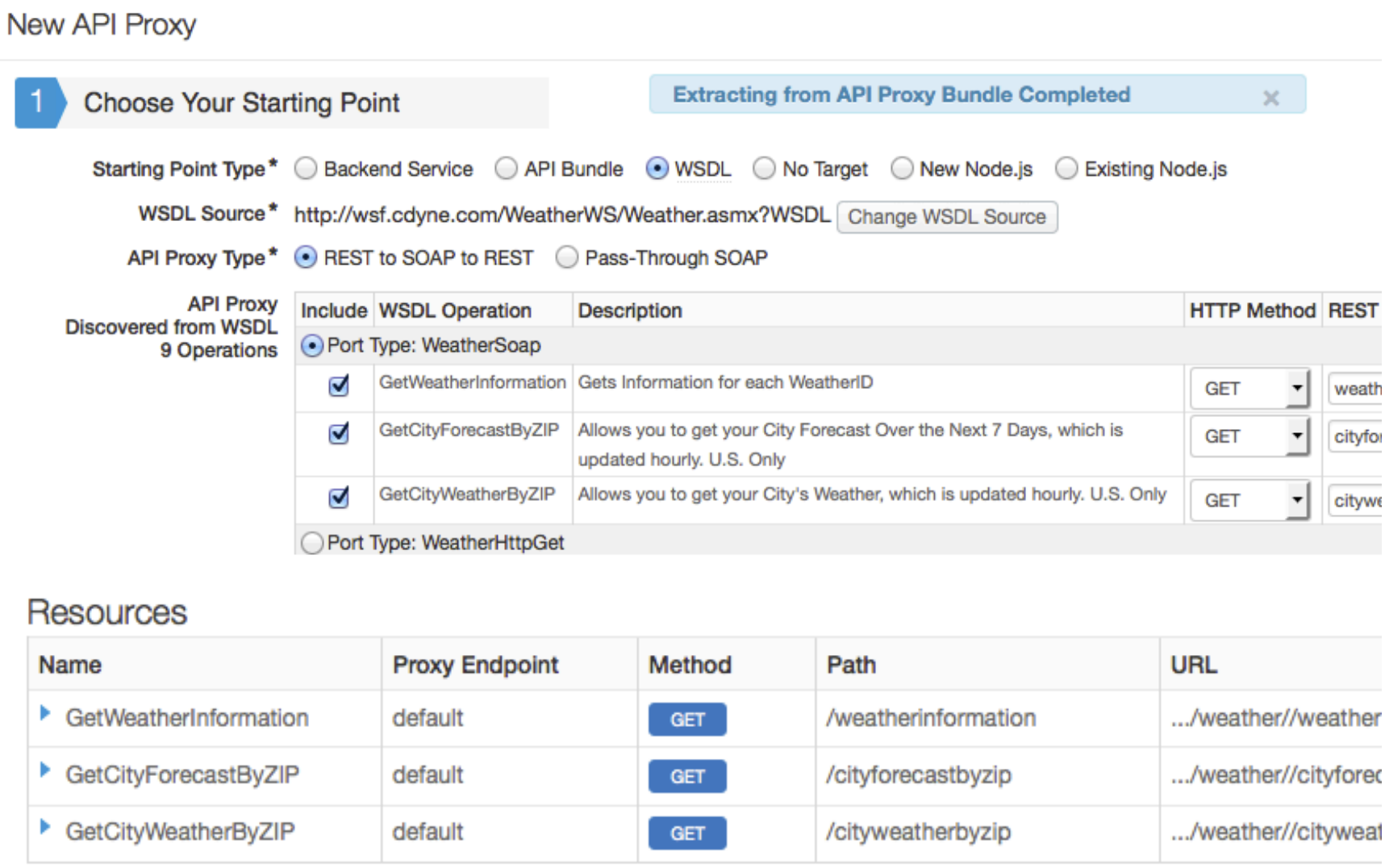
Designing a RESTful Interface

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

Designing a RESTful Interface



Designing a RESTful Interface

Designing a RESTful Interface

GET /weather_soap/weatherinformation HTTP/1.1

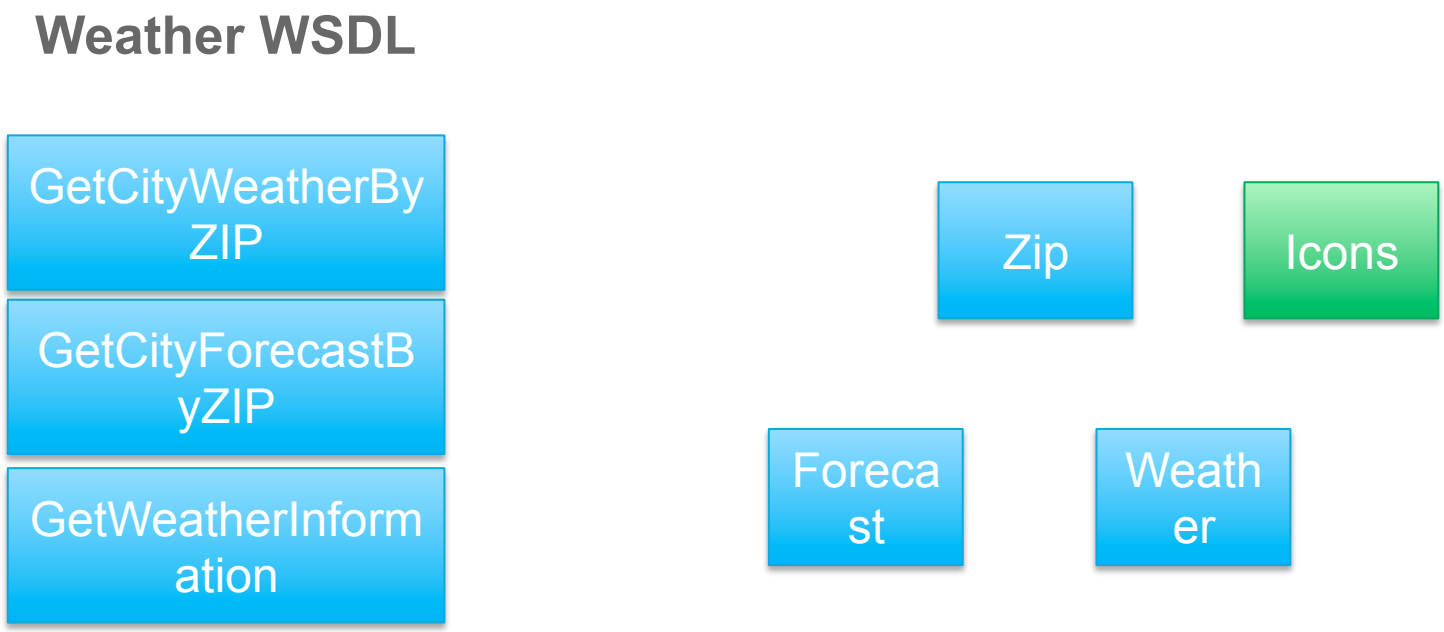

```
{
  "GetWeatherInformationResponse": {
    "GetWeatherInformationResult": {
      "WeatherDescription": [
        {
          "WeatherID": 1,
          "Description": "Thunder Storms",
          "PictureURL": "http://ws.cdyne.com/WeatherWS/Images/thunderstorms.gif"
        },
        {
          "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



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/{icondid}

Building a RESTful Interface

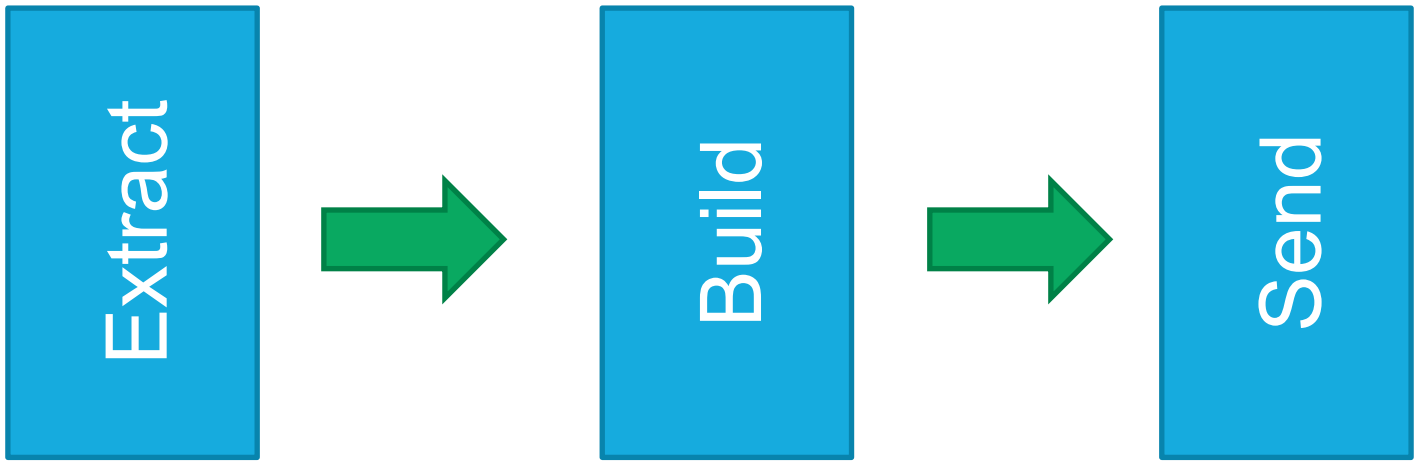
Goal: Take our RESTful design and build it out in the Apigee Platform

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

```
    <FaultRules/>
    <Properties/>
    <IgnoreUnresolvedVariables>true</IgnoreUnresolvedVariables>
    <Source clearPayload="false">request</Source>

    <URIPath>
      <Pattern ignoreCase="true">/zipcodes/{zipcode}/*</Pattern>
    </URIPath>

</ExtractVariables>
```

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=
   <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>
```

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

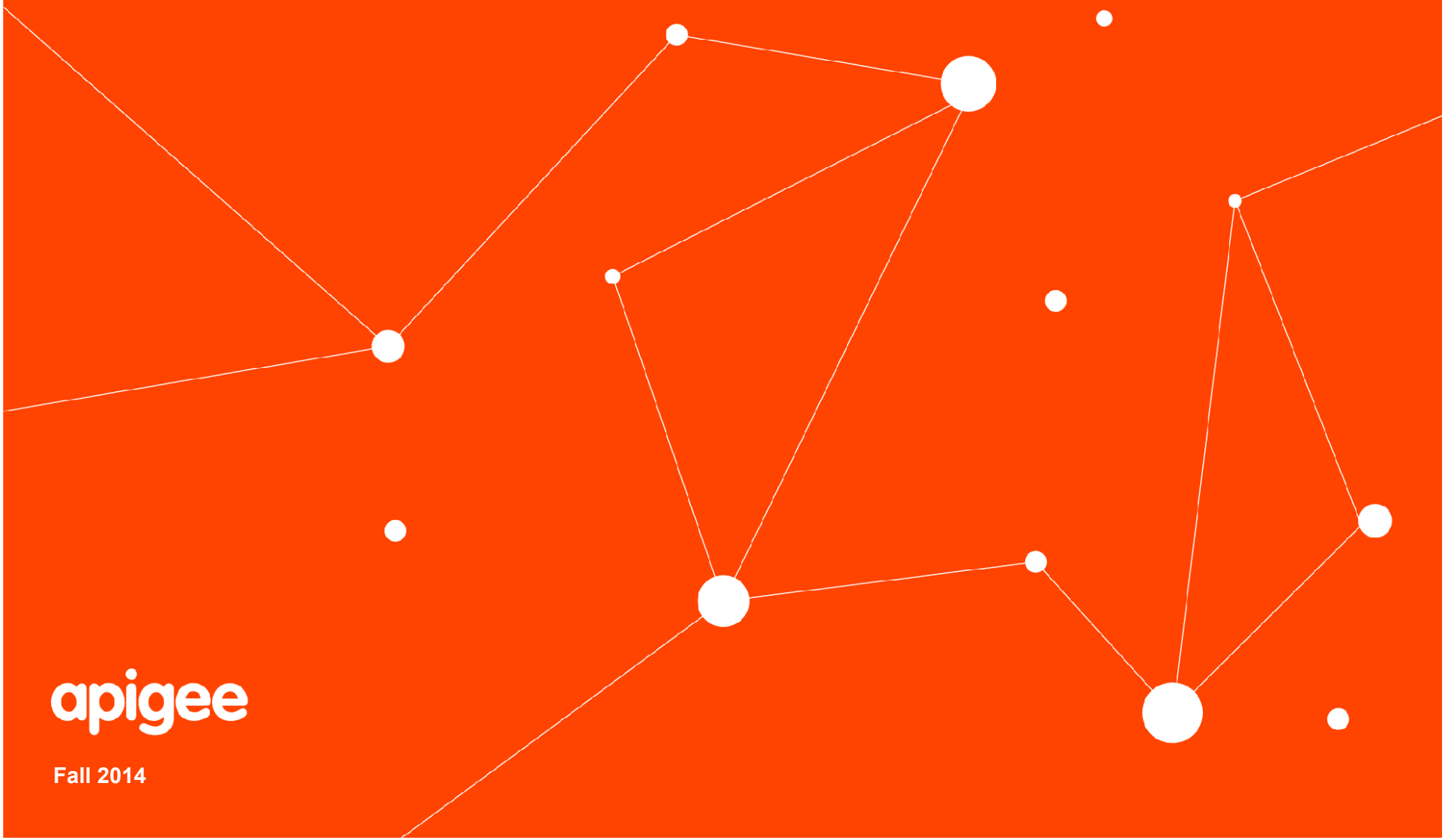
GET /certification/v1/weather/icons


```
{"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



apigee

Fall 2014