Advanced Mediation

Google Cloud

XML/JSON Differences

- XML has a root element, JSON does not need to
  - JSON can have an unnamed object or array at the top level
- JSON is typed, XML is not

  { "aNumber": 14, "aString": "str", "aBoolean": true, "aNull": null }

  <StringOrBoolean>true</StringOrBoolean>
  <StringOrNumber>14</StringOrBoolean>
  <StringOrNull>null</StringOrNull>
- JSON null and null string are different, XML no distinction
  - "" vs. null
  - <XmlNull></XmlNull>
- XML has attributes, JSON does not
- XML has namespaces, JSON does not

apigee

XMLToJSON and JSONToXML

- XMLToJSON can try to guess data types, but it can be wrong
  - Configurable: RecognizeNumber, RecognizeBoolean, RecognizeNull
    - Still may guess wrong (I want ZIP codes to be strings, but they will be made numbers if RecognizeNumber is true)
  - Has to guess for arrays of objects
    - 0 objects = null (want it to be [ ])
    - 1 object = { ... } (want it to be [{ ... }]
    - 2 objects = [{ ... }, { ... }]

- JSONToXML
  - Easier direction
  - Can use JavaScript/Python to manipulate JSON before converting, or use XSL to manipulate XML after converting

JSONToXML Details

Policy Options

The options for this policy can be used to handle scenarios where JSON does not convert to valid xml or you want to customize the output of the transformation.

Example JSON: { “Some%%%%Name”, “Steve”}

Converted : <Some%%%%Name>Steve</Some%%%%Name> --Invalid XML

You can use **<InvalidCharsReplacement>_</InvalidCharsReplacement>:**

Customized Conversion: <Some_Name>Steve</Some_Name>

Additional Policy Options

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<JSONToXML async="false" continueOnError="false" enabled="true" name="JsonToXml">
    <DisplayName>JsonToXml</DisplayName>
    <Options>
        <NullValue>NULL</NullValue>
        <NamespaceBlockName>#namespaces</NamespaceBlockName>
        <DefaultNamespaceNodeName>$default</DefaultNamespaceNodeName>
        <NamespaceSeparator>:</NamespaceSeparator>
        <TextNodeName>#text</TextNodeName>
        <AttributeBlockName>#attrs</AttributeBlockName>
        <AttributePrefix>@</AttributePrefix>
        <InvalidCharsReplacement>_</InvalidCharsReplacement>
        <ObjectRootElementName>root</ObjectRootElementName>
        <ArrayRootElementName>array</ArrayRootElementName>
        <ArrayItemElementName>item</ArrayItemElementName>
    </Options>
    <OutputVariable>response</OutputVariable>
    <Source>response</Source>
</JSONToXML>
```

Returning SOAP as a RESTful Response

XSL for cleaning up the SOAP payload or XPath via Extract Variables policy?

- Use XPath if you just need a couple of fields.
- XSLT (XSL) if you need to extract some value or entity with a dynamic or iterative value
- XSLT is generally used for complex XML payload mediations. You can also convert to JSON and then use JSONPath or manipulate via JavaScript

apigee

Tips and Tricks for Mediation Policies

Inconsistent conversion of arrays and string fields containing numbers can be incorrectly converted by XMLToJSON and cause issues for app programmers

Can fix this with JavaScript using global search and replace functions:

Step 1: Set RecognizeBoolean, RecognizeNumber, RecognizeNull to true

Step 2: Add ~STR~ to fields that should be strings but may be converted to non-strings ("~STR~" prefix forces elements to look like strings)

Step 3: Add 2 ~ARRAY~ dummy array elements to each array (there will be a minimum of 2 elements per array, forcing conversion to JSON array instead of null or object)

Step 4: If top-level JSON should be an array, use TOPARRAY as the XML root element

Step 5: After XMLToJSON, clean up results with following JavaScript code:

Tips and Tricks for Mediation Policies

```
// 1) replaces empty strings with null
// 2) removes remaining string specifiers
// 3) removes array placeholders, including trailing commas
// 4) removes array placeholders without trailing commas
// 5) if the return payload is meant to be an array (at the top level), return that
var jsonResponse = context.getVariable("jsonResponse").replace(/"~STR~"/,"null")
                .replace(/~STR~/g,"").replace(/"~ARRAY~",/g,"").replace(/"~ARRAY~"/g,"");
var jsonVar = JSON.parse(jsonResponse);
var jsonOut = jsonVar.ROOT;
if (jsonOut.TOPARRAY !== undefined) {
  jsonOut = jsonOut.TOPARRAY;
}
response.content = JSON.stringify(jsonOut);
```

apigee

Tips and Tricks for Mediation Policies

```
// 1) replaces empty strings with null
// 2) removes remaining string specifiers
// 3) removes array placeholders, including trailing commas
// 4) removes array placeholders without trailing commas
// 5) if the return payload is meant to be an array (at the top level), return that
var jsonResponse = context.getVariable("jsonResponse").replace(/"~STR~"/,"null")
                                    .replace(/~STR~/g,"").replace(/"~ARRAY~",/g,"").replace(/"~ARRAY~"/g,"");
var jsonVar = JSON.parse(jsonResponse);
var jsonOut = jsonVar.ROOT;
if (jsonOut.TOPARRAY !== undefined) {
  jsonOut = jsonOut.TOPARRAY;
}
response.content = JSON.stringify(jsonOut);
```

apigee

XSLT

Google Cloud

XSL Transform Policy

- This policy applies custom XSL transformations to XML messages, letting you transform them from XML to another format, such as XML, HTML, or plain text
- This policy will transform XML only if the Content-Type header on the message to be transformed is set to XML. For example, text/xml or application/xml
- Edge relies on the Saxon XSLT processor, and supports XSLT 1.0 and 2.0
- The XSL policy requires two inputs:
  - The name of an XSLT stylesheet, which contains a set of transformation rules) stored in the API proxy under /resources/xsl directory
  - The source of the XML to be transformed (typically a request or response message)

XSL Transform Policy

```
<?xml version="1.0" encoding="UTF-8"?>
<catalog>
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
    <country>USA</country>
    <company>Columbia</company>
    <price>10.90</price>
    <year>1985</year>
  </cd>
  <cd>
      <title>Hide your heart</title>
      <artist>Bonnie Tyler</artist>
      <country>UK</country>
      <company>CBS Records</company>
      <price>9.90</price>
      <year>1988</year>
   </cd>
 </catalog>
```

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet version="1.0"
xmlns:xsl="http://www.w3.org/1999/XSL/Transform">
<xsl:template match="/">
  <collections title="My CD Collection">
      <xsl:for-each select="catalog/cd">
      <cd>
        <title><xsl:value-of select="title"
/></title>
        <artist><xsl:value-of select="artist"
/></artist>
      </cd>
      </xsl:for-each>
 </collections>
</xsl:template>
</xsl:stylesheet>
```

Input

XSL Template

apigee

XSL Transform Policy

```
<?xml version="1.0" encoding="UTF-8"?>
<collections title="My CD Collection">
  <cd>
    <title>Empire Burlesque</title>
    <artist>Bob Dylan</artist>
  </cd>
  <cd>
    <title>Hide your heart</title>
    <artist>Bonnie Tyler</artist>
  </cd>
</collections>
```

Output

SOAP Message Validation

Google Cloud

SOAP Message Validation Policy

- This policy enables you to configure API proxies to:
  - Validate any XML message against an XSD schema.
  - Validate SOAP messages against a WSDL definition.
  - Confirm JSON or XML is well-formed, based on content type (if <ResourceURL> element is omitted)

- Benefits :
  - Immediately notifies app developers if their requests are non-conformant or incomplete.
  - Provides information to help developers with proper XML syntax
  - Blocks XML messages with structures that might cause unpredictable behavior, protecting backend services
  - Minimizes time spent troubleshooting
  - Encourages developers to familiarize themselves with the XML schema

apigee

Thank You

Google Cloud