Ingesting SOAP Action from SOAP Payload

Toy SOAP Processing

```
<SOAP-ENV:Envelope xmlns:SOAP-ENV = "http://schemas.xmlsoap.org/soap/envelope/"
SOAP-ENV:encodingStyle = "http://schemas.xmlsoap.org/soap/encoding/">
  <SOAP-ENV:Body>
    <m:MyMessage xmlns:m = "http://electrocommerce.org/abc">
      <m:MyArgument>Hello</m:MyArgument>
    </m:MyMessage>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

SOAP 1.1 example payload

```
<SOAP-ENV:Envelope>
  <SOAP-ENV:Body>
    <m:MyMessage>
      <m:MyArgument>Hello</m:MyArgument>
    </m:MyMessage>
  </SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

```
{
  "SOAP-ENV:Envelope": {
     "SOAP-ENV:Body": {
        "m:MyMessage": [Object
      }
  }
}
```

Toy SOAP Characteristics

- There is no Element attributes
- No Empty Elements
- No Comments or XML Declarations

Real World SOAP Payload

```
<wsdl:definitions xmlns:soap = "http://schemas.xmlsoap.org/wsdl/soap/"
   xmlns:wsdl = "http://schemas.xmlsoap.org/wsdl/" name = "EmployeeService"
targetNamespace = "http://www.jpworks.com/employee">

  <wsdl:portType name = "EmployeeServicePortType">
    <wsdl:operation name = "GetEmployeeById">
      <wsdl:input message = "EmployeeByIdRequest"/>
      <wsdl:output message = "EmployeeResponse"/>
    </wsdl:operation>
  </wsdl:portType>
  <wsdl:binding name = "EmployeeServiceSOAP" type = "EmployeeServicePortType">
    <wsdl:operation name = "GetEmployeeById">
      <soap:operation soapAction = "http://www.jpworks.com/employee/GetEmployeeById"/>
      <soap:body use = "literal"/>
    </wsdl:operation>
  </wsdl:binding>
  <wsdl:service name = "EmployeeService">
    <wsdl:port name = "EmployeeServiceSOAP" binding = "EmployeeServiceSOAP"/>
  </wsdl:service>
</wsdl:definitions>
```

heavily truncated to fit in one slide..

Real World SOAP Payload Inspection Result


```
{
”wsdl:definitions”: {
    ”wsdl:portType”: { ”wsdl:operation”: [Object] },
    ”wsdl:binding”: { ”wsdl:operation”: [Object] },
    ”wsdl:service”: { ”wsdl:port”: ”” }
  }
}
```

Javascript Painless Conversion

XML to JSON

- String Manipulation
- Regex Manipulation *
- Recursive calls *
- Maps and Lists

  XML Element with children XML Element(s) is elementary
  XML Element with Attribute(s) and children XML Element(s) is not obvious in JSON

Painless Script Development

```
POST /apm-7.17.7-transaction-000001/_doc/t9WKMYUB7ikC1HY84JYH/_update
{
  "script": {
     "lang": "painless",
     "source": """
      String unescape(def data){  return / /.matcher(clear(
        /\n|\t/.matcher(data).replaceAll(' '))).replaceAll('');}

      String clear(def data){
       return /[\w:-]+ = "[^"]+"/.matcher(data).replaceAll(''); }

      String data  = '<soap:x xmlns:soap="http://schemas.xmlsoap.org">';
      ctx._source.transaction.name = unescape(data);
     """
  }
}
```

```
GET /apm-7.17.7-transaction-000001/_doc/t9WKMYUB7ikC1HY84JYH?
_source=transaction
```

Painless Script Development

```
HashMap parseXML(def source) {
  def result = new HashMap();
  def data = unescapeString(source);
  while (data =~/<[^\/][^>]*>/){
    String openTag = findTag(data);
    String tag = openTag.substring(1, openTag.length() - 1);
    def value = getElementContent(data, openTag);
    def tmp  = '';
    if (value =~/ <[^\/][^>]*>/) {
      tmp = parseXML(value);
    } else {
      tmp = value;
    }
      result[tag] = tmp;
    }
      return result
    }
```

•

Painless Script Runtime Error

```
{
    "type": "painless_error",
    "reason": "The maximum number of statements
               that can be executed in a loop
               has been reached."
  }
```

Painless Script Runtime Error

```
{
    "type" : "circuit_breaking_exception",
    "reason" : "[scripting] Regular expression
                considered too many characters...,
                this limit can be changed
                by changed by the
                [script.painless.regex.limit-factor] setting"
  }
```

Pull XML Parser

element	match
`<?xml version="1.0"?>`	no match.
`<soap:Envelope xmlns:soap="http://www.w3.org/2003/05/soap-envelope/" soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">`	ignoring soap namespace
`<soap:Body>`	no match.

`<m:GetPrice xmlns:m="https://www.w3schools.com/prices">`	match:
	`namespacePrefix: "m"`
	`tagName: "GetPrice"`
	`namespaceName: "m"`
	`namespaceUri: "https://www.w3schools.com/prices"`

`<m:Item>Apples`	no match.
`</m:Item>`	no match.
`</m:GetPrice>`	no match.
`</soap:Body>`	no match.
`</soap:Envelope>`	no match.

Painless Script Development

```
"script": {
  "lang": "painless",
  "source": """
    String parseSOAPXMLDocument(def data) {
      def result = "";
      // handle whitespace
      String[] elelentList =
      (/\n|\t|\r/.matcher(data).replaceAll(' ')).splitOnToken('<');
       // paginate to one XML element per page
      for (def element : elelentList) {
        // extract tag name from relevant elements
        def tmp = parseSOAPXMLElement('<' + element);
         if (tmp != "") {
          result = tmp;
        }
      }
      return result;
    }
```

Painless Script Development

```
String parseSOAPXMLElement(def data) {
   def result = "";
   def elementMatcher =
     /<([\w]+):([\w]+)\s*(xmlns:)([\w]+)\s*=\s*"([^"]+)"\s*(.*)>/;
   if (data =~ elementMatcher){
      def namespace = elementMatcher.matcher(data).replaceAll('$1');
      def tmp = elementMatcher.matcher(data).replaceAll('$2');
       // look for application-specific elements -
       // not a part of the SOAP namespace
      if (!(namespace =~ /soap/)){
         result = tmp;
      }
   }
   return trim(result);
 }

 String trim(def data){
   return / /.matcher(/\n|\t|\r/.matcher(data)
             .replaceAll(' ')).replaceAll('');
 }
```

Painless Script Development

```
  String data = """
  <?xml version="1.0"?>

  <soap:Envelope
  xmlns:soap="http://www.w3.org/2003/05/soap-envelope/"
  soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding">

  <soap:Body>
    <m:UpdatePrice
    xmlns:m="https://www.w3schools.com/prices"
    soap:encodingStyle="http://www.w3.org/2003/05/soap-encoding"
    >
      <m:Item>Apples</m:Item>
    </m:UpdatePrice>
  </soap:Body>

  </soap:Envelope>
"""
 ctx._source.transaction.name = parseSOAPXMLDocument(data);
```

See Also

- Painless Elastic Search Scripting Language
- Shared Java API
- Custom ScriptEngine
- Circuit Breaker Settings
- Ingest Pipelines to Enhance Observability Data (blog)
- SOAP 1.1
- SOAP 1.2
- SOAP specification with examples
- XML pull parser (Android domain)
- One of XML pull parser repositories