What's new with Web Services in CA Gen

Rajesh Mandava and Tracy Hodges

November 13, 2012

agility
made possible™

ca technologies

web services - basics

- Industry Definition

"A Web service is a method of communication between two electronic devices over the World Wide Web." - Wikipedia

  - Definition is too vague to be useful in practice.

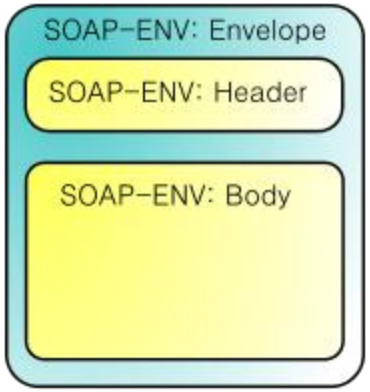
- CA Gen Definition

"A Web service is a cooperative flow performed using SOAP messaging via HTTP over TCP."

  - SOAP is the data exchange protocol
  - HTTP is the application protocol
  - TCP is the transport protocol

web services - terminology

- TCP – Transmission Control Protocol
- HTTP – Hypertext Transfer Protocol
- SOAP – Simple Object Access Protocol
  - XML based message format
  - Envelope, Header, Body
  - WSDL – Web Service Description Language
  - Follows get/response protocol



- Provider – Program that implements the Web Service
  - Maps to CA Gen psteps packaged within cooperative server managers
- Consumer – Program that invokes a Web Service
  - Maps to CA Gen cooperative flow clients

pre CA-Gen 8.5 Web Services support

Web Service support in CA Gen prior to r8.5 was limited to allowing some of the CA Gen servers to expose a Web Service interface to external (non-CA Gen) clients.

- Servers (providers)
  - EJB Web Services (8.0)
  - WebService Wizard (7.6)
  - .NET Proxy using ‘ASP.NET XML Web Services‘ (8.0)
- Clients (consumers)
  - None

CA-Gen 8.5 Web Services support

Expanded Web Services as a native protocol for all clients and added Web Service capabilities to the C Servers.

- Servers (providers)
  - EJB Web Services (8.0)
  - WebService Wizard (7.6)
  - .NET Proxy using ‘ASP.NET XML Web Services‘ (8.0)
  - Transaction Enabler (TE)
- Clients (consumers)
  - C Clients (GUI, COM Proxy and C Proxy)
  - Java Clients (Web Generation, Web View and Java Proxy)
  - .NET Clients (ASP.NET and .NET Proxy)

implementation objectives

Whereas releases prior to 8.5 supported Web Services in an opportunistic way, the 8.5 release attempted to support Web Services in a '1st class', ubiquitous way.

- Full client support
  - C, Java and .NET
- More servers/More platforms
  - TE Servers/Windows, AIX, Itanium, Solaris, Linux
  - Servers interchangeable (fully compatible with each other)
- Minimal configuration
  - Equivalent to TCPIP\CFB support
- Use appropriate WS standards

demo

messages

There are 6 types of request/responses utilized in the web services support within CA Gen.

- Client Request
- Server Response
- WSDL
  - Request
  - Response
- Errors
  - HTTP Errors
  - SOAP:Faults

messages
client request

HTTP POST Header

SOAP Envelope

SOAP Body

View Data

```
POST /ADDDIV/ADD_DIVISION HTTP/1.1
Host: localhost:2008
User-Agent: CA Gen 8.5
SOAPAction: "http://tempuri.org/AddDivision/"
Content-Type: text/xml;charset=UTF-8
Transfer-Encoding: chunked
675
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
                  xmlns:spr="http://tempuri.org/AddDivision/">
  <soapenv:Body>
    <spr:AddDivisioncall>
      <AddDivisionImport command="ADD      "
                         exitState="00000000">
        <ImportDivision>
          <Number>4</Number>
          <Name>ACCOUNTING</Name>
          <ManagerId>1</ManagerId>
        </ImportDivision>
      </AddDivisionImport>
    </spr:AddDivisioncall>
  </soapenv:Body>
</soapenv:Envelope>
```

messages
server response

HTTP Errors result when there is a 'transport' related issue

- HTTP Header will contain:
  - Status Code
  - Status Code Text
- HTTP Body may contain:
  - Error Message Text
- Some possible status codes:
  - 400 (Bad Request)
  - 404 (Not Found)
  - 414 (URL Too Large)
  - 500 (Server Error)

```
HTTP/1.1 404 Not Found
Content-Type: text/xml; charset=utf-8
Content-length: 29

Requested URL was not located
```

SOAP Faults occur when there is an 'application' related issue

- HTTP Header indicates OK status code
- HTTP Body contains the SOAP:Fault
- SOAP:Fault indicates:
  - faultcode
  - faultstring

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-length: 260

<?xml version='1.0' encoding='UTF-8'?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.or
  <soap:Body>
    <soap:Fault>
      <faultcode>soap:Server</faultcode>
      <faultstring>Server Error Occurred (733)</faul
    </soap:Fault>
  </soap:Body>
</soap:Envelope>
```

demo

WSDL – Web Service Definition Language

- XML format
- Describes the location, methods, and data information
- Uses:
  - At design time: generate proxies, create skeleton xml, addressing
  - At runtime: (may or may not be used) validation, dynamic clients
- Very verbose
  - Over-engineered? (REST based web-services)
- More than one way to describe the same web service
- Can be broken up into multiple files
- Does not have to be retrieved from server (may be just a file)

Simple HTTP GET request

- Important information is in the HTTP header
- No HTTP body
- URL will be one of the following:
  - http://***host*:*port*/*LM*/*Pstep***?wsdl
  - http://***host*:*port*/*Pstep*/*Pstep***?wsdl
- Can be sent by entering the URL in a browser window
- Example:

```
GET /P900/ADD_DIVISION?wsdl HTTP/1.1
Host: localhost:4040
Connection: Keep-Alive
```

messages
WSDL response

- HTTP Post
- Standard Sections
  - types/message (data structures)
  - portType/binding (API)
  - Service (address)

```
HTTP/1.1 200 OK
Content-Type: text/xml; charset=utf-8
Content-length: 10730

<?xml version="1.0" encoding="UTF-8"?>
<definitions xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/" x
  <types>
    <xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema" xmln
      <xsd:element name="Exception" type="tns:Exception"/>
      <xsd:element name="AddDivisioncall" type="tns:AddDivisionca
      <xsd:element name="AddDivisioncallResponse" type="tns:AddDi
      <xsd:complexType name="Exception">...</xsd:complexType>
      <xsd:complexType name="AddDivisioncall">...</xsd:complexTyp
      <xsd:complexType name="AddDivisioncall">...</xsd:complexTyp
      <!-- ImportView name=AddDivisionImport -->
      <xsd:complexType name="AddDivisionImpo">...</xsd:complexTyp
      <!-- ExportView name=AddDivisionExport -->
      <xsd:complexType name="AddDivisionExpo">...</xsd:complexTyp
    </xsd:schema>
  </types>
  <message name="AddDivisioncall">...</message>
  <message name="AddDivisioncall">...</message>
  <message name="Exception">...</message>
  <portType name="ADD_DIVISION">...</portType>
  <binding name="ADD_DIVISIONPor" type="tns:ADD_DIVISIO">...</bin
  <service name="ADD_DIVISION">...</service>
</definitions>
```

demo

view data
import and export

The view data for the request and response messages are basically an XML representation of the Pstep's import and export views

- Identical to the 'xml' representation used by the XML API of the C, Java and C# proxies
- Element Naming
  - taken from Gen model
  - 'camel case'
- Structure
  - almost verbatim from Gen model (nesting, sequencing, min/max occurrences)
  - issues with 'optional/mandatory'
- Types
  - closest representation: {string, double, decimal, short, integer, base64Binary}
  - few defined restrictions (compatibility and interpretation issues)

view data
system attributes

Stored as XML attributes on the Import and Export xml elements

- Import:
  - command - string
  - clientId - string
  - clientPassword - string
  - nextLocation - string
  - exitState - integer
  - dialect - string
- Export:
  - command - string
  - exitState - integer
  - exitStateType - string enumeration {“0”, “1”, “2”, “3”}
  - exitStateMsg - string

view data
entity views/work views

- The element name for an entity view is the concatenation of:
  Entity View name & Entity name
- The element name for a work view is simply the Work View name
- The WSDL for each entity/work view becomes:

```
<xsd:element name="viewname">
        <xsd:complexType>
                <xsd:sequence>
                        ....
                </xsd:sequence>
        </xsd:complexType>
 </xsd:element>
```

- Within the sequence element are the modeled attribute views
- The <sequence> forces the ordering of the contained attributes

view data
non-repeating group views

- The element name is simply the Group view name
- The WSDL for each non-repeating group view becomes:

```
<xsd:element name="groupviewname">
        <xsd:complexType>
                <xsd:sequence>
                        ....
                </xsd:sequence>
        </xsd:complexType>
 </xsd:element>
```

- Within the sequence element are the modeled entity, work or group views
- The <sequence> forces the ordering of the contained views

view data
repeating group views

- The element name is simply the Group view name
- The WSDL for each repeating group view becomes:

```
<xsd:element name="groupviewname">
    <xsd:complexType>
        <xsd:sequence>
            <xsd:element name="row" minOccurs="0" maxOccurs="##">
                <xsd:complexType>
                    <xsd:sequence>
                        ....
                    </xsd:sequence>
                </xsd:complexType>
            </xsd:element>
        </xsd:sequence>
    </xsd:complexType>
</xsd:element>
```

- A ‘row’ element is added to represent each occurrence of the repeating group view
- The ‘row’ may be repeated up to maxOccurs times
  (maxOccurs is based on the group view max - as modeled)

view data
attribute views

- The element name is simply the attribute view name
- The WSDL for each attribute view becomes:
  <xsd:element name="***attributename***" type="***type***">
- The type is based on the attribute data type in the model:
  - Date & Time -> xsd:int
  - Timestamp -> xsd:string
  - Blob -> xsd:base64Binary
  - Text, Mixed, & DBCS -> xsd:string
  - Numeric (Decimal Precision) -> xsd:decimal
  - Numeric (with decimal places) -> xsd:double
  - Numeric (digits > 9) -> xsd:double
  - Numeric (9 >= digits > 3) -> xsd:int
  - Numeric (3 >= digits) -> xsd:short
- If attribute is 'optional', then a 'minOccurs="0"' will be added

view data
sample import view

```
<UpdateEmployeeImport command="UPDATE  " clientId="" clientPassword=""
                      nextLocation="" exitState="0" dialect="DEFAULT">
  <ImportEmployee>
    <Number>1</Number>
    <Name>MR SMITH</Name>
    <AddressLine1>5465 LEGACY DRIVE</AddressLine1>
    <AddressLine2>PLANO, TX 75024</AddressLine2>
    <AddressLine3></AddressLine3>
    <AddressLine4></AddressLine4>
    <AddressLine5></AddressLine5>
    <HomePhone>214-473-1280</HomePhone>
    <WorkPhone>972-727-3001</WorkPhone>
    <PayType>S </PayType>
    <PayRate>10000.00</PayRate>
    <MailStop>777</MailStop>
    <ServiceDate>19941107</ServiceDate>
    <FullTimeStatus>Y </FullTimeStatus>
    <DivisionNumber>1</DivisionNumber>
    <DepartmentNumber>11</DepartmentNumber>
  </ImportEmployee>
</UpdateEmployeeImport>
```

view data

sample export with repeating group view

```
<ListEmployeesExport exitStateMsg="" exitStateType="3" exitState="0" command="OK">
  <ListEmployees>
    <row>
      <Employee>
        <Number>1</Number>
        <Name>MR SMITH                          </Name>
        <AddressLine1>5465 LEGACY DRIVE                 </AddressLine1>
        <AddressLine2>PLANO, TX 75024                   </AddressLine2>
        <WorkPhone>972-727-3001    </WorkPhone>
        <PayRate>10000.00</PayRate>
        <ServiceDate>19941107</ServiceDate>
        <DivisionNumber>1</DivisionNumber>
        <DepartmentNumber>11</DepartmentNumber>
      </Employee>
    </row>
    <row>
      <Employee>
        <Number>2</Number>
        <Name>JOHN DOE                          </Name>
        <AddressLine1>1600 PENNSYLVANIA AVE             </AddressLine1>
        <AddressLine2>WASHINGTON, DC                    </AddressLine2>
        <WorkPhone>987-654-3210    </WorkPhone>
        <PayRate>15000.00</PayRate>
        <ServiceDate>20110111</ServiceDate>
        <DivisionNumber>1</DivisionNumber>
        <DepartmentNumber>11</DepartmentNumber>
      </Employee>
    </row>
  </ListEmployees>
</ListEmployeesExport>
```

demo

how to do it
Using Web Services within CA Gen

- Installation
  - New cooperative runtime – 'Web Services Middleware'
- Toolset
  - No Web Service specific changes
- Generation
  - Servers
    - EJB Servers: must be generated with 'EJB WebServices' TP Monitor
    - TE Servers: must be generated with 'IEFAE' TP Monitor
  - Clients
    - No generation impact

how to do it
Using Web Services within CA Gen

- Building/Assembling
  - Builds are unaffected
  - Assembling for Java, .NET and C GUI applications will automatically include the WS cooperative runtimes if they were installed.

    (might want to modify commcfg files prior to assembling)

how to do it
Using Web Services within CA Gen

- Runtime/Deployment
  - Servers
    - EJB Servers: no changes
    - TE WebServices:
      - new runtime DLLs (installed with TE)
      - no new AEFUF/AEFAD command line options (WS is baked in)
  - Clients
    - Runtimes
      - Need DLLs, assemblies, classes present at runtime
      - Should be taken care of by assemble process
    - Commcfg files
      - Modify for WS as needed

commcfg

Currently the only way to perform a Web Service flow is via runtime configuration settings done in a commcfg file.

- Syntax: WS <baseURL> <contextType>
  - <baseURL> is the scheme, domain and port of the web service address
    i.e. “http://myhost:2008/”
  - <contextType> indicates how the remaining part of the address is to be constructed. It will be either: L or P
    - “L” indicates the URL is Load Module based (most servers)
      i.e. “http://hostname:80/LM/PSTEP”
    - “P” indicates the URL is PStep Name based (primarily **WebLogic**)
      i.e. “http://hostname:80/PSTEP/PSTEP”

exits

There are a few user exits for Web Services related to controlling the URL address of the web service.

- C runtime: CIWSCLX.c
  - CI_WS_DPC_Exit(…) – change the baseURL and contextType
  - CI_WS_DPC_URL_Exit(…) – change the final URL
- Java runtime: WSDynamicCoopFlowExit.java
  - getBaseURL() – change the baseURL
  - getContextType() – change the contextType
  - modifyURL(…) – change the final URL
- .NET runtime: WSDynamicCoopFlowExit.cs
  - BaseURL property – change the baseURL
  - ContextType property – change the contextType
  - ModifyURL(…) – change the final URL

demo

Pros and Cons

- Pros
  - Industry Standard Format
  - Interoperability
  - Easy Configuration
  - Small client footprint (TCP/IP only)
  - External client access
  - Can tunnel thru firewalls easily
- Cons
  - Bloated message size due to XML
  - Performance can be slower due to XML parsing
  - Implementation differences

questions

thank you

terms of this presentation
for information purposes only

This presentation was based on current information and resource allocations as of November 2012 and is subject to change or withdrawal by CA at any time without notice. Notwithstanding anything in this presentation to the contrary, this presentation shall not serve to (i) affect the rights and/or obligations of CA or its licensees under any existing or future written license agreement or services agreement relating to any CA software product; or (ii) amend any product documentation or specifications for any CA software product. The development, release and timing of any features or functionality described in this presentation remain at CA's sole discretion. Notwithstanding anything in this presentation to the contrary, upon the general availability of any future CA product release referenced in this presentation, CA will make such release available (i) for sale to new licensees of such product; and (ii) to existing licensees of such product on a when and if-available basis as part of CA maintenance and support, and in the form of a regularly scheduled major product release. Such releases may be made available to current licensees of such product who are current subscribers to CA maintenance and support on a when and if-available basis. In the event of a conflict between the terms of this paragraph and any other information contained in this presentation, the terms of this paragraph shall govern.

Certain information in this presentation may outline CA's general product direction. All information in this presentation is for your informational purposes only and may not be incorporated into any contract. CA assumes no responsibility for the accuracy or completeness of the information. To the extent permitted by applicable law, CA provides this presentation "as is" without warranty of any kind, including without limitation, any implied warranties or merchantability, fitness for a particular purpose, or non-infringement. In no event will CA be liable for any loss or damage, direct or indirect, from the use of this document, including, without limitation, lost profits, lost investment, business interruption, goodwill, or lost data, even if CA is expressly advised in advance of the possibility of such damages. CA confidential and proprietary. No unauthorized copying or distribution permitted.