OpenVMS and web services: theory and practise

Jeff Allen and Brett Cameron
September 2014

Abstract

Web services are now commonplace in enterprise computing environments; however in recent years there has been something of a move away from traditional and standards-based SOAP-style web services to increased adoption of RESTful web services. This talk will introduce web services and the role they play in the enterprise, and will consider the RESTful and SOAP-based styles in terms of both ideological and technical merit. The speakers will discuss examples of how OpenVMS customers are using web services as a means of integration and modernization, and a case study will be presented that illustrates how Quest Diagnostics has used the gSOAP toolkit on OpenVMS to facilitate both SOAP-based and RESTful web service integration between GP-facing and laboratory systems and their core backend OpenVMS-based environment.

About us - Jeff

Jeff Allen has been serving as a software designer in the medical industry for over 20 years focusing primarily in data collection from both interactive users and remote systems. As technologies evolve over the years (expert systems, VPN, web services, etc...), they have been employed and implemented where needed and, when, hardware constraints and budgetary concerns pop up, Jeff can blend old and new technology for innovative solutions. Solutions are never impossible.

About us - Brett

Brett Cameron currently works as a senior architect with HP's corporate Cloud Services group, focusing on the design and implementation of message queuing and related integration services for customers and for internal use. Brett lives in Christchurch, New Zealand, and has worked in the software industry for some 22 years. In that time he has gained experience in a wide range of technologies, many of which have long since been retired to the software scrapheap of dubious ideas. In recent years Brett has specialized in systems integration, and the design and implementation of large distributed systems for HP's enterprise customers. This work has seen Brett get involved in the research and development of low-latency and highly scalable messaging solutions for the Financial Services sector running on HP platforms, and as a consequence of this work, Brett has been involved in several interesting Open Source projects, and he has been responsible (or should that be irresponsible) for porting various pieces of Open Source software to the HP OpenVMS platform. Brett holds a doctorate in chemical physics from the University of Canterbury, and still maintains close links with the University, working as a part time lecturer in the Computer Science and Electronic and Computer Engineering departments. In his spare time, Brett enjoys listening to music, playing the guitar, and drinking beer.

AGENDA

- **A quick look at web services (SOAP-based and RESTful)**
- gSOAP overview
- Some things our customers are doing
- Case study
- Summary/conclusion
- Questions

Early internet/intranet technologies

Early internet technologies:

- SMTP
- FTP
- NNTP
- HTTP/HTTPS, HTML

Most of these facilitated application to human interaction over the internet or intranet

Early intranet technologies:

- DCE from OSF
  - RPC based, procedural
- ORB
  - Object oriented, mostly synchronous
  - CORBA, COM/DCOM, Java RMI/EJB's
- MOM
  - Message oriented
  - Synchronous and asynchronous
  - JMS
  - Many proprietary implementations

Most of these facilitated application to application interaction within a trusted intranet and without much consideration to interoperability across different implementations.

Web services: an essential part of SOA

- What is a web service? (www.W3C.org)
  - A web service is a software system designed to support interoperable machine-to-machine interaction over a network
    - It has an interface described in a machine-processable format (specifically WSDL)
    - Other systems interact with the web service in a manner prescribed by its description using SOAP-messages, typically conveyed using HTTP with an XML serialization in conjunction with other web-related standards
    - Interoperability and connectivity defined by www.WS-I.org

Web services are to computers what browsers are to users…

A modular piece of code on the internet/intranet/extranet that provides one or more business functions, and that can be discovered and used on demand.

7

Web services – another take…

Software components designed to provide specific operations (*services*) accessible using standard internet technology

- Language and platform independent
  - Separation of specification and implementation
- Usually provided through SOAP
  - Messages carrying XML documents, and a HTTP transport protocol (more on SOAP later…)
- Loosely coupled
  - Message based, synchronous and asynchronous interactions
- Over the Internet/intranet
  - No centralized control, use of established protocols, security considerations
- Inter-operable
  - Standards based

Language and platform independent infrastructure for loosely-coupled, inter-operable, application to application communication over the internet.

8

SOA and web services

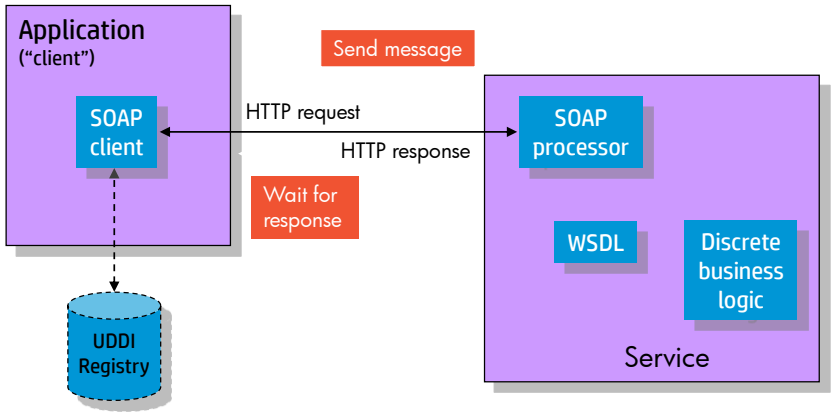
One of the key reasons for focus upon SOA over the past decade has been the emergence of supporting technologies

- SOA is an *architectural* approach, centered around the concept of *services*
- A common source of confusion:
  - SOA ≠ web services
  - SOA can exist without web services
  - Web services can be utilized without an SOA
  - Using web services can significantly enhance our ability to implement SOA
- Why web services for SOA?
  - Web services are (mostly) standardized
  - Web services promote loose-coupling
  - They are platform independent and vendor independent
  - They provide integration at the service level
  - Web services include service description mechanisms
  - Web services include service catalogue mechanisms

Address of a web service

- Uses URI's
  - URI's == Uniform Resource Identifier's and are the standard web naming/addressing mechanism
- URL's are a subset of URI's, and would typically be used to address a web service
  - For example: http://services.xmethods.net/soap/getQuote
- URI's also encompass:
  - Email addresses (mailto:brett.cameron@hp.com)
  - Uniform Resource Names (URN's), which are globally unique and persistent (used by UDDI)

Simple web service interaction



1. Client code calls client stub
2. SOAP request sent across network
3. Server stub receives request and passes request to service code
4. Service passes result(s) to server stub
5. Server stub sends result(s) across network to client stub
6. Client stub passes result(s) to client code

More about SOAP and WSDL later...

Base web services standards

Who comes up with this stuff?

- World Wide Web Consortium (W3C)
  http://www.w3c.org
- Organization for the Advancement of Structured Information Standards (OASIS)
  http://www.oasis-open.org
- WS-Interoperability (WS-I)
  http://www.ws-i.org

Base web service standards

SOAP

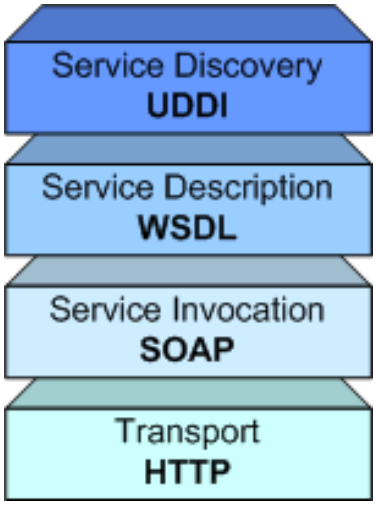
- Standard packaging structure for XML document transport
- Transport over SMTP, HTTP, FTP, …
- Also defines encoding and binding standards for non-XML RPC invocations
- EJBs and .NET components can be exposed through SOAP

WSDL (Web Services Description Language)

- XML technology that describes the interface to a Web service in a standardized way
- Describes I/O parameters of an invocation externally
- Describes service protocol binding

UDDI (Universal Description, Discovery, and Integration)

- Registry of available web services
- Used for advertising, discovering, and integrating web services



13

Base web service standards – WSDL

WSDL is:

- An interface description language (IDL)
  - Provides a way of formally describing a service, what is does, how it is accessed, and so on
- A W3C standard XML document that describes three fundamental properties of a service
  1. What it is
     - The operations (methods) it provides
  2. How it is accessed
     - Data format
     - Protocols
  3. Where it is located
     - Protocol-specific network address

Components of WSDL include:

- Root definitions
  - Namespaces
- Port-type definitions
  - Abstract definition of service(s)
- Message definitions
  - Parameters in method signature
- Type definitions
  - Data type definitions (structures)
- Binding definitions
  - Binding to protocols
  - Typically SOAP over HTTP
- Service definitions
  - Where service resides
  - Ports

14

Base web service standards – SOAP

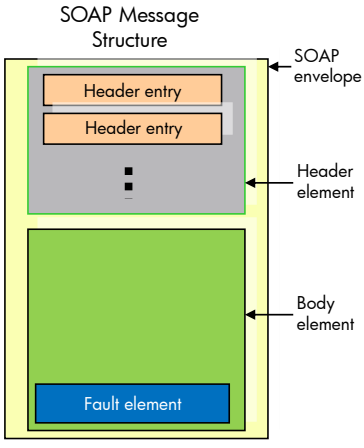
- The cornerstone of interoperability
- Self-describing messages based on proven conventions
- Partners can communicate as long as they:
  - Can send and receive transmissions across a network using HTTP, SMTP, or other transport mechanism
  - Understand MIME encoding rules
  - Can construct and deconstruct XML documents that conform to the SOAP rules
  - Can perform the required action, if one is specified in the SOAP document (for example, invoke a web service method)
- It is possible to create SOAP bindings for almost any protocol
  - The de-facto binding is currently HTTP or SMTP, allowing SOAP to operate through most firewalls
  - Common use of other protocols has also emerged:
    - SOAP over RMI
    - SOAP over JMS (for improved reliability)
    - …

"SOAP" once stood for Simple Object Access Protocol but this acronym was dropped with Version 1.2 of the standard, as it was considered to be misleading.

See http://en.wikipedia.org/wiki/SOAP

SOAP in one slide

- Light-weight protocol based on XML as the marshalling format for data in request and response messages
  - Encoding rules for data type instances
  - Vendor and platform-neutral
  - Language-neutral
  - Object model neutral
  - Transport neutral
- Designed for loosely-coupled distributed computing
  - No remote references
- XML allows data transformation (XSLT)
- XML enables long-term data persistence



UDDI, SOAP, and WSDL are just the beginning

What other services are required by enterprise applications?

- Security
- Context and transaction management
- Business processes
- Reliable messaging
- Policies
- Notification
- Management

... these are the second generation web services, and *many* specifications define these services!

REST

- *Representational state transfer* (REST)
  - A fancy (and somewhat miss-used) term for a fairly simple concept
  - Introduced and formalized by Roy Fielding (2000) in his doctoral dissertation ("*Architectural Styles and the Design of Network-based Software Architectures*", http://www.ics.uci.edu/~fielding/pubs/dissertation/top.htm)
- An architectural style that is most often applied to the development of web services
  - The World Wide Web effectively conforms to the REST architectural style

The concept:

Representational State Transfer is intended to evoke an image of how a well-designed Web application behaves: presented with a network of Web pages (a virtual state-machine), the user progresses through an application by selecting links (state transitions), resulting in the next page (representing the next state of the application) being transferred to the user and rendered for their use.

REST was initially described in the context of HTTP, but it is not limited to that protocol. RESTful architectures may be based on other Application Layer protocols if they already provide a rich and uniform vocabulary for applications based on the transfer of meaningful representational state. RESTful applications maximize the use of the existing, well-defined interface and other built-in capabilities provided by the chosen network protocol, and minimize the addition of new application-specific features on top of it.

http://en.wikipedia.org/wiki/Representational_state_transfer

RESTful web services

- Well-defined HTTP-based RESTful API's adhere to the following characteristics:
  - A base URI/URL
  - An Internet media type (typically JSON or XML) for request and response data
  - Use of standard HTTP methods for certain types of operation (see table below)
  - Use of hypertext links to reference state
  - Use of hypertext links to reference related resources

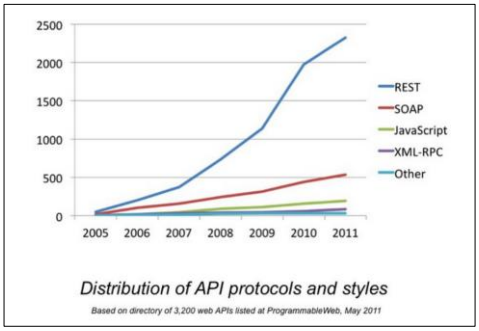
Resource	GET	PUT	POST	DELETE
Collection URI, such as http://example.com/resources	List the URIs and perhaps other details of the collection's members.	Replace the entire collection with another collection.	Create a new entry in the collection. The new entry's URI is assigned automatically and is usually returned by the operation.	Delete the entire collection.
Element URI, such as http://example.com/resources/item17	Retrieve a representation of the addressed member of the collection, expressed in an appropriate Internet media type.	Replace the addressed member of the collection, or if it doesn't exist, create it.	Not generally used. Treat the addressed member as a collection in its own right and create a new entry in it.	Delete the addressed member of the collection.

http://en.wikipedia.org/wiki/representational_state_transfer

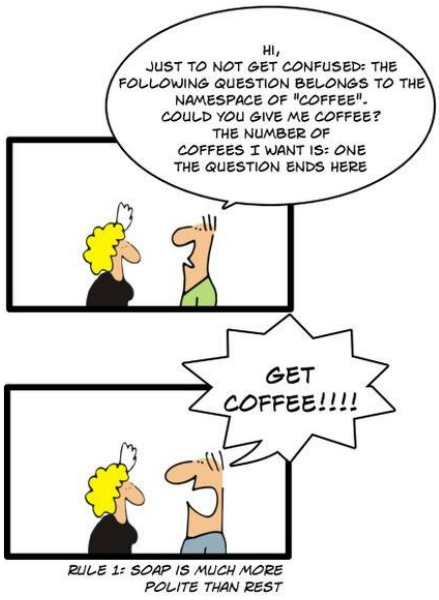
RESTful web services

- Unlike SOAP-based web services, there is no official standard for RESTful web services
  - REST is an architectural style
  - SOAP is a defined standard

- Wide-spread adoption
  - RESTful interfaces are far more common than SOAP-based services
  - Most Cloud-based services provide some form of RESTful interface
  - Many software products provide RESTful interfaces for various purposes
    - See for example the RabbitMQ Management API



Comparing SOAP-based web services with SOAP-based services…



21

Implementing RESTful interfaces on OpenVMS

- There are numerous frameworks in various languages
  - Jetty (Java; see http://www.eclipse.org/jetty/)
  - Ruby Sinatra (http://www.sinatrarb.com/)
  - Axiom (Erlang; see https://github.com/tsujigiri/axiom)
  - …
  - But these frameworks cannot necessarily be easily used on OpenVMS or to hook into existing 3GL applications
- An embedded web server like Mongoose works well on OpenVMS for the development of OpenVMS-based RESTful web services
  - See https://code.google.com/p/mongoose/
- gSOAP includes a comprehensive and efficient HTTP library
  - Client and server
    - Client API not as comprehensive as libcURL (but libcURL is client-only)
  - Simple to use
  - Fast
  - Currently provides limited server-side support for RESTful services via a plugin
    - Would not take much to modify the plugin to support any type of RESTful operation (any HTTP verb)

Mongoose provides a nice framework for mapping URI's to code functions. See here for a trivial example.

22

Implementing RESTful interfaces on OpenVMS

- libcURL provides very comprehensive HTTP(s) client support and is arguably the best library to use for implementing interfaces to RESTful services from existing OpenVMS 3GL code
  - I have used this to develop interfaces to OpenStack-based cloud services
- Note that most RESTful web services return either JSON or XML responses
  - Sometimes both formats are supported and you can specify which format you want
  - Need either an XML or a JSON parser/API
    - I prefer JSON - it's generally more efficient
    - json-c is easily ported to OpenVMS (see https://github.com/json-c/json-c)

AGENDA

- A quick look at web services (SOAP-based and RESTful)
- **gSOAP overview**
- Some things our customers are doing
- Case study
- Summary/conclusion
- Questions

What is gSOAP?

- Full-featured Open Source SOAP technology/toolkit
  - See http://www.cs.fsu.edu/~engelen/soap.html
- 3000+ registered users
- Uses a source-to-source stub and skeleton compiler to automate the integration of SOAP RPC in applications
  - Client and server development
  - Automates the deployment of (legacy) 3GL applications as Web services
    - Primarily intended for use with C/C++, but can be used (with a little effort) with any 3GL
  - Automates the development of clients
- Suitable for high-performance solutions (very fast)
  - 2265 round-trip calls per second (single thread) on an old 2-CPU 1.4GHz Itanium2 system running Linux 2.6.9 IA64
    - Measured with 2.2KB XML messages over HTTP
- Major components ported to OpenVMS (Alpha and IA64)
  - Extensions to simplify use from languages other than C/C++
  - ACMS support (threaded agent gateway, tools to assist with interface creation)
  - Integration with Apache via `mod_gsoap`

gSOAP goals

Application-centric

- Minimize legacy application code adaptation
- Support (de)marshalling of application's native data structures in SOAP/XML
- Preserve the logical structure of data

Minimize data migration overhead and formatting errors

- Avoid (hand-written) wrappers
- Generate fast (de)marshalling routines and streaming XML parsers
- Efficient run-time remote object allocation

gSOAP tools

Stub/skeleton compiler (`soapcpp2`)

- Generates source code stubs and skeletons for SOAP RPC
- Generates XML (de)marshalling routines for native and user-defined C/C++ data types

WSDL/schema parser (`wsdl2h`)

- Imports one or more WSDL files and XSDs to generate a C/C++ header file that defines the service prototypes and data types
  - The C/C++ header file would then be used as input to `soapcpp2`

gSOAP runtime

- Provides low-level HTTP, TCP, SOAP/XML handling, and memory management capabilities

gSOAP development

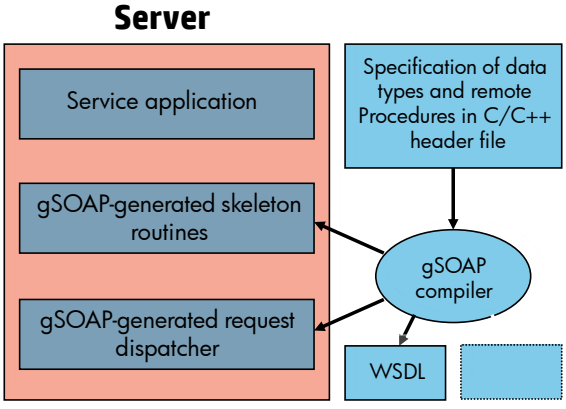
Two basic approaches to development…

Note that this and the subsequent two slides pertain to development of SOAP-based web services using gSOAP. Development of RESTful interfaces with gSOAP is considerably different and not covered here, but will be discussed later during the Case Study.

1. Start with a WSDL (*top down*)
   - Approach typically used when wanting to call an existing Web service
   - Use `wsdl2h` to convert WSDL to C/C++ header file
   - Use `soapcpp2` to generate stubs and skeletons
   - Develop client application
   - Link client application with generated code and gSOAP runtime

2. Start *without* a WSDL (*bottom up*)
   - Approach might typically be used to expose existing (legacy) functionality as Web services
   - Create a C/C++ header file containing the necessary data type and service (function prototype) definitions
   - Use `soapcpp2` to generate stubs and skeletons
   - Link generated code and gSOAP runtime with existing (or new) application code

gSOAP development

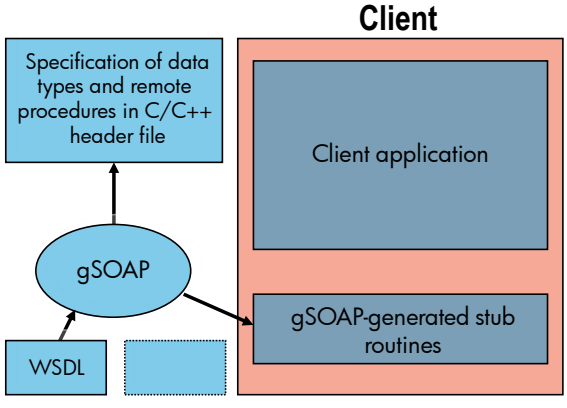


Adapted from http://www.cs.fsu.edu/~engelen/soaptalk.ppt

Supplied header file is processed by `soapcpp2` to generate stub and skeleton routines that are linked with user-written application code. Note that `soapcpp2` can optionally generate WSDL.

gSOAP development

An existing WSDL can be used to develop a gSOAP client (or server) application. The `wsdl2h` tool is used to convert the WSDL into a header file, which can be processed by `soapcpp2` to generate stub and skeleton routines that are linked with user-written application code.



Adapted from http://www.cs.fsu.edu/~engelen/soaptalk.ppt

OpenVMS port

Major components ported to OpenVMS Alpha and IA64

- `soapcpp2.exe`
  - Stub and skeleton compiler
  - Generates proxies (and RPC stubs)
  - Generates the C/C++ Web service skeletons
  - Can optionally generate WSDL and XSDs
- `wsdl2h.exe`
  - WSDL parser
  - Converts WSDL into gSOAP header file specifications of Web services
- Object libraries (runtime libraries)
  - Provide a transport layer with an HTTP stack on top of TCP/IP
  - Optional support for SSL and DIME/MIME attachment
- `mod_gsoap`
  - Apache module for gSOAP

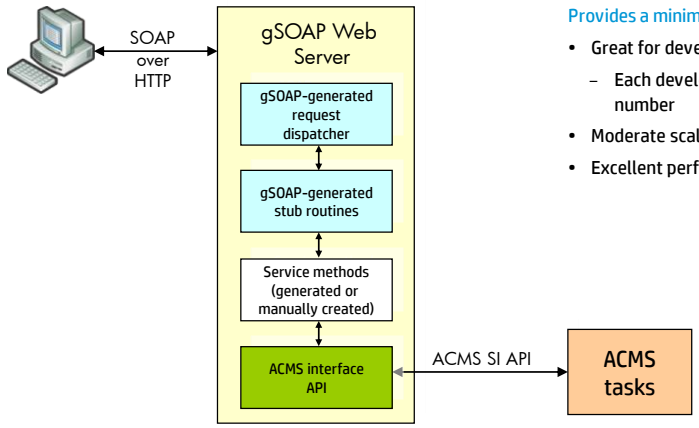
Wrapping ACMS applications

Developing a gSOAP-based Web Services front-end for an ACMS application

- An interface can be automatically generated from STDL
- Can deploy as a standalone gSOAP server or run under Apache using `mod_gsoap`
- At most all you need to do is write a simple “main” routine
- Some more exotic ACMS/COBOL data types currently not supported (but can easily be added)

Wrapping ACMS applications (standalone)

Standalone gSOAP server

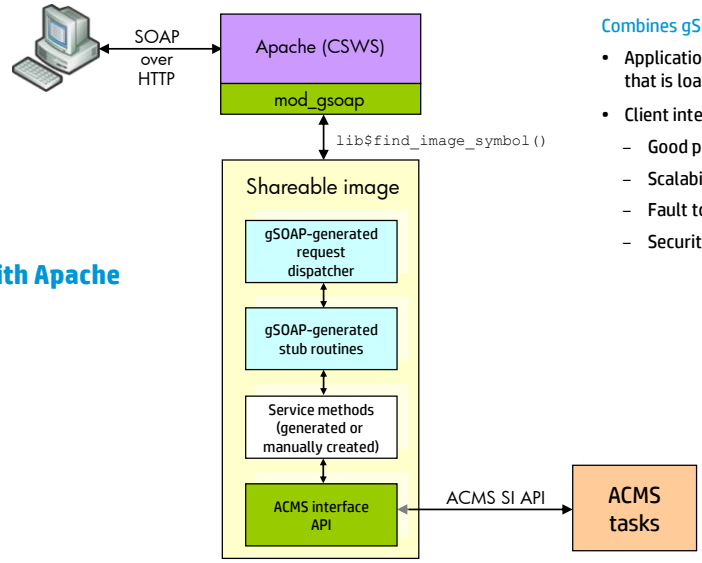


Provides a minimal Web server implementation

- Great for development and unit testing
  - Each developer can use their own port number
- Moderate scalability
- Excellent performance

Wrapping ACMS applications (Apache)

gSOAP with Apache



Combines gSOAP and Apache

- Application linked as a shareable image that is loaded by `mod_gsoap`
- Client interaction managed by Apache
  - Good performance
  - Scalability
  - Fault tolerance
  - Security

AGENDA

- A quick look at web services (SOAP-based and RESTful)
- gSOAP overview
- **Some things our customers are doing**
- Case study
- Summary/conclusion
- Questions

What are people doing with gSOAP?

There are today probably in excess of two dozen organizations using gSOAP on OpenVMS (Alpha and Integrity) in production environments

- Spanning multiple sectors
  - Finance
  - Healthcare
  - Telecommunications
  - Tourism
  - Manufacturing
  - Entertainment
  - ...
- Involving integration with OpenVMS applications written in various languages and using various technologies
  - C, C++, COBOL, FORTRAN, BASIC, Pascal
  - ACMS, DEC MessageQ, ...
- Fulfilling business-critical functions

What are people doing with gSOAP?

Some examples:

- A large medical insurance company uses gSOAP with Apache to provide a web services interface to their "legacy" ACMS environment
- A large healthcare provider uses gSOAP on OpenVMS to integrate a .NET application used by doctors across the US with their back-end systems
  - More on this later!
- A steel manufacturer uses gSOAP to communicate between their OpenVMS based systems and factory cranes fitted with on-board computers running a .NET application
- A large European travel company uses gSOAP and Apache to provide a web services interface to their ACMS application and to call web services from COBOL code within their applications
- … and many more

Solutions range from the simple to the complex; however in all cases gSOAP has worked well

- Good performance and stability
- Flexibility
  - Excellent interoperability with other web services technologies
- Relative ease of integration with "legacy" code

AGENDA

- A quick look at web services (SOAP-based and RESTful)
- gSOAP overview
- Some things our customers are doing
- **Case study**
- Summary/conclusion
- Questions

Link to Jeff's slides!

AGENDA

- A quick look at web services (SOAP-based and RESTful)
- gSOAP overview
- Some things our customers are doing
- Case study
- **Summary/conclusion**
- Questions

Concluding remarks

- Building web services on OpenVMS and calling web services from OpenVMS-based applications is readily possible and many OpenVMS users have successfully implemented web service-based interfaces between their OpenVMS applications and other systems

- Useful tools include:
  - gSOAP and libcURL can be readily incorporated into existing 3GL applications
  - gSOAP provides support for SOAP-based services and limited support for RESTful service
  - libcURL can be used to call RESTful services
  - WSIT can be used (with Apache AXIS2) to create SOAP-based web services on OpenVMS (Java-based)
  - An XML or JSON parser/API may be required when working with RESTful services

- REST-based services have become more polular than SOAP-based services
  - RESTful services are in some ways easier to implement and use, but REST is not a standard and RESTful services tend to be less strictly defined
  - The RESTful approach will continue to dominate

And some concluding remarks from Jeff…

AGENDA

- A quick look at web services (SOAP-based and RESTful)
- gSOAP overview
- Some things our customers are doing
- Case study
- Summary/conclusion
- **Questions**


Thank you