Volume 4, Issue 1, January 2014 ISSN: 2277 128X
International Journal of Advanced Research in Computer Science and Software Engineering
Research Paper
Available online at: www.ijarcsse.com

Supported Formats in Web Services Incorporation with Mobile Applications

B.Meena*, I.S.L.Sarwani
ANITS
India

Abstract :Web services are playing vital role in every small internet and web technologies. Web services in simple word is an end-user way to communicate through electronic devices over the web. Web services include wide range of services like website designing, web development, coding, programming, software functionality and other web application technologies.XML Web Services enable any form of distributed processing to be performed using a set of standard Web- and XML-based protocols and technologies. SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment. It is an XML based protocol that consists of three parts: an envelope that defines a framework for describing what is in a message and how to process it, a set of encoding rules for expressing instances of application-defined data types, and a convention for representing remote procedure calls and responses REST has been applied to describe desired web architecture, to identify existing problems, to compare alternative solutions, and to ensure that protocol extensions would not violate the core constraints that make the Web successful

Keywords : Web services, XML ,SOAP,REST

I . INTRODUCTION

A Web service is a method of communications between two electronic devices over the World Wide Web. It is a software function provided at a network address over the web with the service *always on* as in the concept of utility computing. Web Services is a set of standards and a programming methods for sharing data between different software applications .Web services is a standardized way to distribute services on the Internet. Web Services achieves its goal in a technology neutral ; it provides well-defined interfaces for distributed functionalities, which are independent of the hardware platform, the operating system, and the programming language. The distributed functionalities, or services, running on different OS can communicate through web Service interfaces. Web services have special behavioral characteristics like XML-based, Loosely coupled , Coarse-grained , Ability to be synchronous or asynchronous ,Supports Remote Procedure Calls (RPCs) ,Supports document exchange

Interoperability of Web Services mainly stems from its Extensible Markup Language (XML) based open standards. The Simple Object Access Protocol (SOAP) is defined in XML. Since it is text-based and self describing, SOAP messages can convey information between services in heterogeneous computing environments without worrying about conversion problems, there are many other Web Service specifications. Two of them, which are based on XML, are Web Service Description Language (WSDL) and Universal Description, Discovery and Integration (UDDI) . WSDL defines a standard method of describing a Web Service and its capability, and UDDI defines XML-based-rules for publishing Web Service information. Messages are exchanged through the SOAP protocol. SOAP works by exchanging information using GET/POST over HTTP. This allows the data to be exchanged regardless of where the client is in the network.

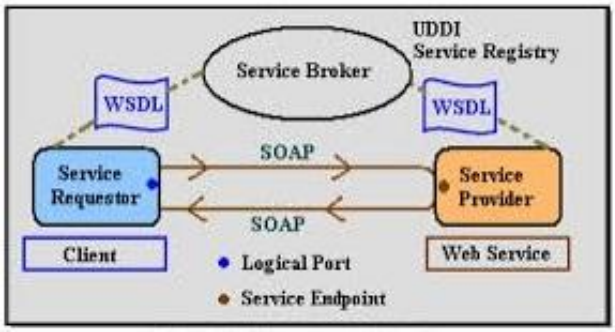

Fig 1 :Web services

People use mobile devices anytime and anywhere, they may use their mobiles to check Email, access the Internet, or run other web applications. Web Services technology recognizes mobile computing as an area to which it should expand.

Through integration, Web Services enable pervasive accessibility by allowing for user mobility as it overcomes the physical location constraints of conventional computing. the integration of mobile computing with Web Services technology will give many advantages to both sides. Mobile devices getting computationally capable, so mobile devices enabled with web services can be equal participant of web services architectures (can be web service client or web service provider).

II. Mobile devices and architecture

A. Mobile architecture

Mobile architecture allows maintaining this connection whilst during transit. Each day the number of mobile devices is increasing, mobile architecture is the pieces of technology needed to create a rich, connected user experience. Currently there is a lack of uniform interoperability plans and implementation. There is a lack of common industry view on architectural framework. This increases costs and slows down 3rd party mobile development. An open approach is required across all industries to achieve same end results and services.

Important features of a mobile architecture

- Scalability – A Mobile Architecture must be able to be utilized with all recovery requirements on both large and small scale.
- Secure – Encryption is important, transmission protocols must support encryption (SSL) via secure transit such as HTTPS
- Reliable – Reliability is always important in all technologies and mobile architecture is no different.
- The constraints of mobile device applications are very different than the traditional web or desktop applications. Unlike a desktop or web application, the screen real estate is very compact and the controls are limited in functionality.

Integration middleware was developed to allow incompatible systems to communicate. Enterprise Application Integration (EAI) is used to integrate applications inside the firewall. B2B integration extends integration beyond the enterprise to customers, partners, and suppliers. The emerging Web Services model goes even further by defining a single set of standards for integration both inside and outside the enterprise. As a result, vendors in EAI and B2B markets will be profoundly affected by Web Services.

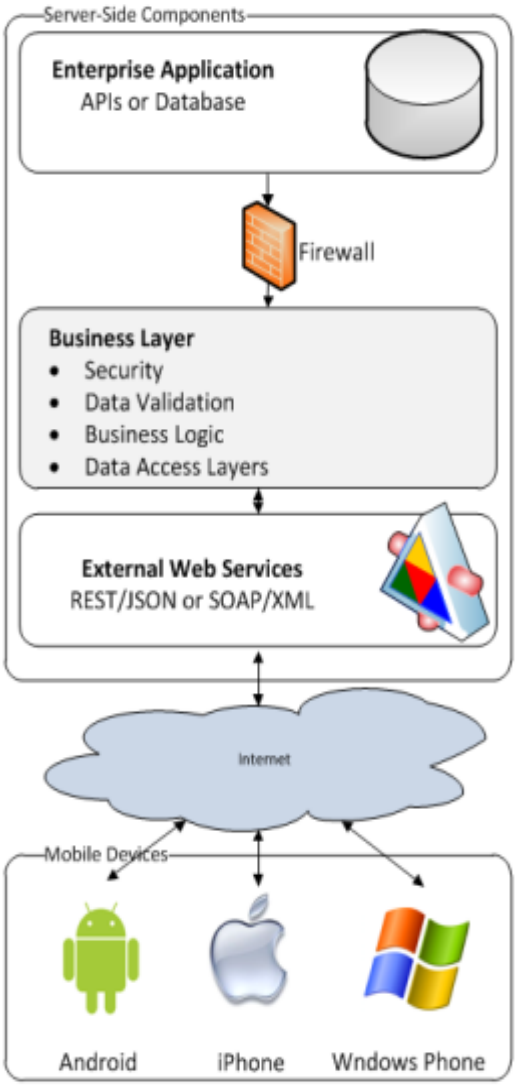

Fig 2 : Web services with mobile devices

III. ServerSideComponents

A. Enterprise Application Integration

Enterprise Application Integration vendors have introduced packaged integration solutions to help the enterprise develop a consistent approach to integration for all applications. EAI solutions generally include:

- Message-oriented Middleware. Message-oriented middleware (MOM) products provide connectivity between applications by message passing. Messages are sent to a queue and are then forwarded to the destination application for processing. This method is known as "store-and-forward" messaging.
- Application Servers Application servers also provide connectivity between applications, but instead of using a message queue, the client sends a request directly to a server and waits for the server to respond. This method is known as "request/response."
- Adapters. Most EAI solutions include standard connectors for the most common packaged applications and databases. These adapters alleviate much of the manual coding required to map data formats and object models between applications.
- Adapter Development Frameworks. Adapter frameworks are the tools that allow developers to build adapters into applications where a standard adapter does not exist or where the amount of customization of the packaged application is significant.

- Workflow and Process Management Tools. One of the key tenets of EAI is automating transaction processing. Workflow and process management tools allow the developer to define, implement, test and monitor transaction flow associated with specific business processes.

B2B Integration

Extending integration to applications outside the boundaries of the enterprise has proved equally complex. Enterprises have typically had two choices for B2B integration:

- Electronic Data Interchange (EDI). EDI enables computer-to-computer exchange of business data in a standard format, and information is organized according to a specified format set by both parties. This allows a "hands-off" computer transaction that requires no human intervention or re-keying on either end. However, EDI was expensive and lacked the flexibility to support the wide range of business processes and data formats that companies needed.
- XML Trade Vocabularies. With the arrival of eXtensible Markup Language (XML), industry specific trade vocabularies, such as RosettaNet for the electronics industry and ACORD for the insurance industry, have simplified integration between companies and created the B2B Integration market.

B. External Web services

The Web Services model represents a universal acceptance on the part of software vendors that integration middleware built on open standards is both possible and beneficial. The largest industry players are uniting behind a single set of core standards based on:

- eXtensible Markup Language (XML). XML is a universal syntax for describing and structuring data independent from the application logic. It is really a "meta-language," meaning a language that describes other languages. XML can be used to define unlimited languages for specific industries and applications.
- Simple Object Access Protocol (SOAP). SOAP is a lightweight XML-based protocol for exchange of information in a decentralized, distributed environment. It functions as a standard envelope for messages passing between different systems.
- Web Services Description Language (WSDL). WSDL is an XML grammar for specifying a public interface for a Web service. This interface describes the functional and operational requirements for accessing Web Services, such as protocol binding requirements and location information.
- Universal Description, Discovery and Integration (UDDI). UDDI is the standard that defines the repository in which available web services are stored, indexed, and organized.
- Web Services Interoperability (WS-I). WS-I in an industry consortium focused on ensuring interoperability between vendor solutions through its Web Services Interoperability Basic Profile. The consortium is also mandated to develop interoperability profiles for security and other products that leverage Web Services.
- Web Service Extensions. The core standards are being extended to address critical issues, such as reliable messaging, security, process orchestration, and long-running transactions

C. RESTful

Structured Web services as RESTful resources that can be invoked using either JSON or plain XML inputs and outputs. REST stands for Representational State Transfer and describes a pattern for interacting with content on remote systems, typically using HTTP. REST describes a way that can access and modify existing content and also how to add content to a system. RESTful HTTP Web services use the standard set of HTTP verbs to indicate the action the user wants to perform.

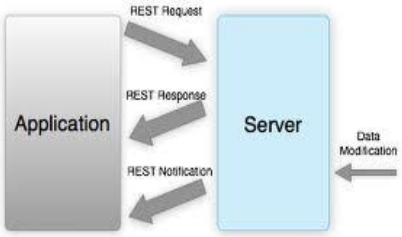

Fig 3 : REST message

It is up to the RESTful Web service to choose which formats are supported. XML and JavaScript Object Notation (JSON) are two of the most popular formats used by RESTful Web services. SAS BI Web Services supports both of these message formats.

D. XML

The format of input and output XML messages when using RESTful SAS BI Web Service endpoints mimics the SOAP format for a given stored process or generated Web service. The only differences are:

- None of the SOAP XML elements should be present in requests or responses.
- Namespaces are optional in requests. If they are used in the request, then they are used in the response. If they are used with the REST endpoint for a generated Web service, then they must match the namespace of the generated Web service. Generally, it is advisable to avoid using namespaces for the plain XML message format.
- Binary content is Base64 encoded and inlined.

The similarities between the plain XML format and the SOAP message format make it easier to write client code if there are no tools that can automatically generate client stubs.

E. JSON

JSON is a simple text-based message format that is often used with RESTful Web services. Like XML, it is designed to be readable, and this can help when debugging and testing. JSON is derived from JavaScript, and therefore is very

popular as a data format in Web applications. Because JSON has extensive support in JavaScript, it is often used in AJAX Web applications for creating rich, dynamic user experiences that incorporate remote data and service execution. However, JSON can be read and written by many programming languages.
JSON has only a limited set of basic, built-in data types.

F .SUPPORTED TYPES OF INPUT AND OUTPUT FOR XML AND JSON MESSAGES

When using the JSON and XML message formats with RESTful Web service endpoints, the type and format of inputs and outputs .They can produce three types of output: output parameters, data targets, and packages. Prompts enable to supply simple parameters for stored process execution. The SAS prompting framework enables to create prompt definitions that describe the type and format of stored process's input parameters. When a web service is invoked , the prompting framework validates the values that is supplied for the stored process prompts. The prompting framework translates these values into a macro variable that can be used in stored process. Data sources enable to supply arbitrary streams of data to stored process to be processed. These data streams can be text such as XML or raw binary content. The data that supplied in a data source is streamed to a fileref in stored process. Output parameters contain the values from macro variables from stored process after it is finished running. This enables to return simple values from the stored process. Data targets are like data sources in that they are streams of arbitrary data. However, data targets are produced by the stored process. Stored processes can use the SAS Publishing Framework to create a package during stored process execution. A package is a collection of assets that can be returned to the stored process client (in the case of SAS BI Web Services, the client is the Web service client). Packages can contain binary and textual data and are a convenient way to package complex reports that contain multiple images and text or HTML content produced by SAS. If a stored process returns a package, that package can be returned to the Web service client as a list of entries and the contents of those entries.

Supported Input and Output for XML Messages

When using RESTful XML SAS BI Web Services endpoints, all types of input and output are supported. send XML requests using the same format as for SOAP messages. Always omit SOAP elements from requests to the RESTful XML endpoints, and can also omit namespaces. SOAP Message Versus Plain XML Message

SOAP Message

```
<soapenv:Envelope
 xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
 xmlns:biw="http://www.sas.com/xml/namespace/biwebservices">
  <soapenv:Header/>
  <soapenv:Body>
    <biw:addfloats>
      <biw:parameters>
        <biw:num1>2.3</biw:num1>
        <biw:num2>4.2</biw:num2>
      </biw:parameters>
    </biw:addfloats>
  </soapenv:Body>
</soapenv:Envelope>
```

Plain XML Message

```
<addfloats>
  <parameters>
    <num1>2.3</num1>
    <num2>4.2</num2>
  </parameters>
</addfloats>
```

The other difference between SOAP and plain XML messages is how binary content is handled. Data sources, data targets, and packages can all contain binary data. SOAP endpoints use WS-* standards to handle binary content.

Supported Input and Output for JSON Messages

Because JSON supports only a limited number of native data types, JSON SAS BI Web Service endpoints accept only simple values for stored process prompts and can return only results from output parameters (not data targets or packages). The JSON SAS BI Web Service endpoints support prompt values that are supplied using application/x-www-form-urlencoded encoding in the HTTP request body. It must use HTTP POST when supplying values to JSON endpoints and can use HTTP GET when stored process does not require any input. For example, a JSON invocation of a stored process that adds two numbers, num1 and num2, might look like the following:
JSON Invocation of a Stored Process

Request

```
num1=2.0&num2=3.5
```

Response

```
{   "outputParameters":
 {    "Sum": "5.5"
 }
}
```

Be sure to specify the Content-type HTTP header as application/x-www-form-urlencoded when use POST. The following table lists supported prompt types for JSON messages (note that no multi-value prompt types, including ranges and selections, are supported when using JSON):

Table :Supported Prompt Types for JSON Messages

Prompt Type	Supported	Notes
Text	yes	
Numeric	yes	
Color	yes	Supports hexadecimal values only.
Date	yes	Supply a text value that matches the date type in the prompt definition.
Time	yes	
File	yes	
Data source	yes	
Data library	yes	Supply this in the form libraryPath :: libref. For example, /Products/SAS Intelligence Platform/Samples/STP Samples(Library) :: stpsamp is a valid data library value.
Data source item	yes	Supply this in the form dataSourceLocation :: dataSourceType :: columnName :: columnLabel :: columnType.
OLAP member	yes	
Ranges	no	
Multi-value prompts	no	

Accessing RESTful Web Service Endpoints

RESTful plain XML endpoints are available from the server resource /SASBIWS/rest/. When accessing endpoints for stored processes, append the SAS folder path for the stored process to the resource /SASBIWS/rest/storedProcesses/ (for example, /SASBIWS/rest/storedProcesses/stored/process/path). Generated Web services have a unique name and contain either a single or multiple named stored processes. To access the RESTful plain XML endpoint for generated Web services, append the generated Web service name to the resource /SASBIWS/rest/ (for example, /SASBIWS/rest/generatedServiceName). Because generated Web services can contain multiple stored processes,

For both generated and stored process Web services, SAS BI Web Services are interested only in a single specific aspect of the stored process output. It can be done by accessing the Web service at a specific URL. When doing this, the Web service returns only that specific output resource. Output parameters, output streams (also called data targets), and packages are all supported output resources. To access a specific output parameter, append the resource /parameters/parameterName to the RESTful URL for the Web service, replacing parameterName with the name of the actual parameter. When accessing the /parameters/ resource of a RESTful Web service, the HTTP response body contains only the string value of that parameter and no additional XML.

To access a specific output stream, append the resource /streams/streamName to the RESTful URL for the Web service, replacing streamName with the name of the actual stream. When accessing the /streams/ resource of a RESTful Web service, the HTTP response is the exact output that stored process sends to that stream and the HTTP response headers include an appropriate content type if it is available. A stored process can produce an output package during execution that can contain any number of entries. It can be accessed as an individual entry within a package by appending the resource /packages/entryNum to the RESTful URL for the Web service, replacing entryNum with the index of the package entry. Entries in packages are not always named, so must use the package entry index. The index starts at 0 for the first entry in the package.

IT cannot access more than one output resource at a time by appending multiple /parameters/, /streams/, and /packages/ resources. It can be used only one at a time.

Accessing RESTful JSON Web Service Endpoints

JSON Web service endpoints are available from the server resource /SASBIWS/json/. When accessing endpoints for stored processes, append the SAS folder path for the stored process to the resource /SASBIWS/json/storedProcesses/ (for example, /SASBIWS/json/storedProcesses/stored/process/path). Generated Web services have a unique name and contain either a single or multiple named stored processes. To access the JSON endpoint for a generated Web service, append the generated Web service name and the stored process name to the resource /SASBIWS/json/ (for example, /SASBIWS/json/generatedServiceName/storedProcessName). JSON endpoints do not support output resource specifications, so the response is always JSON data.

IV. CONCLUSION

This paper presented survey on the architecture of web services in tie up with different mobile based systems and their supported formats in web services It also describes the server side formats with integration of webservices in Enterprise application , B2B along with RESTful web services . RESTful Web services are "Resources" that are identified by unique URIs. These resources are accessed and manipulated using a set of uniform methods (GET, POST, PUT and DELETE) where each resource has one or more representations (XML, JSON, Text, User-defined, etc) that are transferred between the client and the service during a Web service invocation. RESTful Web services are perceived to be simple because REST leverages the existing well-known W3C/IETF standards (HTTP, XML, URI and MIME type) as well as the necessary infrastructure that has already become pervasive . the requests and responses for RESTful Web services are typically HTTP messages that are far less in size compared to SOAP messages. In REST architecture a

resource can be directly identified by its URI, therefore extensive SOAP parsing can be avoided that is required for invoking a service .

REFERENCES

[1] http://www.w3.org/
[2] http://msdn.microsoft.com/en-us/library/dd483215.aspx
[3] http://www.datamation.com/entdev/article.php/11070_3612721_2/Web-Services-on-Mobile-Devices.htm
[4] http://ieeexplore.ieee.org/xpl/articleDetails.jsp?reload=true&arnumber=6263811
[5] "Web Services in Mobile Devices " paper by Varun Goyal
[6] Performance Analysis of web services on mobile devices
[7] "web services" By B.V.Kumar, S.V.Subramanyam
[8] "performance Evaluation of RESTful web services for Mobile Devices

