Merging the Worlds of the Mainframe and the Web

For decades, the computer industry has struggled with the problem of how to get heterogeneous systems to interact and to pass data in a form that is self-describing and through interfaces that are open. Probably nowhere is that problem more acute than in situations where open systems need to access and use mainframe data.

Fortunately, Web services and eXtensible Markup Language (XML) represent a great leap forward in making such access and interaction possible. This article first presents an overview of XML and Web services focusing on the void that they fill. The article then explains the relevance of Web services to the mainframe and gives an example of how Web services and XML are being implemented to enable the mainframe to participate in this new technology.

Introduction to XML and Web Services

With the advent of any practical technology comes the inevitable deluge of acronyms. These acronyms (e.g., XML, WSDL, SOAP, DISCO, UDDI) tend to mystify rather than clarify the features and advantages the technology offers. The goal of Web services is to provide a mechanism with which data can be provided and consumed using cross-platform standards. The goal of Web services is the exchange of information in a platform-neutral manner.

The key problem is how to make Product X talk to Service Y and then export the data to Customer Z. Exchanging data between applications is a non-trivial task. A divergence of operating systems (e.g. OS/390, Unix/Linux, Windows), development platforms (e.g. Java, VBA, C++, VB.NET), and data formats (e.g., DB2, VSAM, IMS, Excel) all contribute to the complexity of the task.

Enter XML, the Holy Grail of data exchange. XML is a standard backed by all the major players (e.g. IBM, Sun, Oracle, Microsoft). As a standard there are well-accepted protocols associated with accessing, generating, validating and converting XML. Do all the players and all versions of their products support the standard and affiliated protocols with the same level of compliance? Of course not, but a middle ground can be reached that enables XML to behave in a cross-platform manner.

(A side note. When purchasing a book on XML, get as recent a text as possible. As a standard, XML was much more volatile in 1998 through 2001. This standards volatility extended into early 2002.)

XML is composed of a set of standards that, in turn, are used to build practical entities and protocols. For example, the base protocol for Web services, the Simple Object Access Protocol (SOAP), is built on top of XML. The basic idea of XML is to have one standard parser/editor and keep all data in that form. For example, this might be useful in representing the documentation provided with an SDK or representing the project files

used to build source code. Both of the aforementioned XML uses are exploited by Microsoft's Visual Studio .NET and IBM's WebSphere Application Studio.

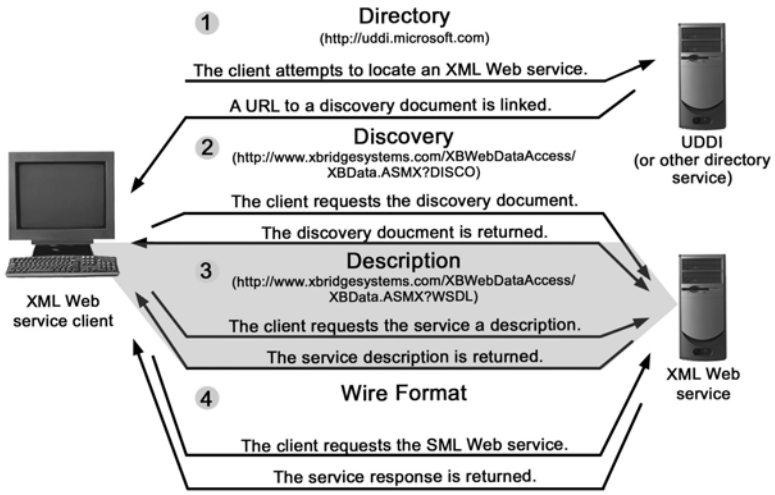
The purest roots of XML are in Standard Generalized Markup Language (SGML). The pervasiveness of SGML was limited compared to more prolific data-exchange formats such as CSV (Comma Separated Value) and HTML. CSV is primitive but useful, and HTML focuses on how data is displayed (e.g., boldface, italics) and not on what the data is meant to represent. Concepts such as data conversions, validations and typing are not found in either HTML or CSV.

What Web Services Need to Provide

The XML standard emphasizes what the data is and not how it is displayed. Still, XML, in and of itself, does not provide the infrastructure needed for Web services. The following diagram demonstrates the infrastructure associated with Web services and their consumers.

The diagram can be broken down as follows:

- Step 1 (Optional) – A Web service can be registered with a directory service. Many directory services expose their list of Web services using the Universal Directory Discovery Interface (UDDI) protocol. A client can use a directory service in order to find a Web service that implements the functionality required by the client.



- Step 2 (Optional) – A Web service creates a discovery document (DISCO), which defines both the discovery document standard (an XML-formatted standard) and the protocol for retrieving the discovery documents.
- Step 3 – A Web service needs to publish the format it used for input into and out of the server. The input and output are messages. Web Service Description Language (WSDL) is the XML-based representation used to describe the format of these messages and the way the messages are exchanged. The protocol used for message exchange is SOAP, which (to no one's surprise) is an XML-based representation.
- Step 4 – Knowing where the service is (thanks to UDDI), what the service has to offer (DISCO), what the service expects for input/output and how this is presented (WSDL), a client can make requests. Such requests are HTTP requests

or, in layperson's terms, the client is surfing the Web by providing URLs. The URLs pass data to the Web server exposing the Web service, and the standard HTTP reply mechanism is used to return the results of the operation.

There is no need to memorize XML standards because vendors provide tools for working with these standards. Web servers handle the nuances of the HTTP protocol. IBM, Sun, Microsoft, BEA and many other companies provide the tools for developing and interoperating with Web services. Memorizing XML intricacies may be needed but only in certain limited circumstances.

The last piece of the puzzle is how to transport the messages. SOAP provides the cross-platform communication. SOAP describes a request message used to access a Web service and the response message used by the service to send results back to the client. In theory, SOAP can be sent over any communications protocol; in practice, HTTP is the mechanism used in the cross-platform (heterogeneous) world of Web services.

The XML used to describe a SOAP message can be broken down as follows where the theme of the XML is that (SOAP describes a message):

- Envelop – the root level XML node that contains the message
- Header – an optional element within the SOAP message. The header contains information that influences how a message is processed (such as username and password)
- Body – contains the payload of the message (the actual data being processed by the message)
- Fault (optional) – shows errors in the body (when things go wrong)

Web Services and the Mainframe

What does all this have to do with mainframes? A lot. Information is the enterprise's most important asset. Approximately 70 percent of business data is stored in OS/390® and z/OS™ mainframes, and this situation is not likely to change soon. The mainframe's performance, stability, security and scalability make it the most popular method for storing historical and current data. Finding a way to access that critical information and make it available to applications and users across many different platforms enables a company to discover new ways of exploiting what they already know.

The goal is that the mainframe should look like just another application platform on the Net, and its data would be useable by all connected applications. The practical application of Web services and XML to retrieve mainframe data would be to define and implement a set of Web services that would provide the read-only functionality of Open DataBase Connectivity (ODBC), Java DataBase Connectivity (JDBC), ActiveX Data Objects (ADO) or ADO.NET-type interfaces.

The Web services needed to provide this function will run on a mid-tier server and be described by WSDL. The protocol used for message exchange is SOAP. The specific

Web services needed to enable data to be delivered from the mainframe in a relational structure are described below.

MainframeDataConnect – A Web service is provided to establish a session with the mainframe computer. This service would validate the user's mainframe UserID and Password. When the session is established, a token will be returned that will anchor this session to the other services. This service will also be used to close the session. All returned information will be in XML form.

MainframeGetTables – A second Web service is provided to retrieve from the mainframe a list of Tables and/or Views that are available to access. Once a Table and/or View is selected, this service will be used to retrieve the column names and the properties for the Tables and/or Views. All returned information will be in XML form.

MainframeExecute – This service is invoked to execute a SQL-like command string that will retrieve the data for the Table, View or Columns requested. The token generated at session initiation would be provided to the service for each call. The result set for this query is returned as an XML data stream ready to be used by any XML-literate application.

MainframeAbort – The final service provided supports an asynchronous termination of the in-process command. The function is useful when it is necessary to abort a "Runaway Query." The token generated at session initiation will be provided to the service for each call.

Practical Application of XML and Mainframe Technology

To appreciate what XML could mean for mainframe applications, consider the real case of a state government department that needed to make available current information on the status of purchase requests.

Departments required access to a Purchase Authority (PA) document from the Web application platform; the end users needed to continually interact with the system in order to process their purchase requests.

Previously, users tried to access the forms via the data warehouse, using a legacy tool. However, there were definite limitations with that application. Requests were often delayed (up to three days), dropped or simply became "runaways." Users could not sort the data, which was "typically out of date and hard to use effectively." A new request was required every time a PA was needed. The application could not process more than one request at a time.

Approach – Recognizing that their existing application did not meet their needs, the organization moved to a new technology that enables Windows®, Web and mobile applications to securely access data in System/390® and z/OS® mainframes, and to present that information in the format of the requesting application, all in real time. The

technology quickly and securely accesses virtually all OS/390 mainframe data including Virtual Storage Access Method (VSAM), (Information Access Method (IAM®), Queued Sequential Access Method (QSAM), DataBase 2 (DB2) and Information Management System (IMS DB),

Even more essential for the success of this application, the technology supports XML and ASP.NET, as well as the DISCO, UDDI, WSDL and SOAP standards.

Here's how the application works. The user calls up a form from the department's Web application platform and enters information for a query.

The query is passed onto the mainframe-data-access software, which accesses a DB2 table and several VSAM files on the department's mainframe.

A custom ASP.NET application leverages the technology to build an XML representation of the requested data. The forms engine seamlessly processes the XML-formatted data and presents it to the user in HTML. The new capability required limited custom development, but no additional programming or rewriting of applications. In effect, the technology stripped away the complexity and made implementation extremely easy for the IT staff and the process virtually transparent for the end user.

Results – The application meets the department's needs for on-demand access to real-time data. The implementation provides this capability at a fraction of the cost and time the department had been spending on its legacy application, which did not meet the staff's needs.

The new source of XML data is far more reliable than the XML stream generated by the previous application. Also, the new XML stream can handle multiple simultaneous requests, something that the department had been unable to do even after working with the previous implementation for two years.

Furthermore, the new application is so flexible, the department has been able to significantly enhance the system's error-handling capabilities to accommodate invalid user input and old/missing data on the mainframe.

Conclusion

The technology behind Web services and XML is reaching a level of maturity and stability. Interoperability (cross-platform development, deployment within a heterogeneous environment) is achievable.

The challenge now is to implement the Web services technology as the application described above does, thus making available all of OS/390 and z/OS's critical business information to the shared community of Web services and Web applications.

About the Author

Ron Hankison is president and CEO of Xbridge Systems, Inc., developers of Xbridge Host Data Connect. Hankison, who has more than 20 years experience with mainframe technology, is one of the industry's foremost experts in mainframe data access. Email: Hankison@xbridgesystems.com. Website: www.xbridgesystems.com