

RTCA Special Committee 186, Working Group 3

ADS-B 1090 MOPS, Rev. A

Meeting #7

**Separating Data From The Transport
In Support of FIS-B Services**

Presented by Mike Culver, Microsoft Corporation

SUMMARY

The author proposes separating data from the underlying transport protocol in order to make data delivery extensible and standards-based. In addition, he proposes that adopting XML as a data format, while recognizing there are implementation issues (addressable) that need careful attention in order to make XML function in a multicast data environment, as well as recognizing that there will be a performance degradation.

Separating Data From the Transport

Mike Culver
Group Program Manager, .NET Framework Early Adopter Program
Microsoft Corporation
mculver@microsoft.com
425.703.8419

Summary

The author proposes separating data from the underlying transport protocol in order to make data delivery extensible and standards-based. In addition, he proposes that adopting XML as a data format, while recognizing there are implementation issues (addressable) that need careful attention in order to make XML function in a multicast data environment, as well as recognizing that there will be a performance degradation.

Discussion

By divorcing data from the transport, it becomes possible to devise new data delivery methods on an "as needed" basis, without requiring negotiations for bandwidth on existing channels. Essentially this recognizes the advent of "Internet dial tone" in General Aviation cockpits.

By utilizing XML, an industry standard, inter-op, and extensibility are assured.

The software industry is quickly rallying around XML and XML Web services as a new data delivery mechanism that extends the reach of software applications, and that empowers the rise of an entire new class of devices and solutions. Including devices in aircraft.

No stronger evidence exists that the explosion in cockpit applications and devices available today at very modest costs (often less than \$1500 for hardware and software combined). These devices are finding their way into the low end of the General Aviation fleet, and 2001 sales of cockpit devices capable of consuming XML: Web services are very strong.

What is XML?

The following characteristics define it (paraphrased from the W3C website):

- A way of representing data in a self-descriptive manner. For example:

```
<aircraft>
  <N-Number>2660K</N-Number>
  <Make>Luscombe</Make>
  <Model>8E</model>
</aircraft>
```
- XML offers extensibility, internationalization support, and platform independence. It is extensible because someone could add Owner Name to the example above, without disturbing applications that are unaware of the new field. Internationalizable because Unicode characters are supported, and platform independent because it's a message format agreed upon by the industry.
- XML looks vaguely like HTML, but isn't. Like HTML, XML makes use of tags (words bracketed by '<' and '>') and attributes (of the form name="value"), but while HTML specifies what each tag & attribute means (and often how the text between them will look in a browser), XML uses the tags only to delimit pieces of data, and leaves the interpretation of the data completely to the application that reads it.
- XML is text, but isn't meant to be read

- XML is verbose, but that is not a problem since XML is a text format, and it uses tags to delimit the data, XML files are nearly always larger than comparable binary formats. That was a conscious decision by the XML developers. The disadvantages are usually compensated for at a different level. Disk space isn't as expensive anymore as it used to be, and programs like zip and gzip can compress files very well and very fast. Those programs are available for nearly all platforms (and are usually free). In addition, communication protocols such as modem protocols and HTTP/1.1 (the core protocol of the Web) can compress data on the fly, thus saving bandwidth as effectively as a binary format.
- XML is license-free, platform-independent, and well supported. That is, it's an open standard
- XML represents binary data (images) via UUENCODE, a defacto standard

Why Does XML Matter?

Simply put, the software world is turning faster than anyone can catch up. New applications are coming to market on a monthly basis that provide functionality thought impossible just a year ago. And these new programs are almost immediately leapfrogged by even more creative ones! XML Web services are a solution because they are a standard, and don't require constant developer attention in order to develop additional (new) applications.

Accordingly, there are several new rules that apply:

1. Data should be abstracted from its application (that is, allow you to use my data in new ways)
2. Data should be separate from the transport. I may see an application for weather radar in the cockpit, but you may have a new and faster way of delivering the data. Or perhaps you want to deliver the data to a moving map in an alternate vehicle, such as an automobile. No single, fixed, transport protocol is appropriate.
3. There should be a well-known location where you can discover my data feed. (www.uddi.org)
4. There should be a well-known interface that I can query to find out how to consume your data. My architect and your architect shouldn't need to have lunch together. That is, if we need to meet to agree on a common interface then the application can't scale (too expensive to meet all these people).
5. Authentication is often important, in order to control inappropriate uses of the data

XML Web Services

Think of XML Web Services as a way to make programming calls across the Net. For example, a program can call a function called `GetWeather(Lat, Long, Radius)` that returns a weather image from across the Internet. From the programmer's point of view, it doesn't matter where the data came from (local machine vs. National Weather Service). That's an overstatement, in that bandwidth and connectivity are issues, but from an architectural point of view there's no difference.

Message format across the Net is an XML protocol known as SOAP (Simple Object Access Protocol), also a standard. This is usually a request/response protocol, although one-way messages are supported. Also, SOAP is not just for use over HTTP. See the SOAP-RP (http://www.gotdotnet.com/team/xml_wsspecs/soap-rp/default.html) and DIME specifications (http://www.gotdotnet.com/team/xml_wsspecs/dime/default.htm) for a more general and efficient approach.

Environmental Issues: Streaming vs. Request/Response

XML per-se is agnostic about how it's delivered. After all, it's nothing more than a way to represent data. However XML is a "balanced" approach to data representation, in that all tags are paired. That is, for each opening tag, there is a closing tag – including at the end of the document.

As stated above, XML Web services are usually request/response based, and usually ride on top of traditional Internet protocols such as TCP/IP and HTTP. However the SOAP protocol does support one-way messages.

So how does this fit into a streaming environment? Perhaps not at all; maybe it's better to simply use a satellite phone and provide Internet connectivity. However the author feels that there is a compelling benefit to pilots if they are able to choose the most appropriate connection for their specific needs.

One approach is making certain that each packet is self-contained. That is, a given node in the XML document is complete within a packet, perhaps something like `<p>45E1B0C7</p>` to represent a fraction of the image. Overall image transmit time goes up, but compatibility is ensured. If any particular packet gets lost, overall document integrity is affected in a minimal manner. This point may or may not be relevant in the final design.

Further technical discussion seems appropriate...