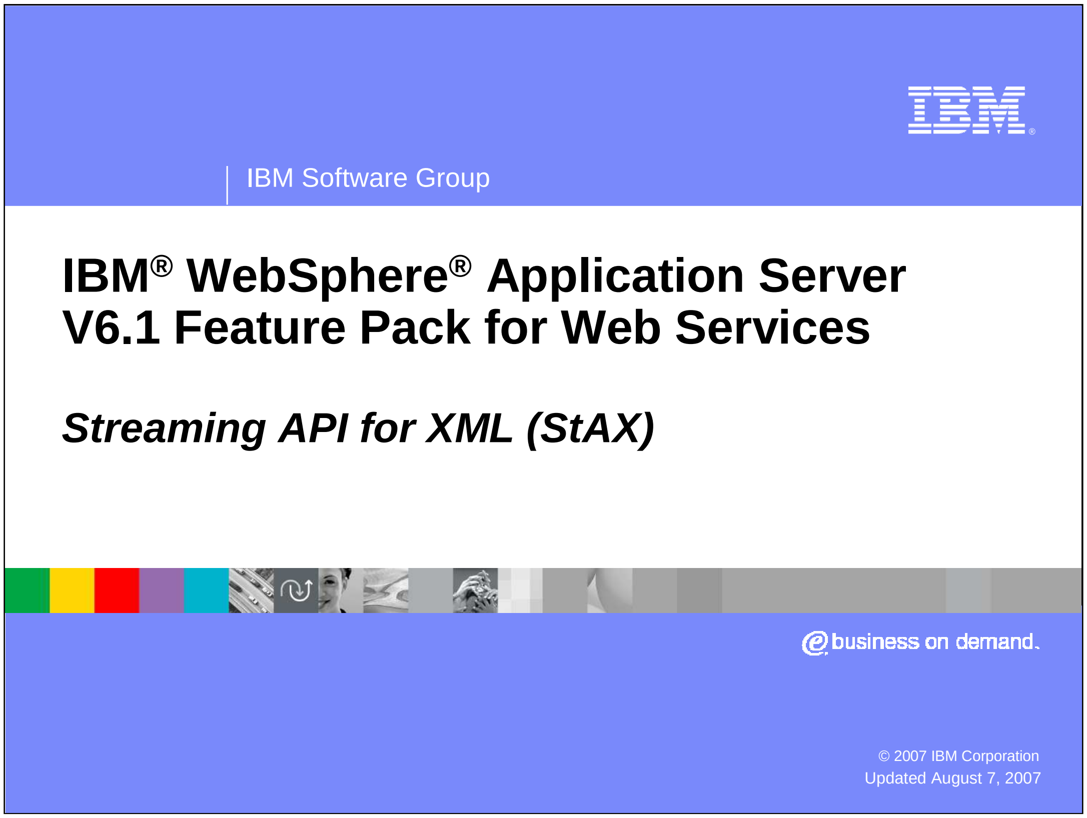


IBM Software Group

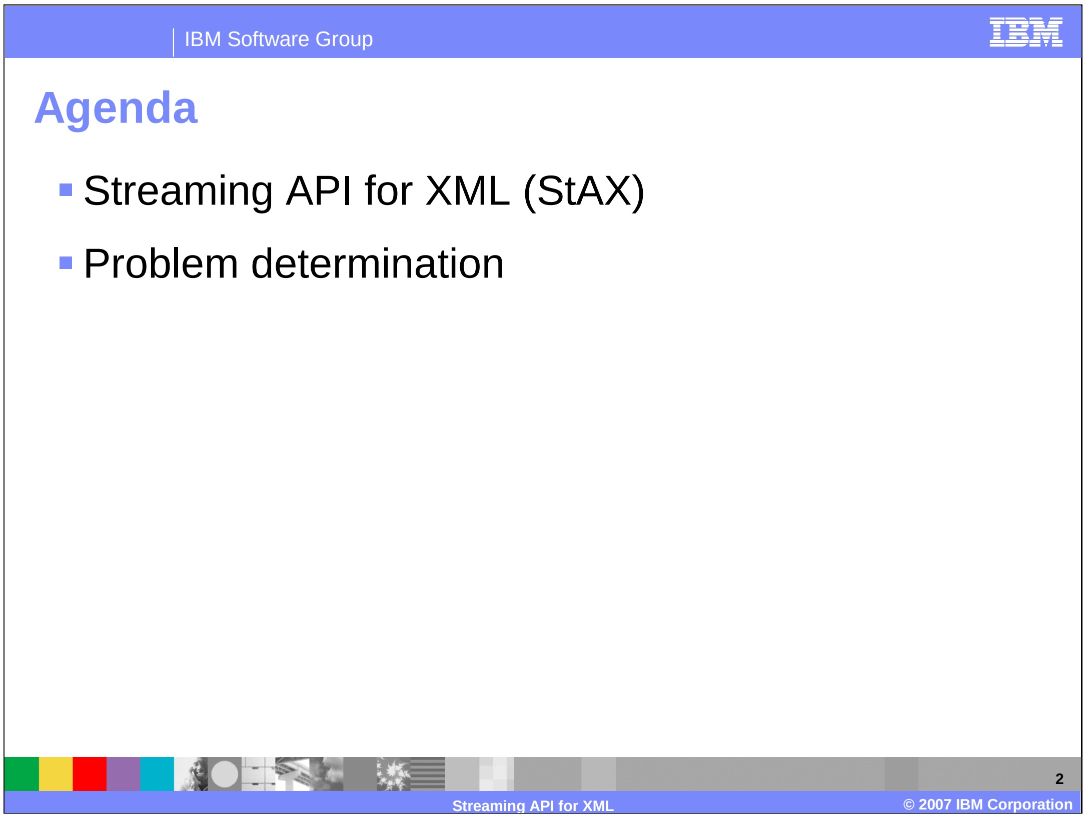
IBM® WebSphere® Application Server V6.1 Feature Pack for Web Services

Streaming API for XML (StAX)

business on demand.

Updated August 7, 2007

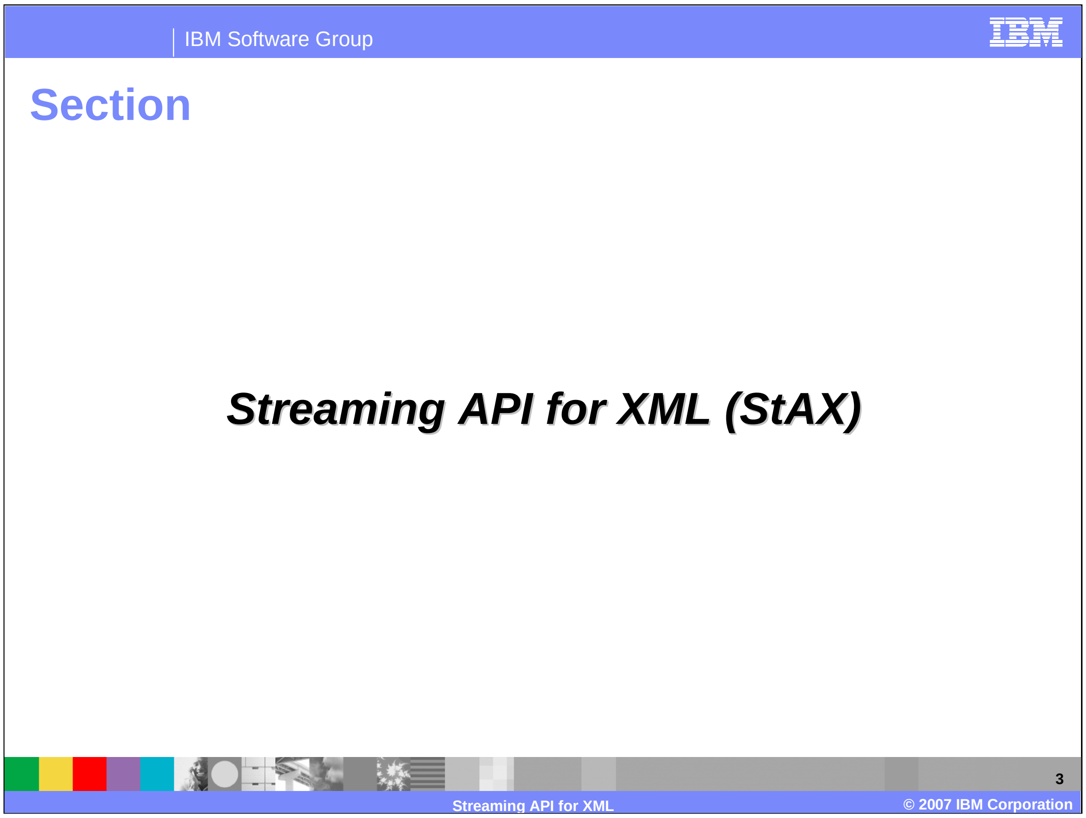
This presentation will provide an overview of the Streaming API for XML or StAX technologies provided by the Feature Pack for Web Services.



Agenda

- Streaming API for XML (StAX)
- Problem determination

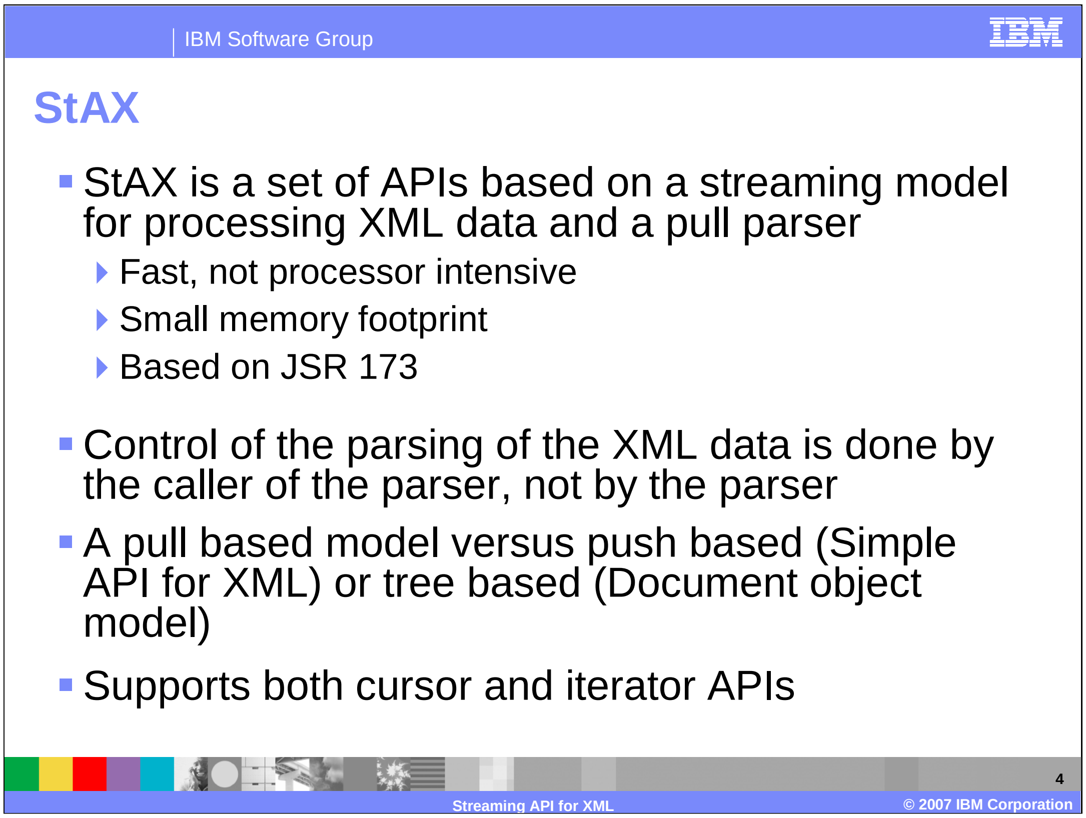
This presentation will explain the Streaming API for XML or StAX support in the Feature Pack for Web Services, and explain problem determination for this component.



Section

Streaming API for XML (StAX)

This section will provide an overview of the Streaming API for XML (StAX).



IBM Software Group

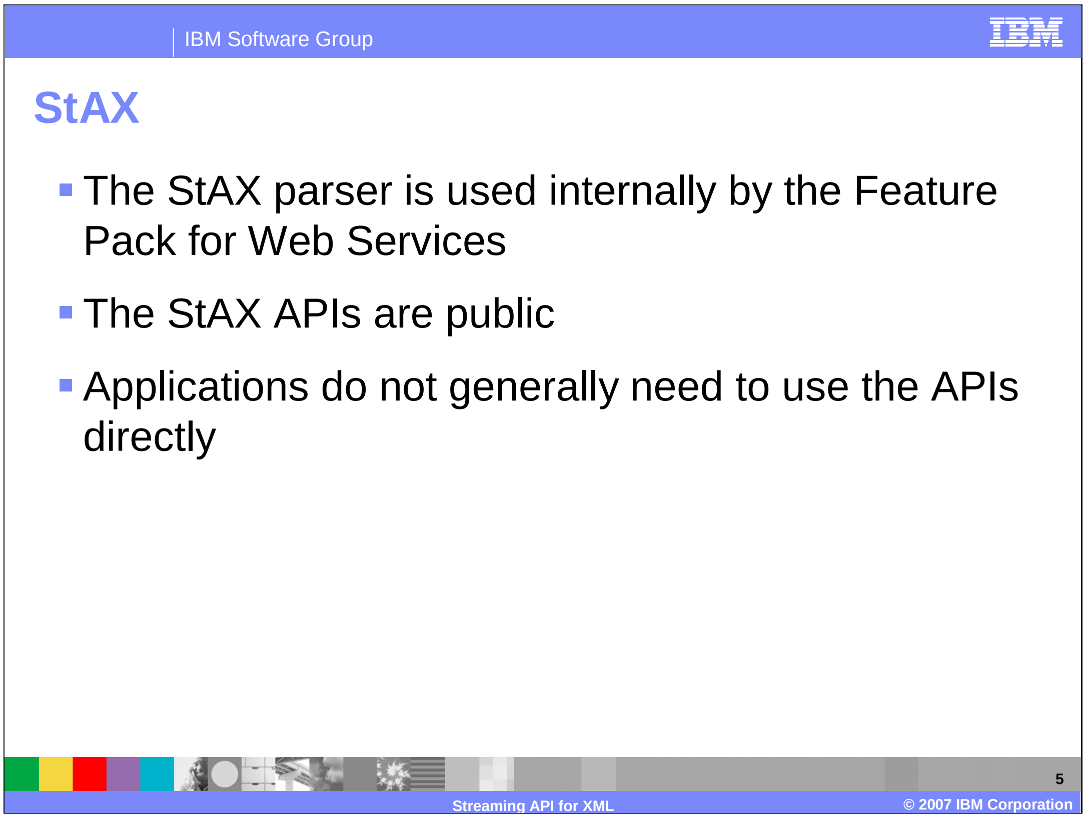
StAX

- StAX is a set of APIs based on a streaming model for processing XML data and a pull parser
  - Fast, not processor intensive
  - Small memory footprint
  - Based on JSR 173
- Control of the parsing of the XML data is done by the caller of the parser, not by the parser
- A pull based model versus push based (Simple API for XML) or tree based (Document object model)
- Supports both cursor and iterator APIs

4

Streaming API for XML

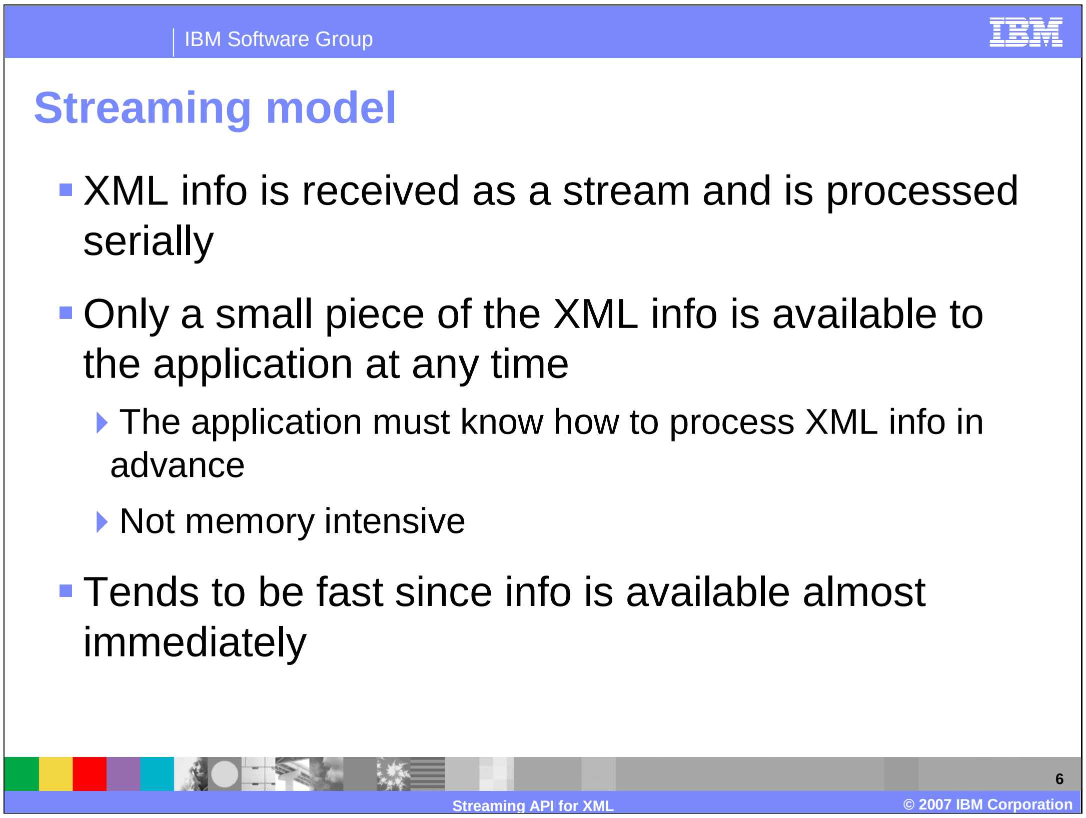
StAX provides a set of APIs for the stream (serial) processing of XML data. It is fast to start because there is no complex data structure to construct as in tree based processing models such as the Document Object Model (DOM). StAX uses a model know as “pull parsing” in which the application code call methods in the parsing library as contrasted to “push parsing” in which the parsing code calls methods in the application code. This is a more natural model for most programmers. StAX provides a *cursor* API that allows the programer to traverse each part of an XML document from beginning to end. It also provides an *iterator* API in which the XML stream is viewed as a sequence of event objects.



StAX

- The StAX parser is used internally by the Feature Pack for Web Services
- The StAX APIs are public
- Applications do not generally need to use the APIs directly

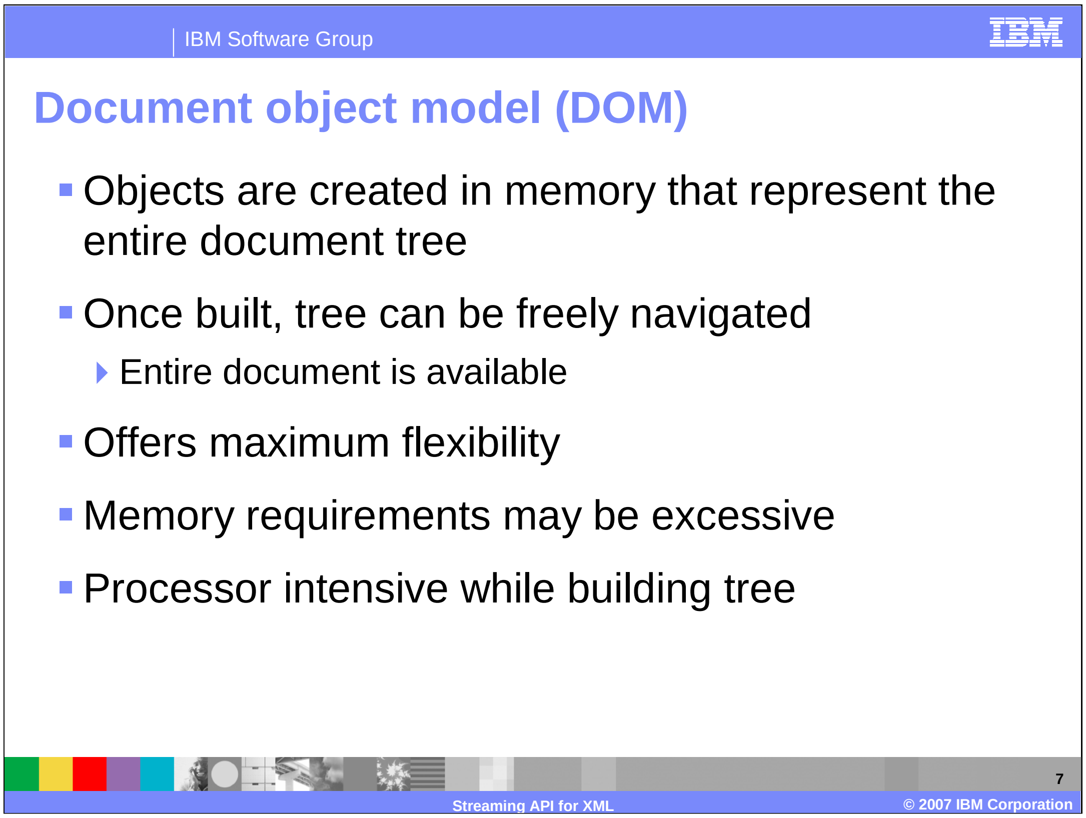
The StAX parser is used internally by the Feature Pack for Web Services runtime and most applications will not need to be aware of it. However, for cases where there is a need to process application specific XML data, the StAX APIs are public.



Streaming model

- XML info is received as a stream and is processed serially
- Only a small piece of the XML info is available to the application at any time
  - The application must know how to process XML info in advance
  - Not memory intensive
- Tends to be fast since info is available almost immediately

As stated earlier, StAX uses the streaming model for processing XML data and only a small piece of XML information is available to the application at any time. It is only possible to move forward in the XML data stream – it is not possible to "back up" to a previous position. The streaming model tends to have a small memory footprint since there is no requirement to build large data structures in memory and tends to be very fast, especially when starting up.



IBM Software Group

IBM

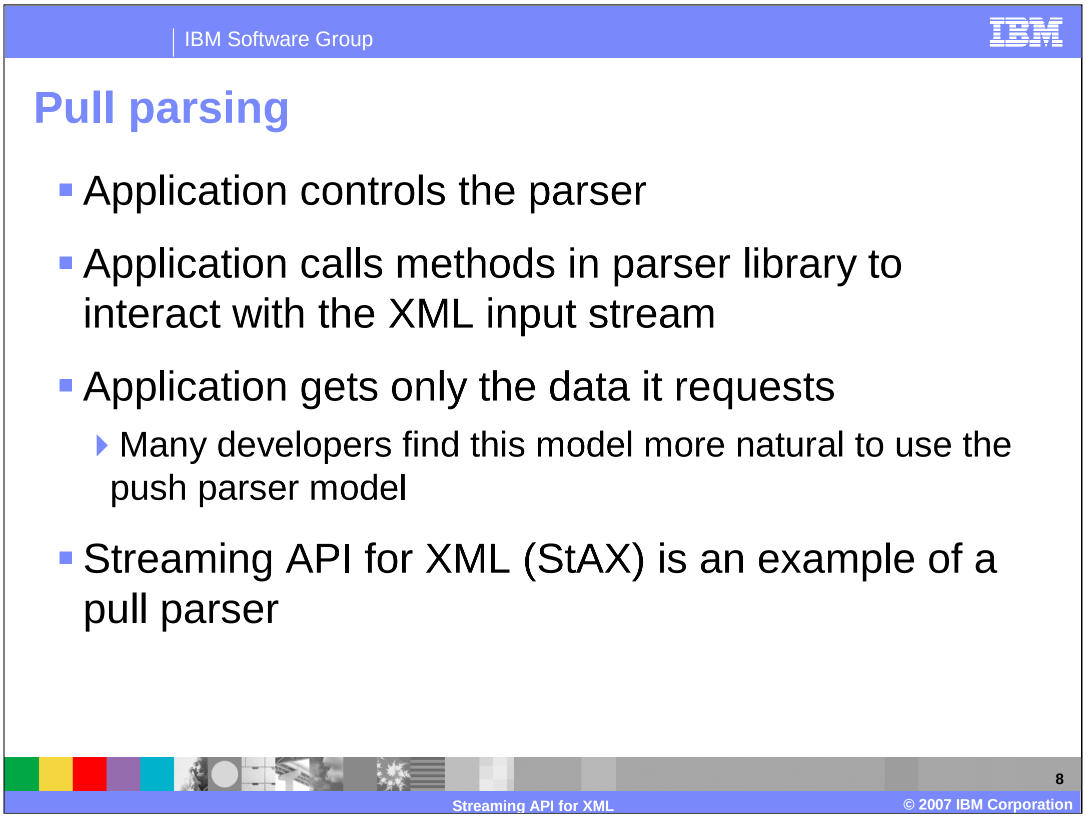
Document object model (DOM)

- Objects are created in memory that represent the entire document tree
- Once built, tree can be freely navigated
  - Entire document is available
- Offers maximum flexibility
- Memory requirements may be excessive
- Processor intensive while building tree

7

Streaming API for XML

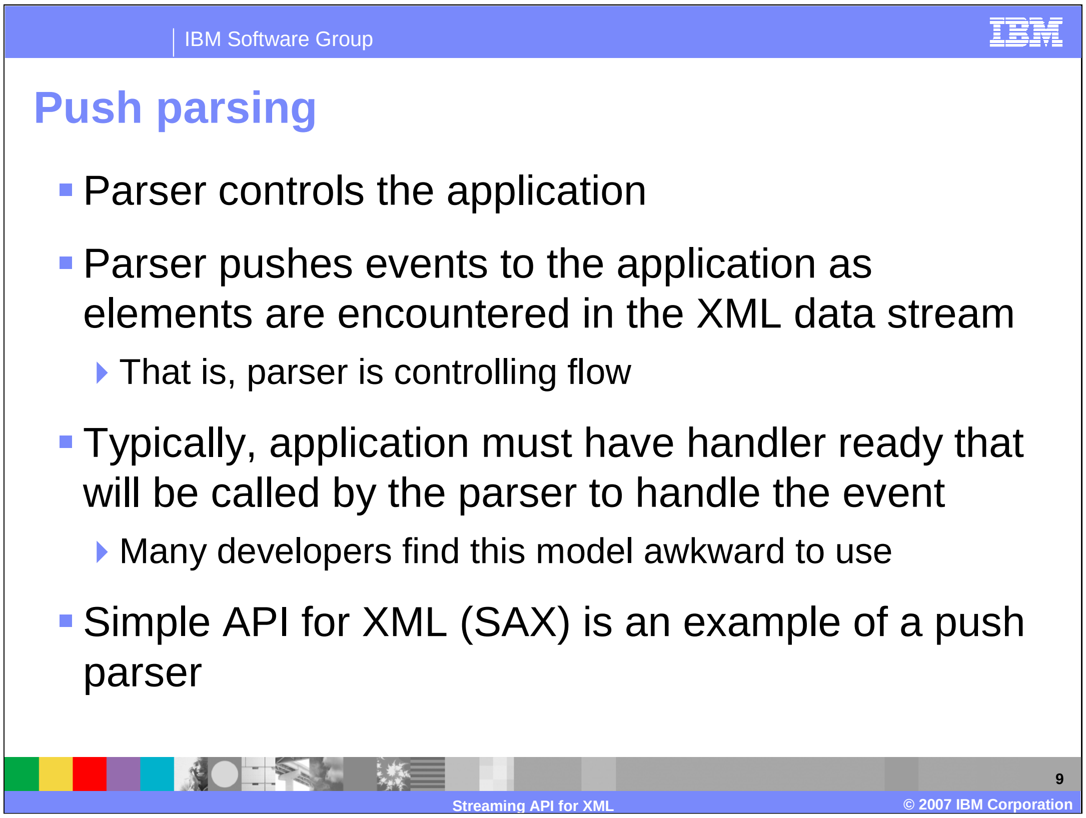
Another approach to processing XML data is known as Document Object Model (DOM). In this model, a tree is constructed in memory that represents the entire document. Once the tree is built, it can be freely navigated, that is, you are not limited to only forward movement. This approach offers maximum flexibility but at the potentially high costs of memory footprint and startup time, since building the document tree in memory can be a very expensive operation for large documents.



Pull parsing

- Application controls the parser
- Application calls methods in parser library to interact with the XML input stream
- Application gets only the data it requests
  - Many developers find this model more natural to use the push parser model
- Streaming API for XML (StAX) is an example of a pull parser

In pull parsing, the application makes calls into the parsing library to interact with the XML data stream. The application gets only the data it requests and only when it requests it. StAX uses this processing model. Many developers find this approach to parsing easier to develop applications for.



IBM Software Group

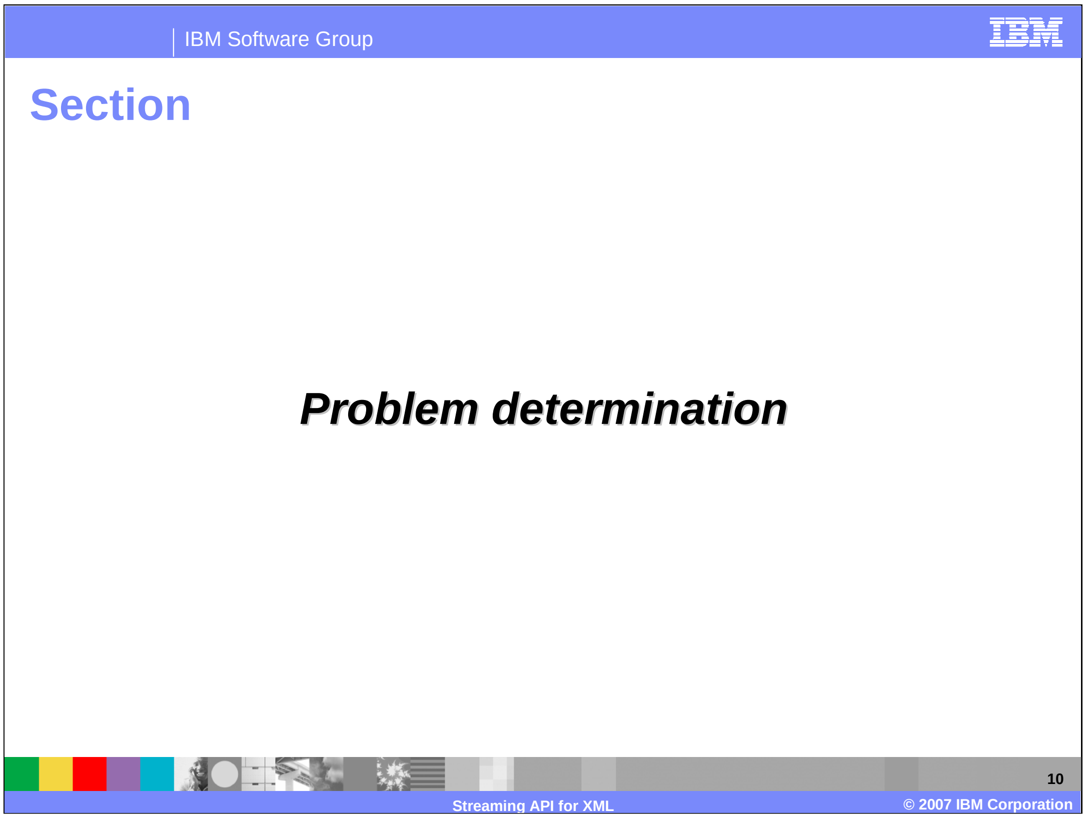
Push parsing

- Parser controls the application
- Parser pushes events to the application as elements are encountered in the XML data stream
  - That is, parser is controlling flow
- Typically, application must have handler ready that will be called by the parser to handle the event
  - Many developers find this model awkward to use
- Simple API for XML (SAX) is an example of a push parser

9

Streaming API for XML

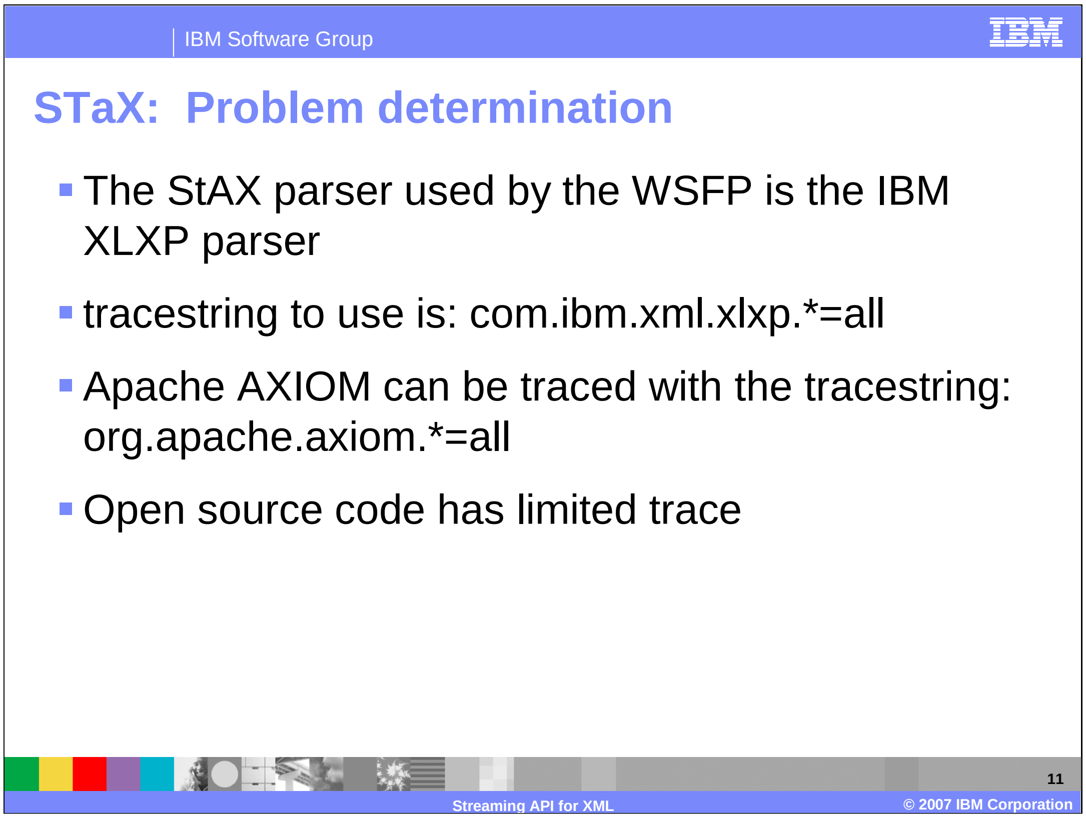
In push parsing, the parsing library calls methods in the application code. That is the parser “pushes” events to the application and the application must have a handler that is ready to be called for these events.   Many developers tend to find this model awkward to use. The Simple API for XML (SAX) is a well known example of a push parser.



Section

Problem determination

The next section will explain problem determination for the StAX technologies.



STaX: Problem determination

- The StAX parser used by the WSFP is the IBM XLXP parser
- tracestring to use is: com.ibm.xml.xlxp.*=all
- Apache AXIOM can be traced with the tracestring: org.apache.axiom.*=all
- Open source code has limited trace

The StAX based parser used by the Feature Pack for Web Services is called the XLSP parser. Tracing can be done using the tracestring: com.ibm.xml.xlxp.*=all.

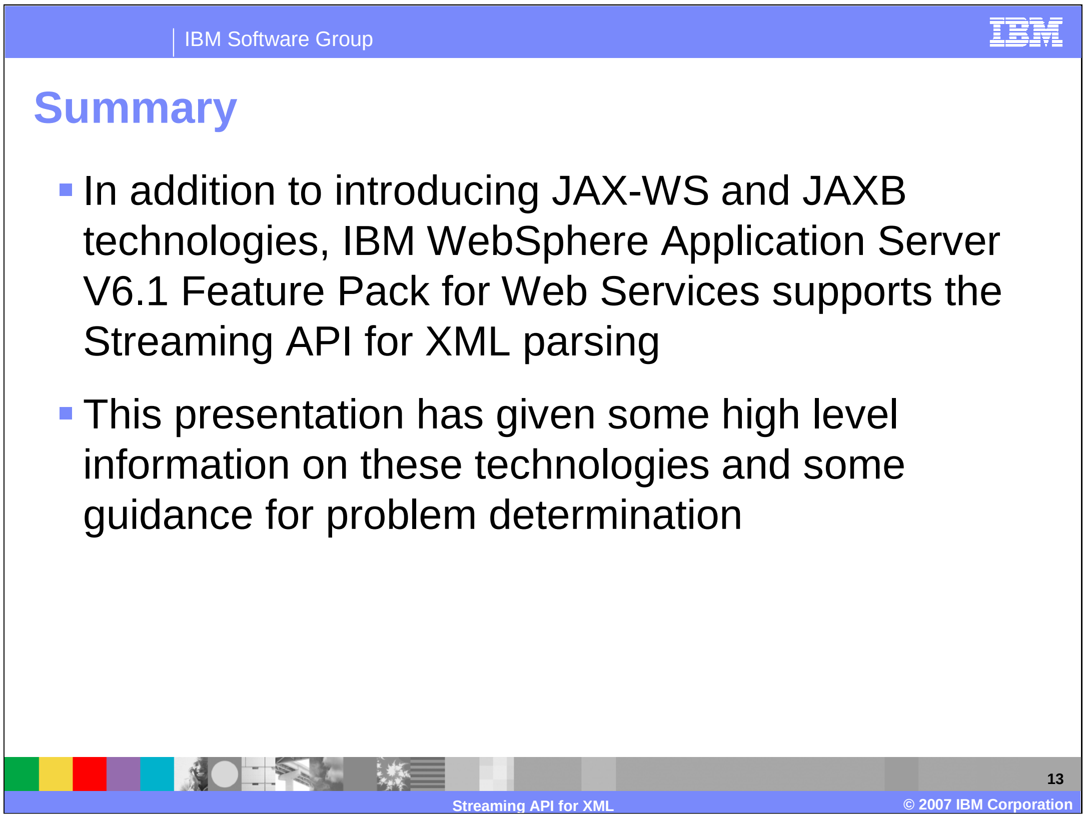
Some parts of IBM WSFP are based on Apache AXIOM code and this can be traced using a tracestring of org.apache.axiom.*=all.

Users should be aware that much of the open source code has limited trace capability, and will return less trace information.

Section

Summary

Now for a summary of the information from this presentation.



IBM Software Group

Summary

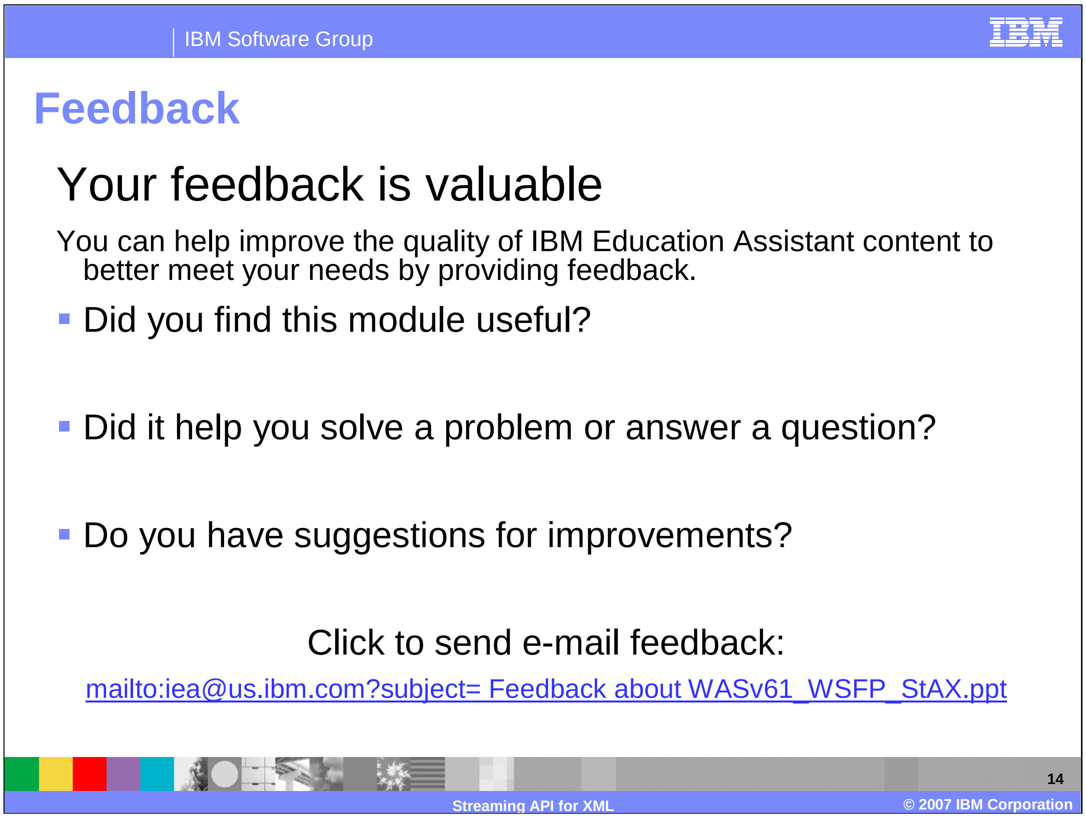
- In addition to introducing JAX-WS and JAXB technologies, IBM WebSphere Application Server V6.1 Feature Pack for Web Services supports the Streaming API for XML parsing
- This presentation has given some high level information on these technologies and some guidance for problem determination

13

Streaming API for XML

© 2007 IBM Corporation

In addition to introducing JAX-WS and JAXB technologies, IBM WebSphere Application Server V6.1 Feature Pack for Web Services supports the Streaming API for XML parsing. A new parser technology that is both easier to develop for and has a better performance in many situations. This presentation has given some high level information on these technologies and some guidance for problem determination related to the StAX technologies.



Feedback

Your feedback is valuable

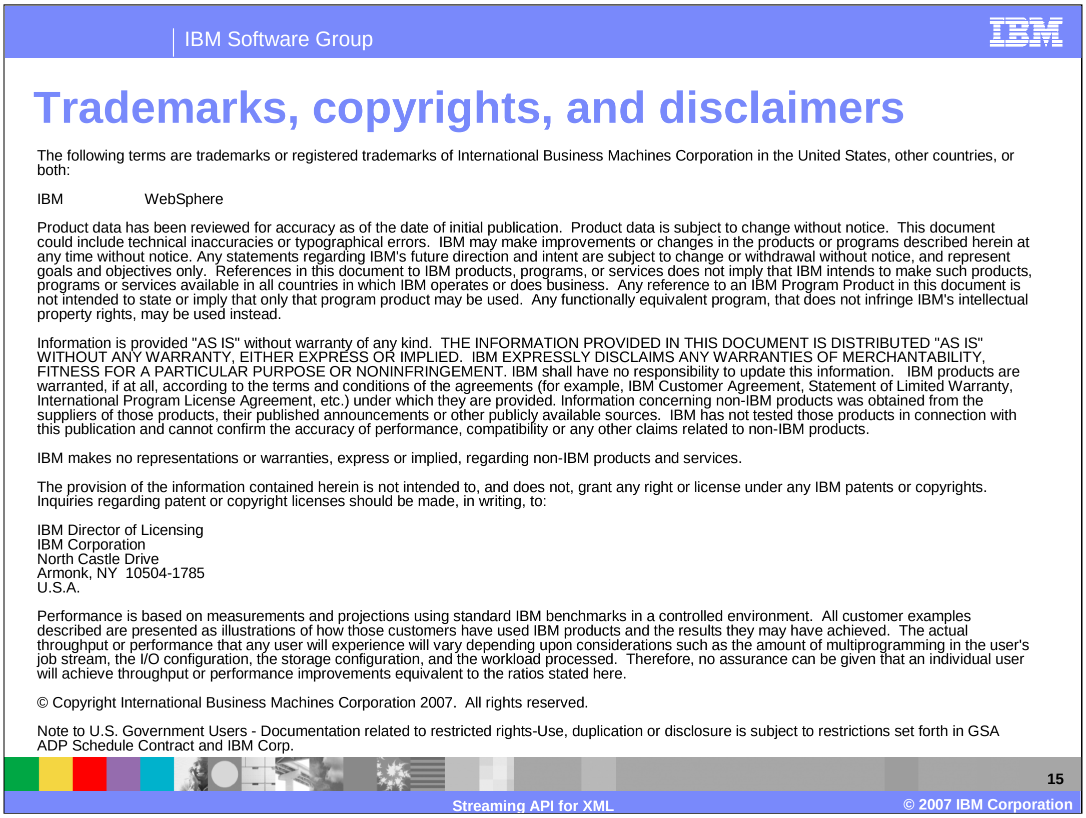
You can help improve the quality of IBM Education Assistant content to better meet your needs by providing feedback.

- Did you find this module useful?
- Did it help you solve a problem or answer a question?
- Do you have suggestions for improvements?

Click to send e-mail feedback:

mailto:iea@us.ibm.com?subject= Feedback about WASv61_WSFP_StAX.ppt

You can help improve the quality of IBM Education Assistant content by providing feedback.



Trademarks, copyrights, and disclaimers

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States, other countries, or both:

IBM	WebSphere

Product data has been reviewed for accuracy as of the date of initial publication. Product data is subject to change without notice. This document could include technical inaccuracies or typographical errors. IBM may make improvements or changes in the products or programs described herein at any time without notice. Any statements regarding IBM's future direction and intent are subject to change or withdrawal without notice, and represent goals and objectives only. References in this document to IBM products, programs, or services does not imply that IBM intends to make such products, programs or services available in all countries in which IBM operates or does business. Any reference to an IBM Program Product in this document is not intended to state or imply that only that program product may be used. Any functionally equivalent program, that does not infringe IBM's intellectual property rights, may be used instead.

Information is provided "AS IS" without warranty of any kind. THE INFORMATION PROVIDED IN THIS DOCUMENT IS DISTRIBUTED "AS IS" WITHOUT ANY WARRANTY, EITHER EXPRESS OR IMPLIED. IBM EXPRESSLY DISCLAIMS ANY WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NONINFRINGEMENT. IBM shall have no responsibility to update this information. IBM products are warranted, if at all, according to the terms and conditions of the agreements (for example, IBM Customer Agreement, Statement of Limited Warranty, International Program License Agreement, etc.) under which they are provided. Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products in connection with this publication and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products.

IBM makes no representations or warranties, express or implied, regarding non-IBM products and services.

The provision of the information contained herein is not intended to, and does not, grant any right or license under any IBM patents or copyrights. Inquiries regarding patent or copyright licenses should be made, in writing, to:

IBM Director of Licensing
IBM Corporation
North Castle Drive
Armonk, NY 10504-1785
U.S.A.

Performance is based on measurements and projections using standard IBM benchmarks in a controlled environment. All customer examples described are presented as illustrations of how those customers have used IBM products and the results they may have achieved. The actual throughput or performance that any user will experience will vary depending upon considerations such as the amount of multiprogramming in the user's job stream, the I/O configuration, the storage configuration, and the workload processed. Therefore, no assurance can be given that an individual user will achieve throughput or performance improvements equivalent to the ratios stated here.

Note to U.S. Government Users - Documentation related to restricted rights-Use, duplication or disclosure is subject to restrictions set forth in GSA ADP Schedule Contract and IBM Corp.