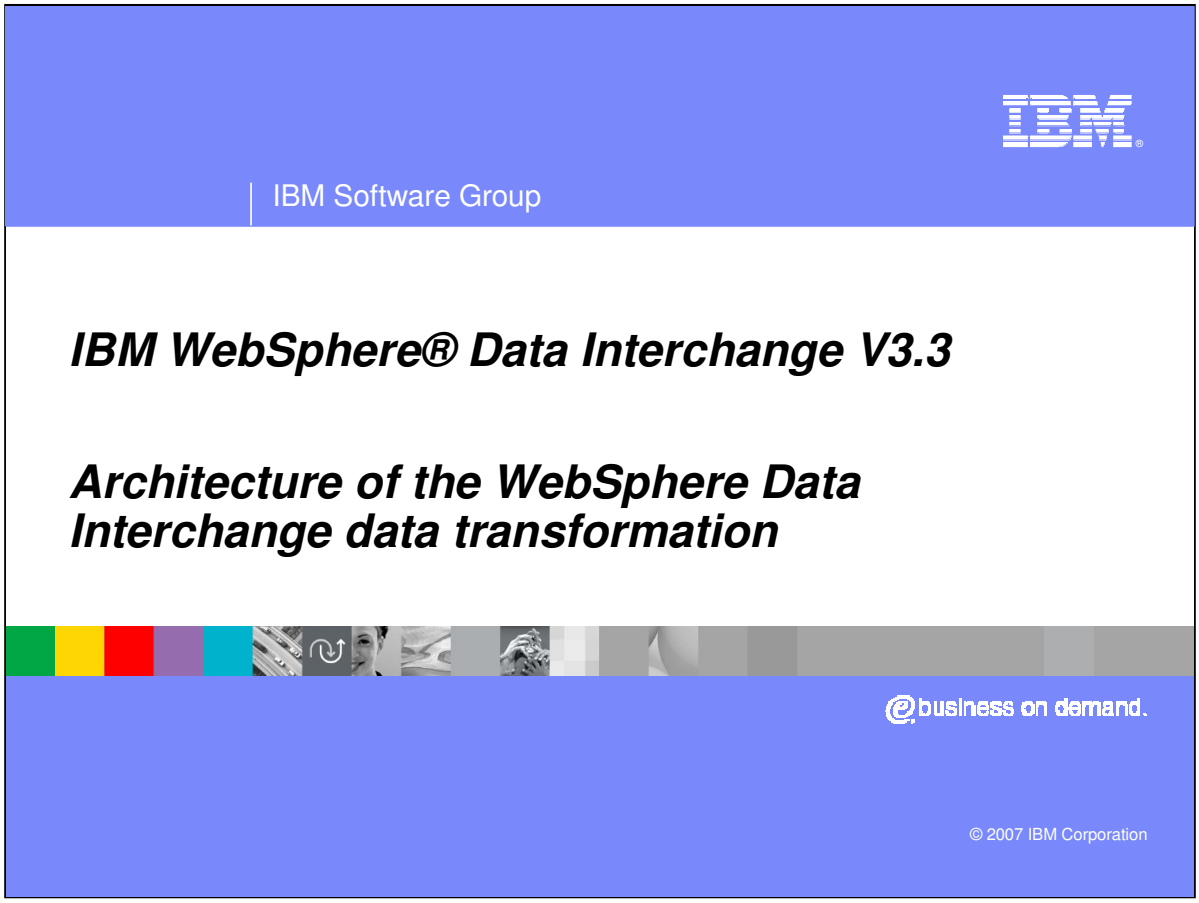
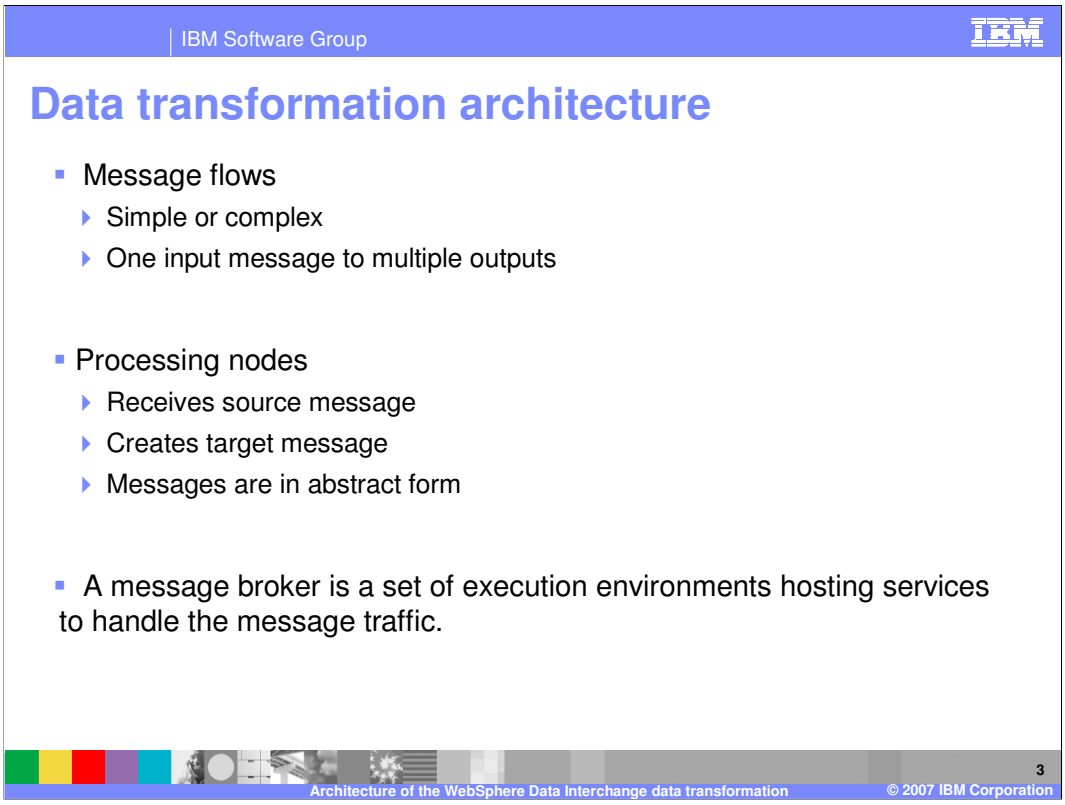


This presentation will describe the WebSphere Data Interchange Data Transformation architecture.

Agenda

- Describe message flow
- Describe WebSphere Data Interchange message flow
- Identity data transformation components
- WebSphere Data Interchange parsers and serialization
- Describe delayed enveloping flow
- Describe delayed translation flow
- Document store and optional records

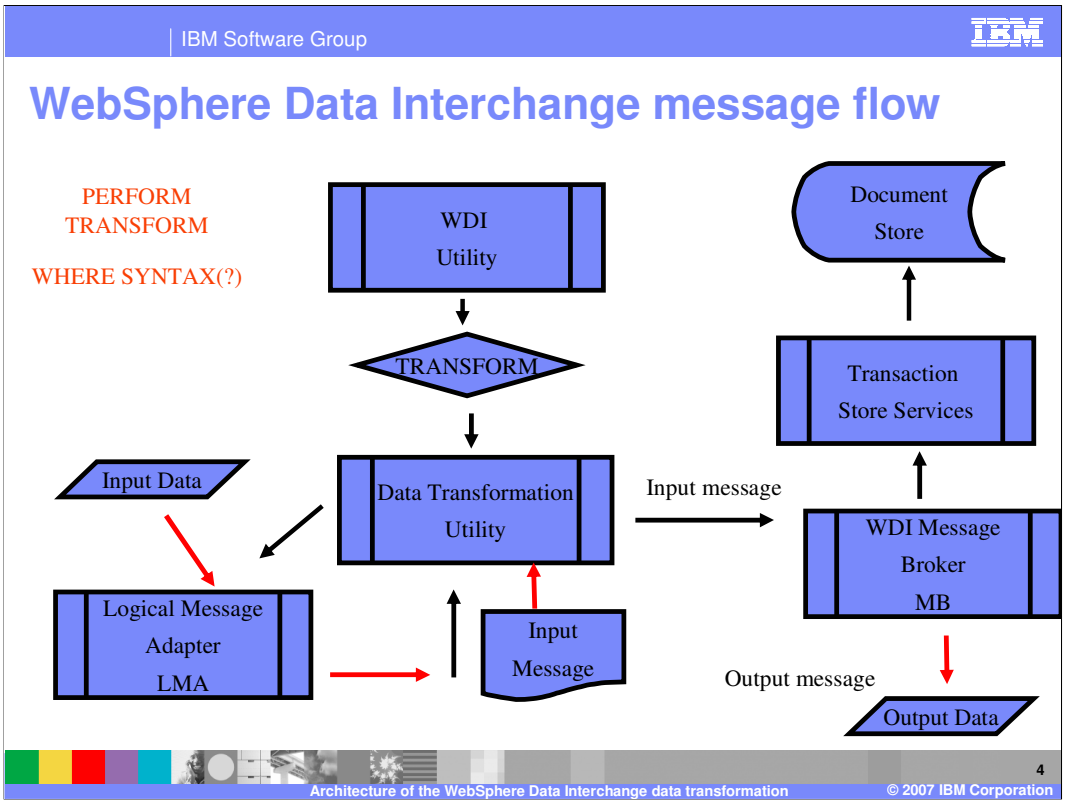
The presentation will describe message flow, the WebSphere Data Interchange message flow, identify data transformation components, and review processing options.



Message flows can range from the very simple, performing just one action on a message, to the complex, providing a number of actions on the message to transform its format and content. A message flow can process one message in several ways to deliver a number of output messages, perhaps with different format and content, to a number of target applications.

The other nodes between input and output provide the actions you want taken against the messages. Each node receives the source message and creates a target message. The source and target messages are represented in an abstract form using an Abstract Message Model (AMM).

A message broker is a set of execution environments hosting services to handle the message traffic.



The WebSphere Data Interchange Utility parses the PERFORM command and determines what process to call. The Utility component ID is FF and it logs messages beginning with FF.

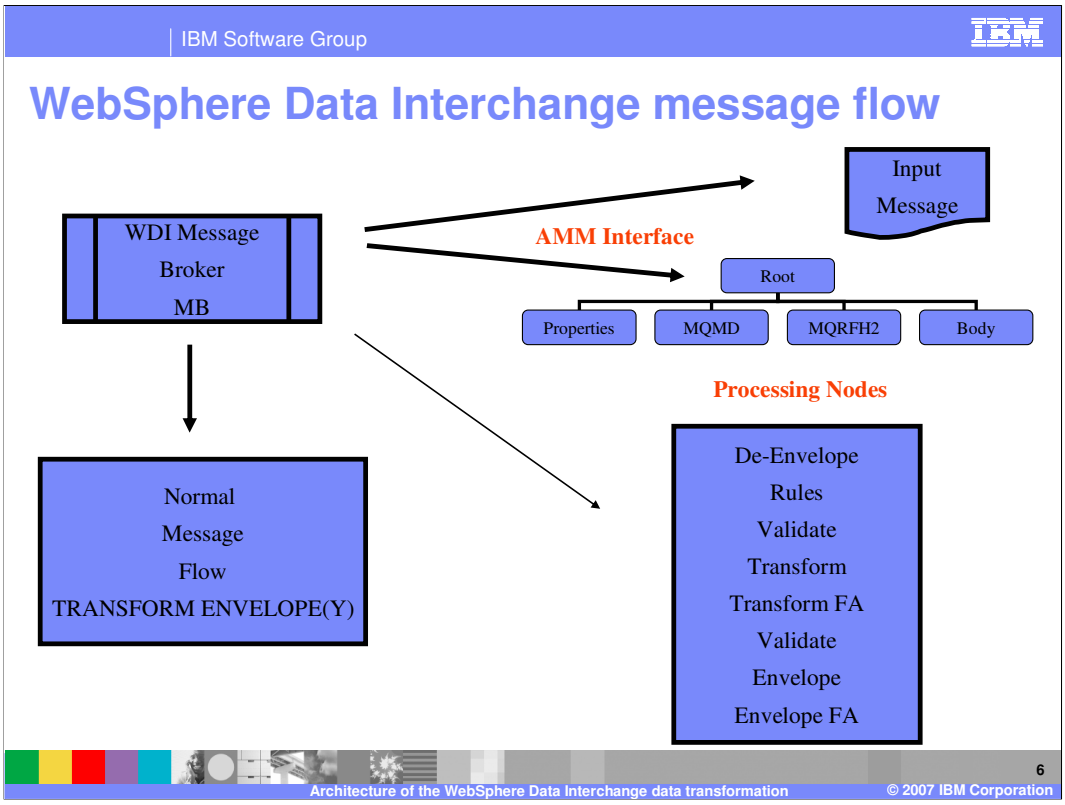
With PERFORM TRANSFORM command, the Data Transformation (DT) Utility is called to parse the input data into messages based on syntax. The Data Transformation Utility calls the Logical Message Adapter (LMA) to read the input data and parse out 1 logical input message. It does setup for the Message Broker (MB) including PERFORM keyword options, Document Store active. And passes EACH logical message to the Message Broker for processing.

The Message Broker initializes the message flow for processing and creates and initializes the source Logical Message and the source document properties in the Abstract Message (AMM). The Message Broker also, causes the Target Abstract Message (output message) to be serialized to output and calls Transaction Store Services to make the Document Store updates. The Message Broker component ID is MB and it logs messages beginning with MB.

WebSphere Data Interchange message flow

- Data transformation processing
  - The logical message adapter reads the input data and parses out 1 logical message based on SYNTAX() keyword
  - *EDI* source SYNTAX(E). A logical message is 1 interchange
  - *Data Format* or DF source SYNTAX(D)
    - Uses the DF metadata (control string) to identify the beginning or end of a logical message
    - Also returns partner identification fields
  - *XML* source SYNTAX(X). Uses the "<?xml" in the input file to identify a logical message

The SYNTAX keyword is required with the PERFORM TRANSFORM command. Syntax E is EDI source and a logical message is one interchange. Syntax D is application data or DF source and a logical message is identified in the metadata definition. Syntax X is XML source and a logical message is identified by the beginning of the XML prologue.

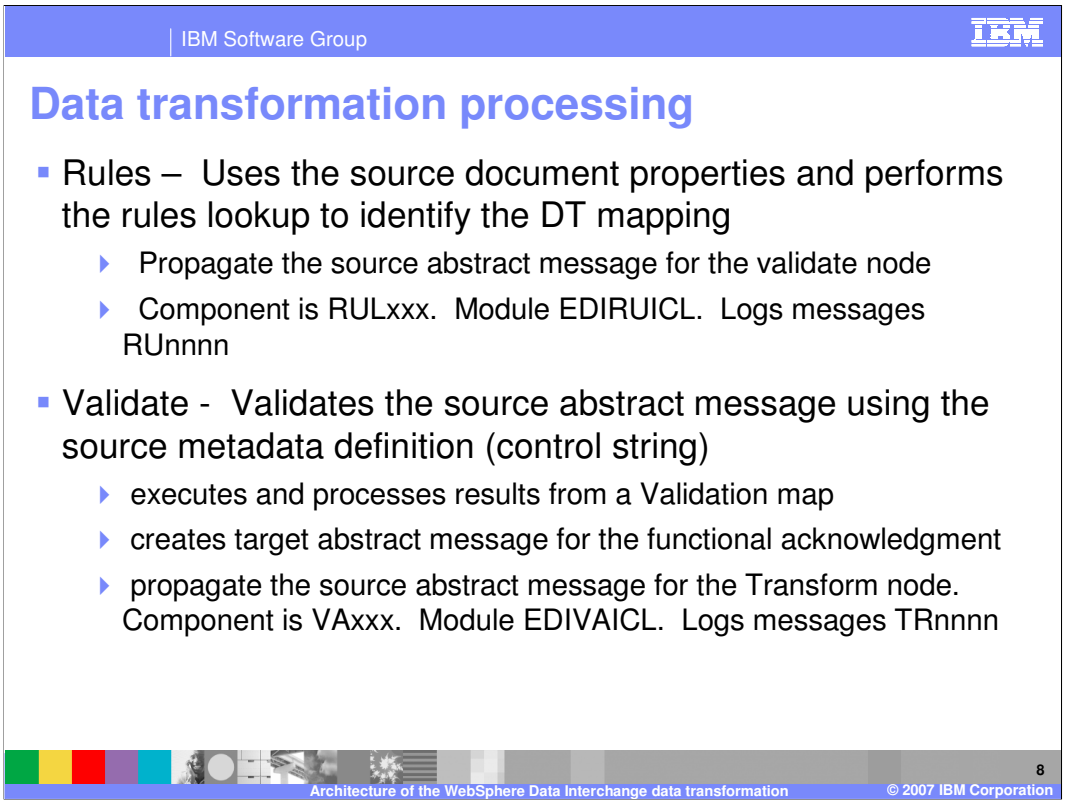


A Normal Message Flow is the message flow for a translate and envelope process as opposed to delayed enveloping processing. It defines the processing NODES for the message. All processing nodes create and update information for the Document Store and optional record processing. Each node also has a source abstract message and propagate a target abstract message for the next processing node.

Data transformation processing

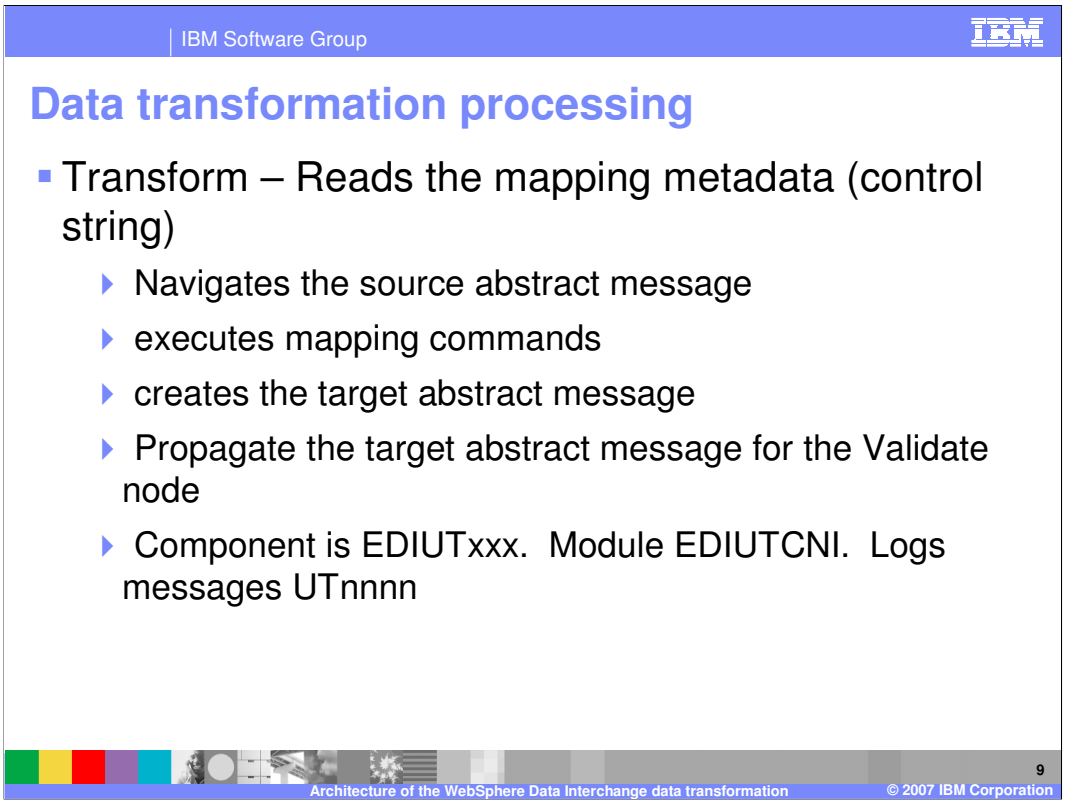
- De-envelope – Uses input message syntax and assigns appropriate de-enveloper
  - Message navigation causes the input message to be parsed into the Source Abstract Message
  - Navigates the source AMM and sets the Source document properties in the AMM Properties folder
  - Propagate the Source AMM for the Rules node Component is EVxxx. Logs messages EVnnnn
    - Data Format (DF) – module EDIEVADF
    - XML – module EDIEVXML
    - EDI (Support for X12, EDIFACT, UNTDI, and UCS).. Modules EDIEVX12, EDIEVFAC, EDIEVTDI, EDIEVUCS)

The De-Envelope node uses the input message syntax and assigns the appropriate de-enveloper. Abstract message navigation causes the input message to be parsed into the Source Abstract Message (AMM). The deenveloper navigates the source abstract message and sets the Source document properties in the Abstract Message Properties folder and propagates the Source Abstract Message for the Rules node. The De-Envelope node component is EV and it logs messages beginning with EV.

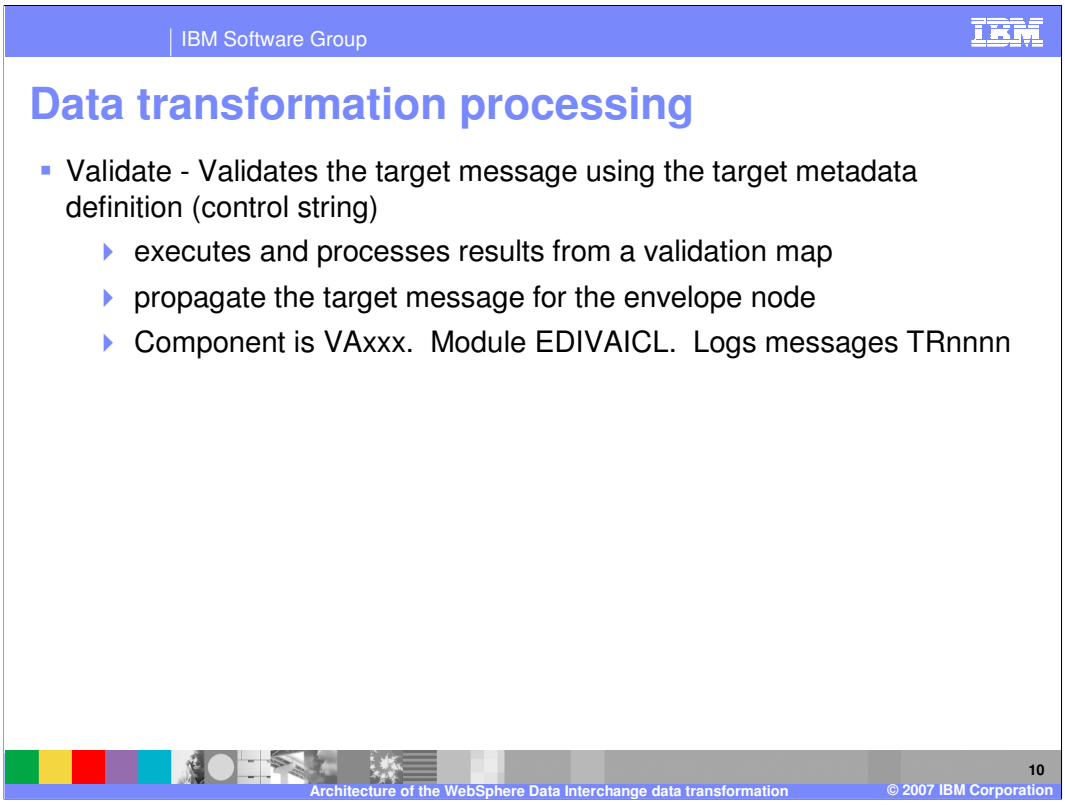


The Rules node uses the Source document properties and performs the Data Transformation (DT) Rules lookup to identify the Data Transformation mapping to be used and any Rule information for the Target message. The Rules node propagates the Source Abstract Message for the Validate node. The Rules component is RU, the module name is EDIRUICL, and it logs messages beginning with RU.

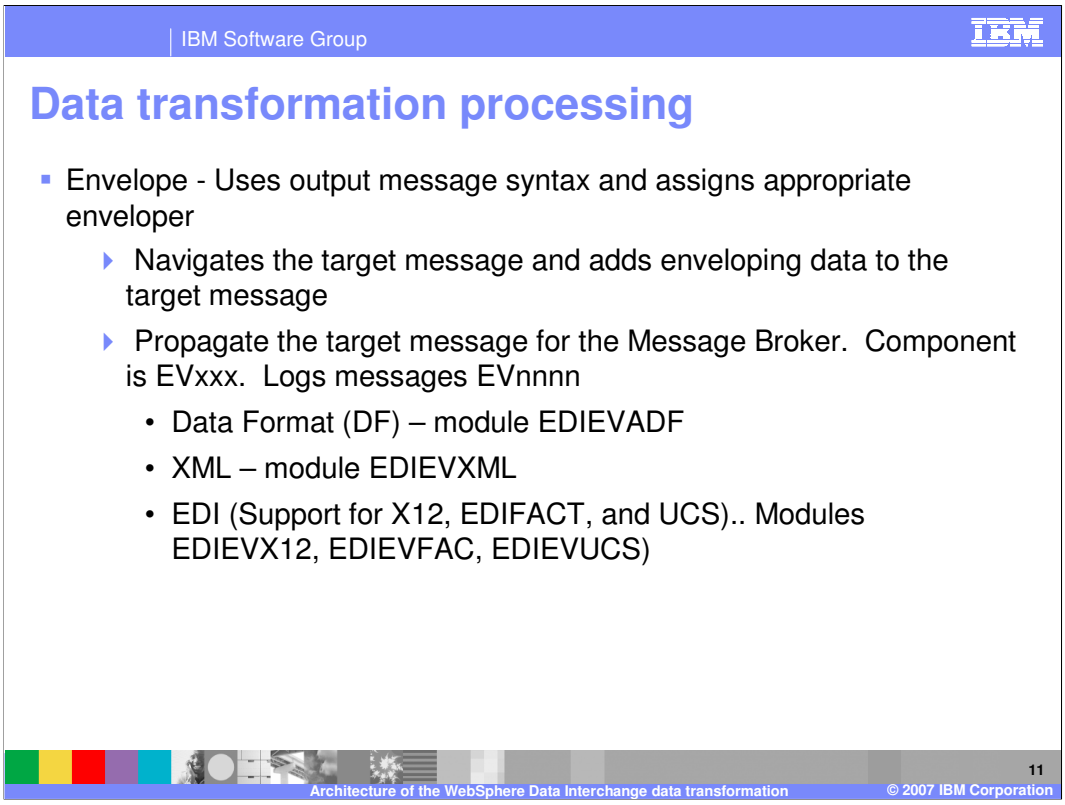
The Validate node validates the Source Abstract Message using the source metadata definition, executes and processes results from a Validation map, creates the target Abstract Message for functional acknowledgment processing, and propagates the Source Abstract Message for the Transform node. If functional acknowledgment processing is identified, the target Abstract Message for the acknowledgment is propagated to the Transform node to execute the mapping commands for the functional acknowledgment map. The Validate node component is VA, the module name is EDIVAICL, and it logs messages beginning with TR.



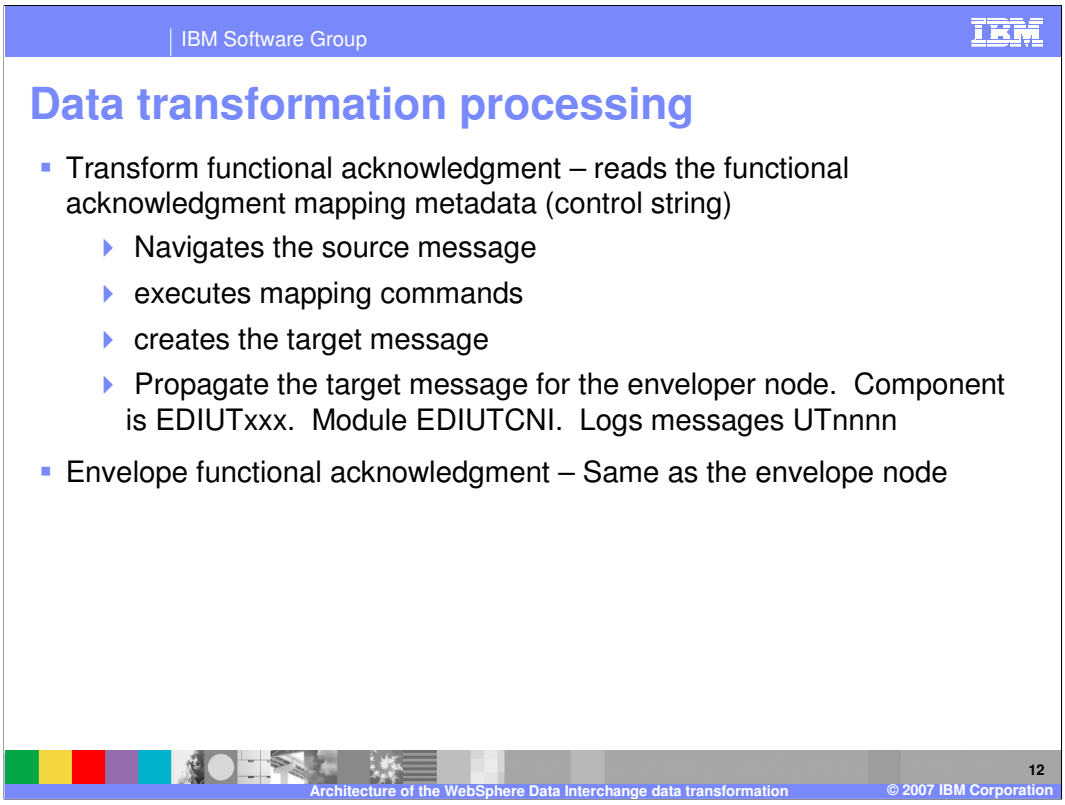
The Transform node reads the Data Transformation mapping metadata or mapping control string, navigates the Source Abstract Message, executes mapping commands, creates the Target Abstract Message and propagates the Target Abstract Message for the Validate node. The Transform component is UT, the module name is EDIUTCNI, and it logs messages beginning with UT.



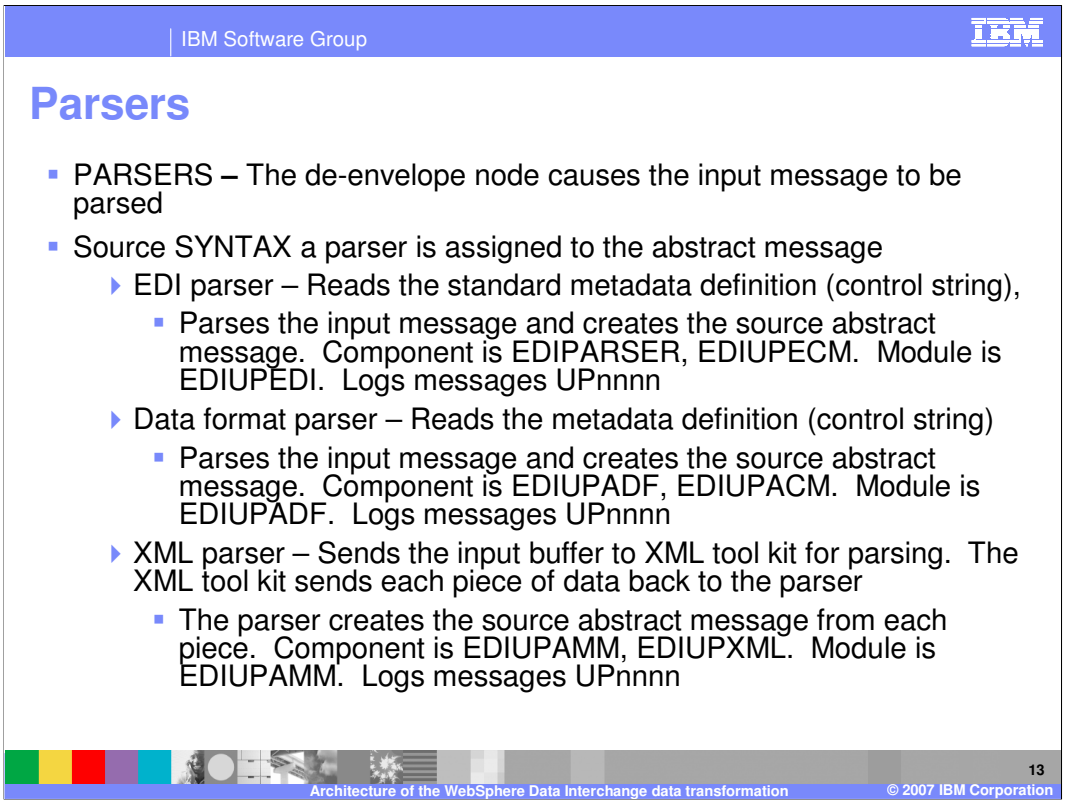
The Validate node validates the Target Abstract Message using the target metadata definition, executes and processes results from a Validation map, and propagates the Target Abstract Message for the Envelope node.



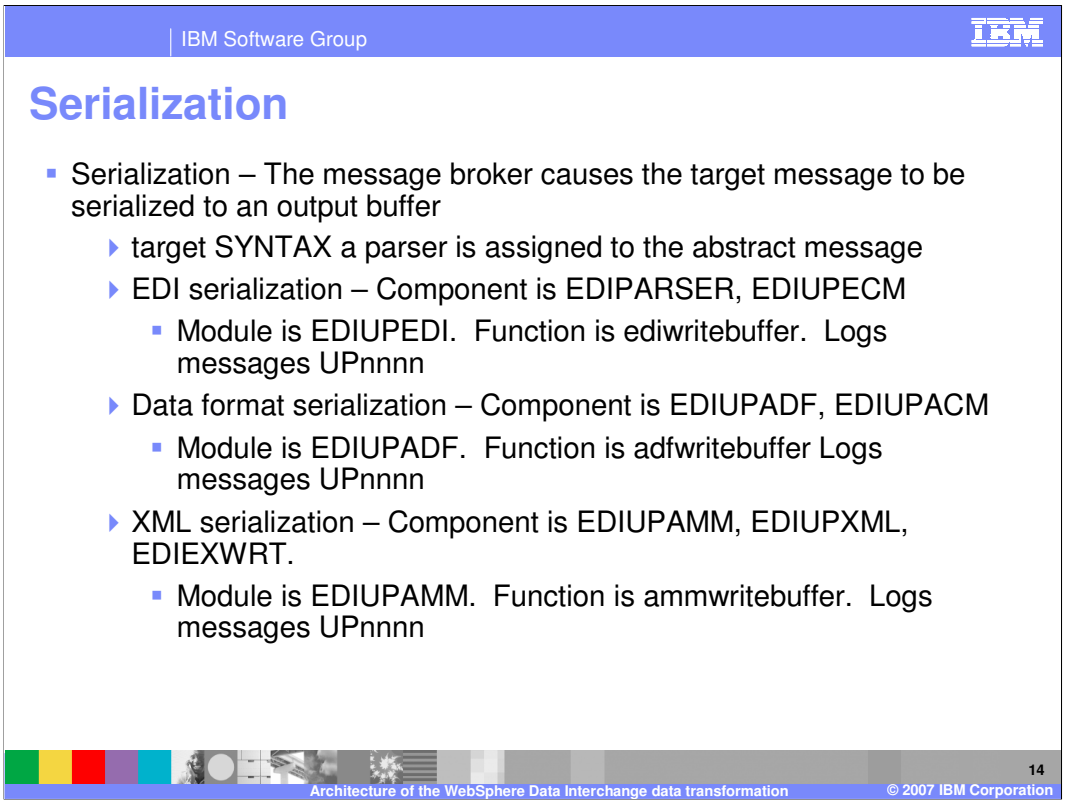
The Envelope node uses output message syntax and assigns the appropriate enveloper. The Envelope node navigates the Target Abstract Message, adds enveloping data, and propagates the Target Abstract Message for the Message Broker (MB). The Envelope node component is EV and it logs messages beginning with EV.



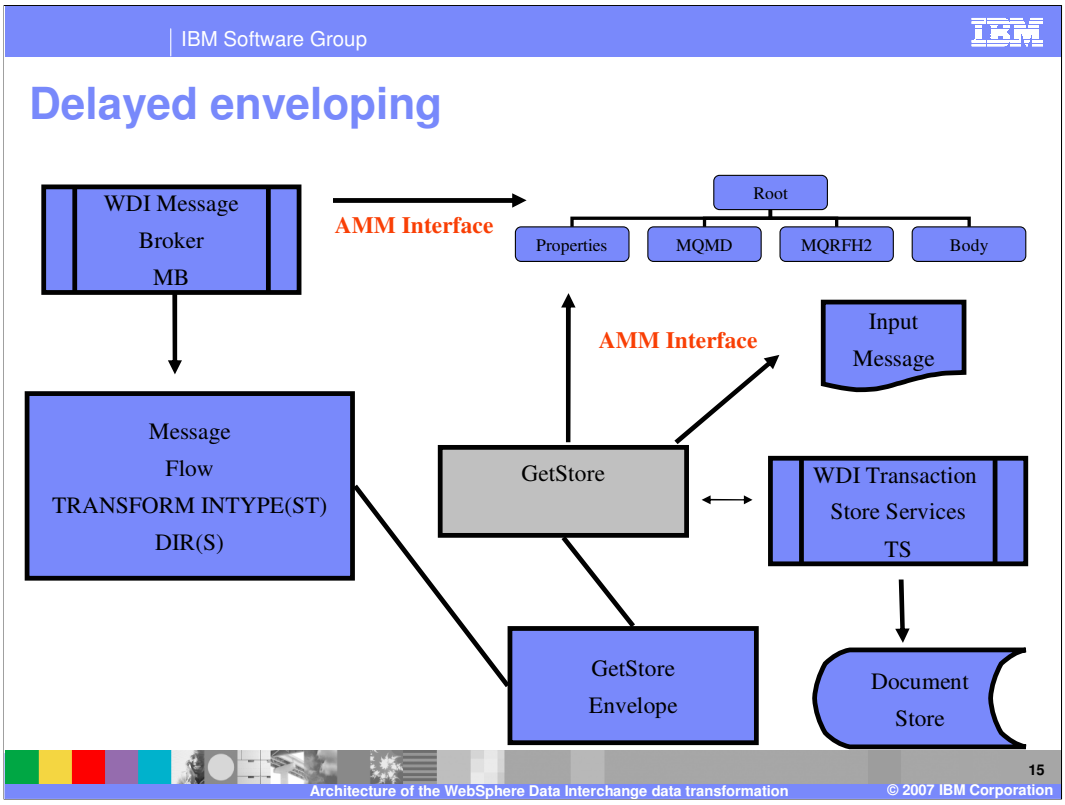
The Transform Functional Acknowledgment is the Transform node. It reads the Data Transformation Functional Acknowledgment mapping metadata or control string, navigates the Source Abstract Message, executes mapping commands, and creates the Target Abstract Message. The Transform node propagates the Target Abstract Message for the Enveloper node to envelope the Functional Acknowledgment.



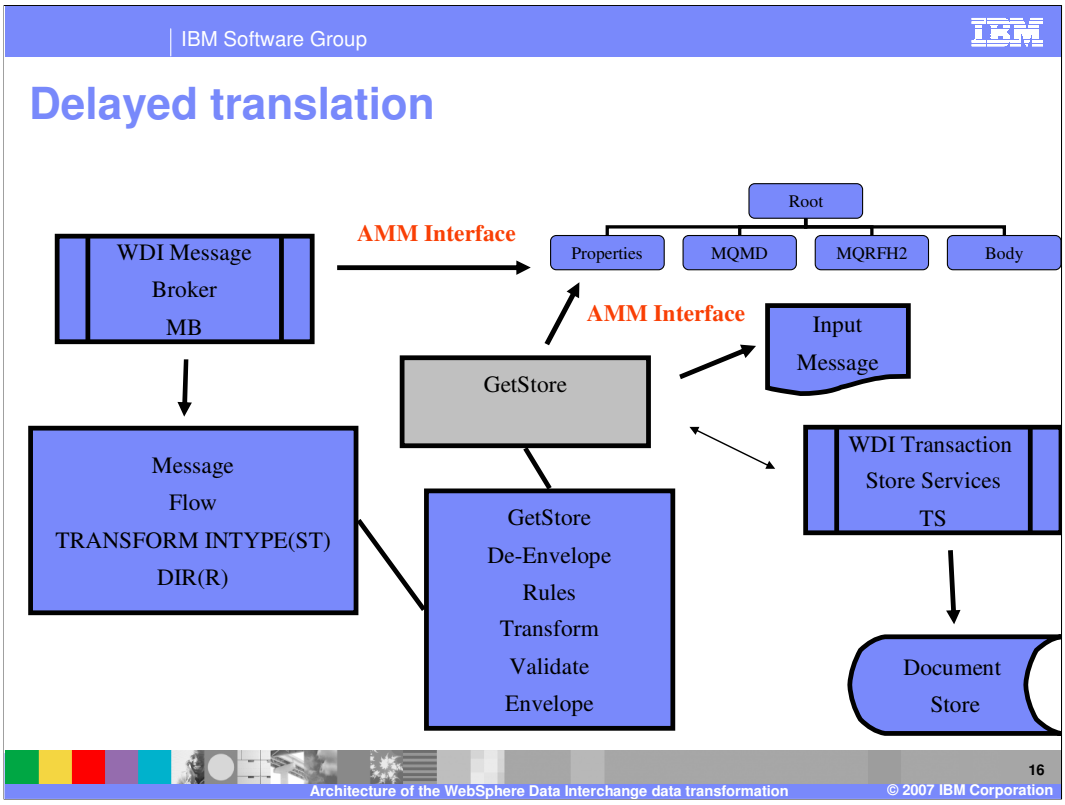
Parsers are used to parse the source data. All parsers use the WebSphere Data Interchange abstract message interface and create the source abstract message. The De-envelope node causes the input message to be parsed. Based on the source SYNTAX a parser is assigned to the abstract message (AMM). All parsers use the Abstract Message interface and create the source abstract message. The Parsers component is UP and they log messages beginning with UP.



The Message Broker causes the Target abstract message to be serialized to an output buffer. All parsers contain some kind of writebuffer function and use the WebSphere Data Interchange abstract message interface to navigate the target abstract message. Based on the target SYNTAX a parser is assigned to the abstract message (AMM).



With delayed enveloping, the Data Transformation Utility gathers information about the messages to be enveloped based on selection criteria and passes the information to the Message Broker. The Message Broker initializes the message flow with two processing nodes, GetStore and Envelope. The GetStore node calls Transaction Store services to gather the messages to be enveloped. It creates the input message and the Abstract Message Properties for the delayed enveloping. The Abstract Message is propagated for the Envelope Node and the message flow continues along the normal message flow.



With delayed translation, the Data Transformation Utility gathers information about the messages to be translated or transformed based on selection criteria and passes the information to the Message Broker. The Message Broker initializes the message flow with the GetStore, De-Envelope, Transform, Validate and Envelope processing nodes. The GetStore node calls Transaction Store services to gather the messages to be translated. It creates the input message and the Abstract Message Properties for the delayed translation. The Abstract Message is propagated for the De-Envelope Node and the message flow continues along the normal message flow.

Document store and optional records

- **Document store –** The processing nodes that create the information stored in document store
  - interface to create the information
  - passed back to the message broker. Component is TSUPD, TSUTL. Module is EDIDTUTL. Logs messages TSnnnn
- **Optional records** – The processing nodes that create the information needed to produce optional records
  - interface to create the information
  - passed back to the message broker. Component is EDIOPUPD, EDIOPUTL. Module is EDIDTUTL

Any processing node that creates information that should be stored in the Document Store (DS) have an interface to create the information. When complete, an internal representation of the Document Store records are created in a linked list to pass back to the Message Broker. The Message Broker uses Transaction Store Services to update the Document Store. Component is TSUPD, TSUTL. Module is EDIDTUTL. Logs messages beginning with TS.

Template Revision: 04/25/2006 11:09 AM

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	CICS	IMS	WebSphere MQ	Tivoli
IBM(logo)	Cloudscape	Informix	OS/390	WebSphere
e(logo)business	DB2	iSeries	OS/400	xSeries
AIX	DB2 Universal Database	Lotus	pSeries	zSeries

Java and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Microsoft, Windows, Windows NT, and the Windows logo are registered trademarks of Microsoft Corporation in the United States, other countries, or both.

Intel, ActionMedia, LANDesk, MMX, Pentium and ProShare are trademarks of Intel Corporation in the United States, other countries, or both.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Linux is a registered trademark of Linus Torvalds.

Other company, product and service names may be trademarks or service marks of others.

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 and/or changes in the product(s) and/or program(s) 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 (e.g., 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.