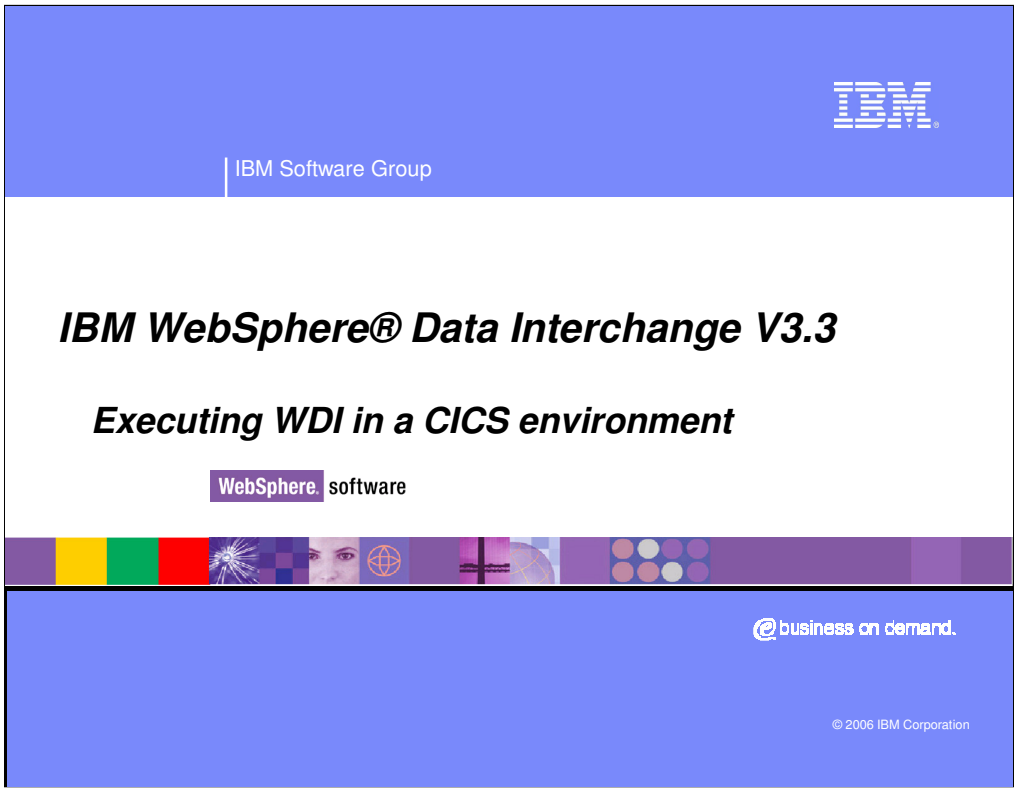
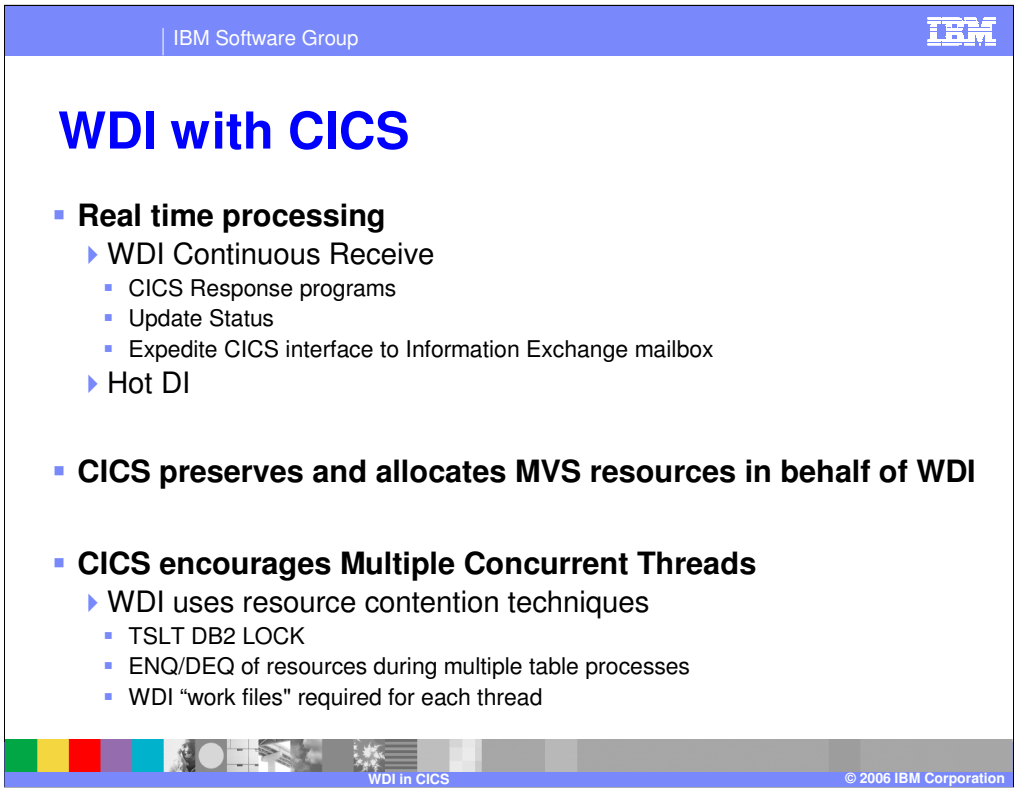


This presentation discusses how to execute WDI in a CICS environment.



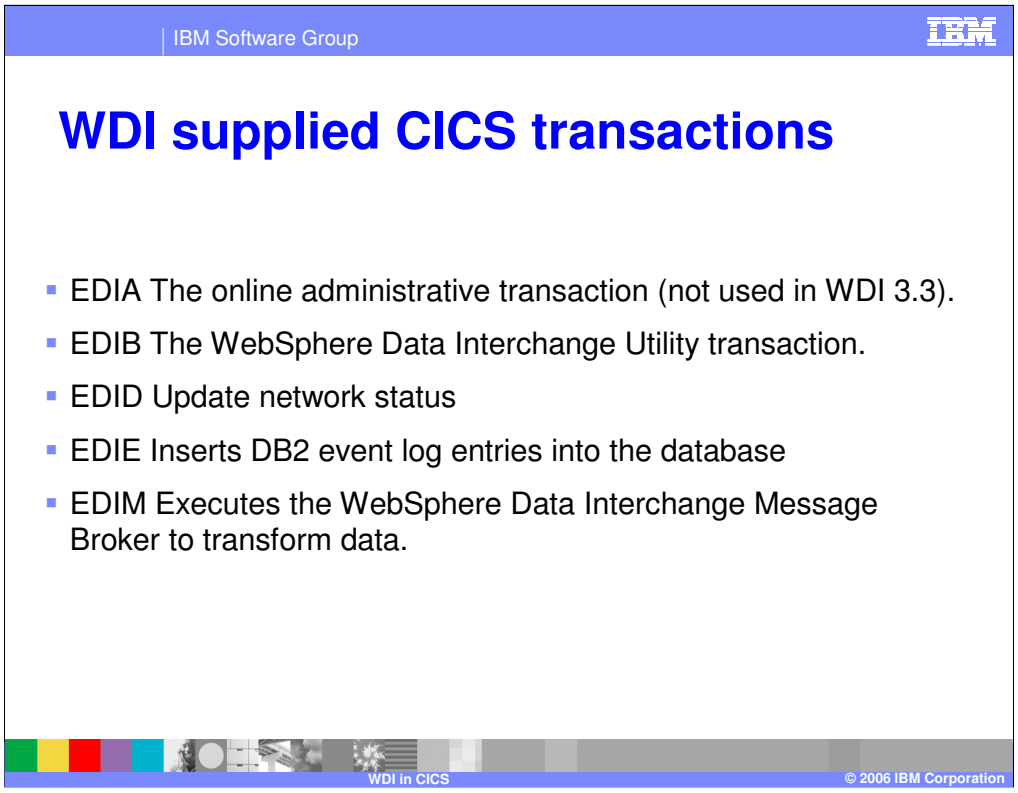
CICS is a real-time Transaction processing system. To maintain performance, CICS “mimics” operating system functions to reduce wait time. It is also designed to handle large volumes by providing multiple processing threads for transactions. WebSphere Data Interchange created two techniques to work with CICS design points: WDI Continuous Receive and WDI Hot DI. Continuous Receive takes advantage of the VAN capability of CICS and the Expedite product. Hot DI is a performance technique using CICS TS Queue capability for “remembering” status. CICS and WDI use main storage to replace slower DASD accesses for work files and temporary data storage. WDI also uses DB2 LOCK commands and CICS ENQ and DEQ techniques to “single thread” parts of processes.

WDI with CICS

- **Support Functions**
  - EDIW - Interactive Utility Interface
  - TSQE - Temporary Storage Queue Editor
- **Debugging uses CEDF**

A utility program named EDIW is available to support user-initiated, transaction-oriented requests. Another CICS transaction, TSQE, allows for the getting and putting of data for use with EDIW.

CEDF can be used to to trace thru WDI translations.



WebSphere Data Interchange supplied CICS transactions

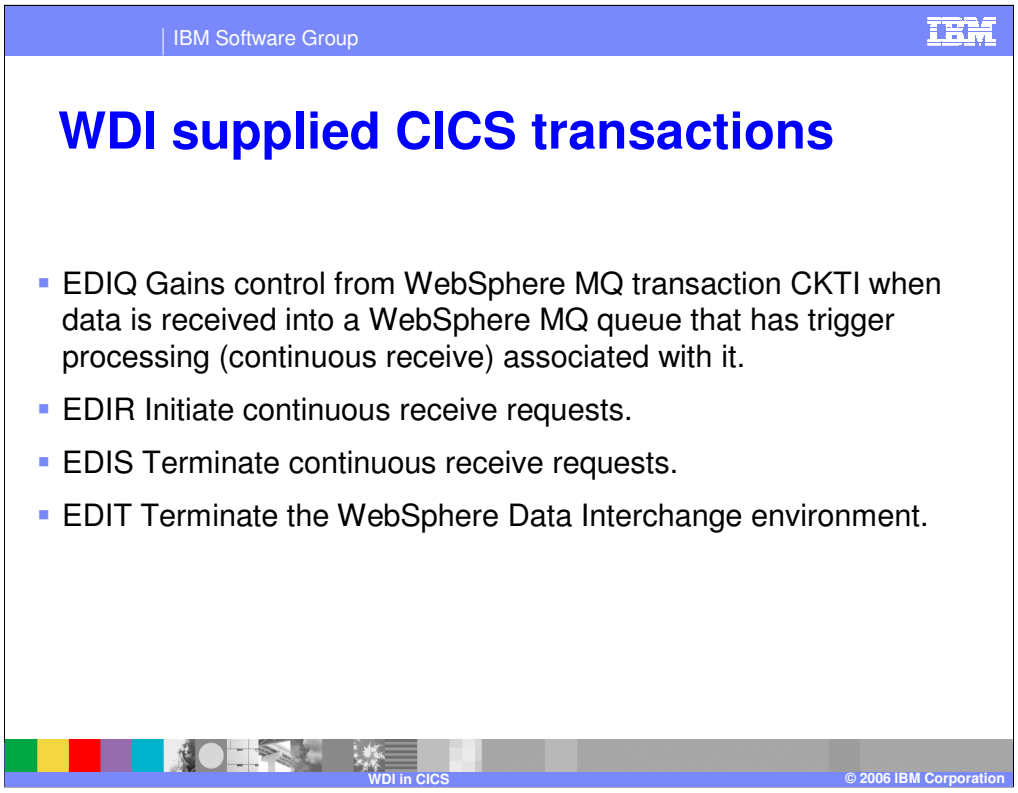
EDIA The online administrative transaction. Use this transaction to customize in the CICS environment. **Note:** To prevent processing from being halted while EDIX waits to proceed, make sure to establish a DB2 thread pool that is used by only EDIA.

EDIB The WebSphere Data Interchange Utility transaction. If you run the WebSphere Data Interchange Utility asynchronously, use EXEC CICS START to start this transaction. You can also start EDIB with WebSphere Data Interchange as part of continuous receive processing. **Note:** To prevent processing from being halted while EDIX waits to proceed, make sure to establish at least three DB2 threads that is used only by EDIB. The maximum number of threads needed for this transaction is the sum of all TRANClass values for the transactions in EDIB. A typical value is five threads.

EDID Used internally by WebSphere Data Interchange. A background transaction used to update network status. EDID does not have a user interface.

EDIE Used internally by WebSphere Data Interchange. A background transaction used to insert DB2 event log entries into the database (see TDQs EDI1, EDI2, and EDI3). EDIE keeps DB2 event log insertions out of the main commit scope. EDIE does not have a user interface. **Note:** To prevent processing from being halted while EDIX waits to proceed, make sure to establish a DB2 thread pool that is used only EDIX and EDIE. Two or more threads must be available for this pool.

EDIM A transaction that executes the WebSphere Data Interchange Message Broker to transform data.



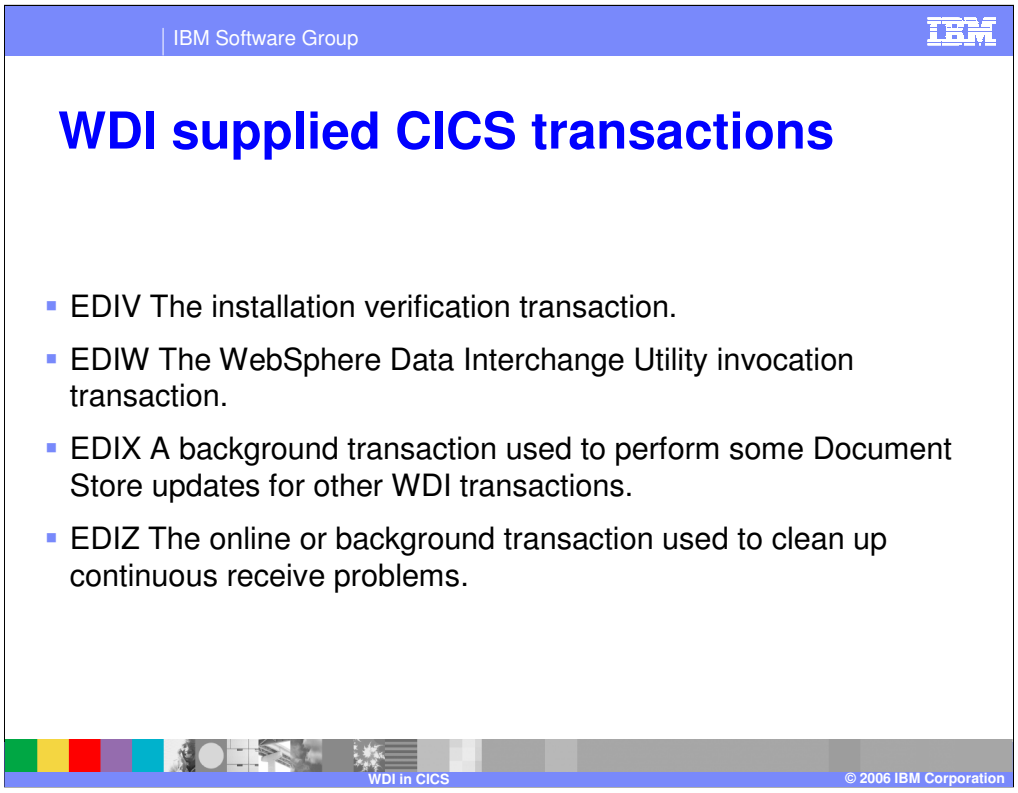
WebSphere Data Interchange supplied CICS transactions

EDIQ The transaction that gains control from WebSphere MQ transaction CKTI when data is received into a WebSphere MQ queue that has trigger processing (continuous receive) associated with it.

EDIR The online or background transaction used to initiate continuous receive requests.

EDIS The online or background transaction used to terminate continuous receive requests.

EDIT A transaction that can be used to terminate the WebSphere Data Interchange environment. The WebSphere Data Interchange environment is usually terminated by placing EDIXSOX in the CICS pre-termination PLT, but you can also terminate the WebSphere Data Interchange environment by running EDIT. Certain pieces of information are maintained from the start of the first WebSphere Data Interchange activity within a CICS region, continuing throughout the session, until the WebSphere Data Interchange environment is terminated. You can use EDIT to reset this information. EDIT shuts down EDIX, releases CSD storage, and deletes the EDICSDA and EDITV00 TS queues. Using EDIT to perform these functions is not part of standard procedures.



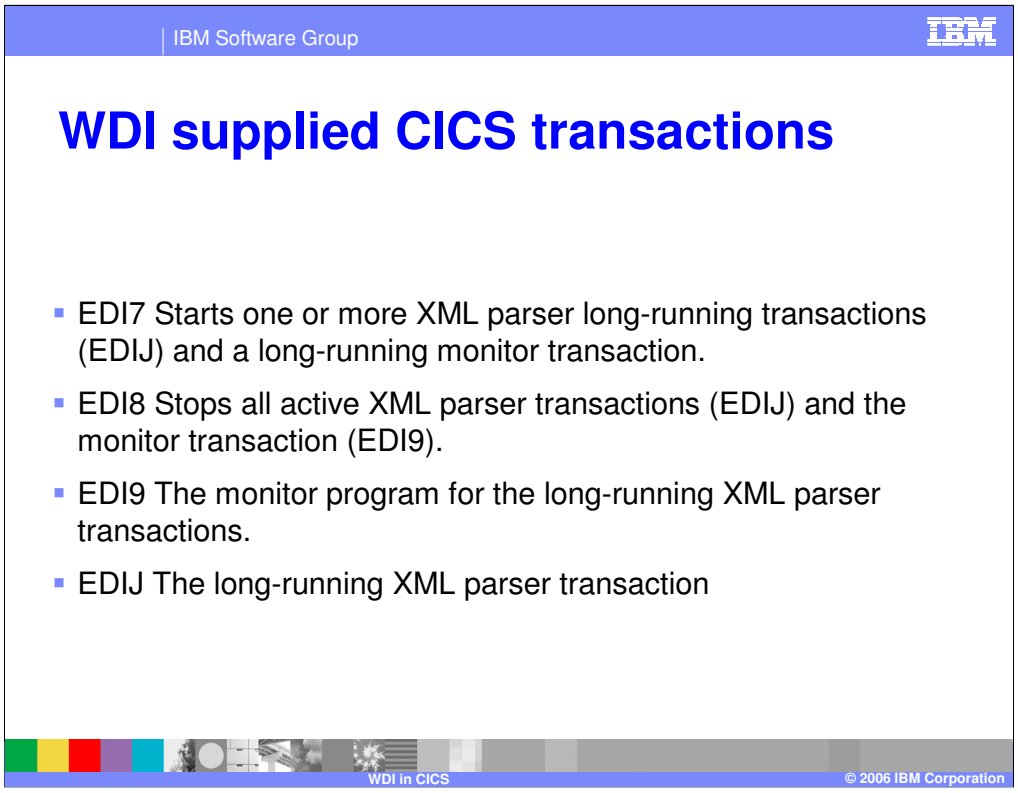
WebSphere Data Interchange supplied CICS transactions (continued)

EDIV The installation verification transaction. For more information on this transaction, refer to the WebSphere Data Interchange for z/OS Installation Guide.

EDIW The WebSphere Data Interchange Utility invocation transaction. For more information, see “Using EDIW to invoke the WebSphere Data Interchange Utility” on page 251.

EDIX A background transaction used to perform some Document Store updates for other WebSphere Data Interchange transactions. This transaction does the following acquires a transaction handle and deletes an envelope for a DEENVELOPE command with the DUPENV(**Y**) option. You must define EDIX in the CICS Resource Control Table for DB2 installations. EDIX processes a request and remains idle in the system for up to one minute. When the minute expires, and no other request has been issued by a WebSphere Data Interchange transaction, EDIX removes itself from the system. If another request is generated in the one-minute wait period, the request is honored and the one-minute wait is reset. Eventually, as all WebSphere Data Interchange work is quiesced, EDIX removes itself from the system. If need be, EDIX can be removed from the system manually by typing **EDIX**. No parameters are necessary. **Note:** To prevent processing from being halted while EDIX waits to proceed, make sure to establish a DB2 thread pool that is used only EDIX and EDIE. Two or more threads must be available for this pool.

EDIZ The online or background transaction used to clean up continuous receive problems.



WebSphere Data Interchange supplied CICS transactions (continued)

EDI7 Program name EDIJSTRT. This starts one or more XML parser long-running transactions (EDIJ) and a long-running monitor transaction (EDI9). These transactions must be started in order to do XML processing. The number of parser transactions to be started is controlled by the parser.number property in the EDIParser.properties file.

EDI8 Program name EDIJSTOP. This stops all active XML parser transactions (EDIJ) and the monitor transaction (EDI9).

EDI9 Program name EDIJMNTR. This is a monitor program for the long-running XML parser transactions. If a parser ends abnormally, the monitor will restart it. This transaction is normally started by transaction EDI7.

EDIJ Program name EDIJPXML. This is the long-running XML parser transaction. It waits for data to be put on a queue by another WebSphere Data Interchange component, parses the data, and passes the results on another queue. One or more of these transactions are normally started by transaction EDI7.

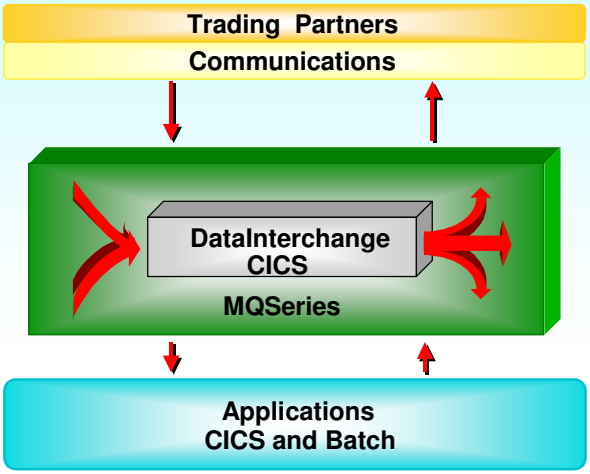
Basic Flow

DataInterchange transforms data from one format type to another

- Created to transform "flat file" data to / from EDI formats, X12, EDIFACT
- Also transforms “flat file” and EDI data to and from XML

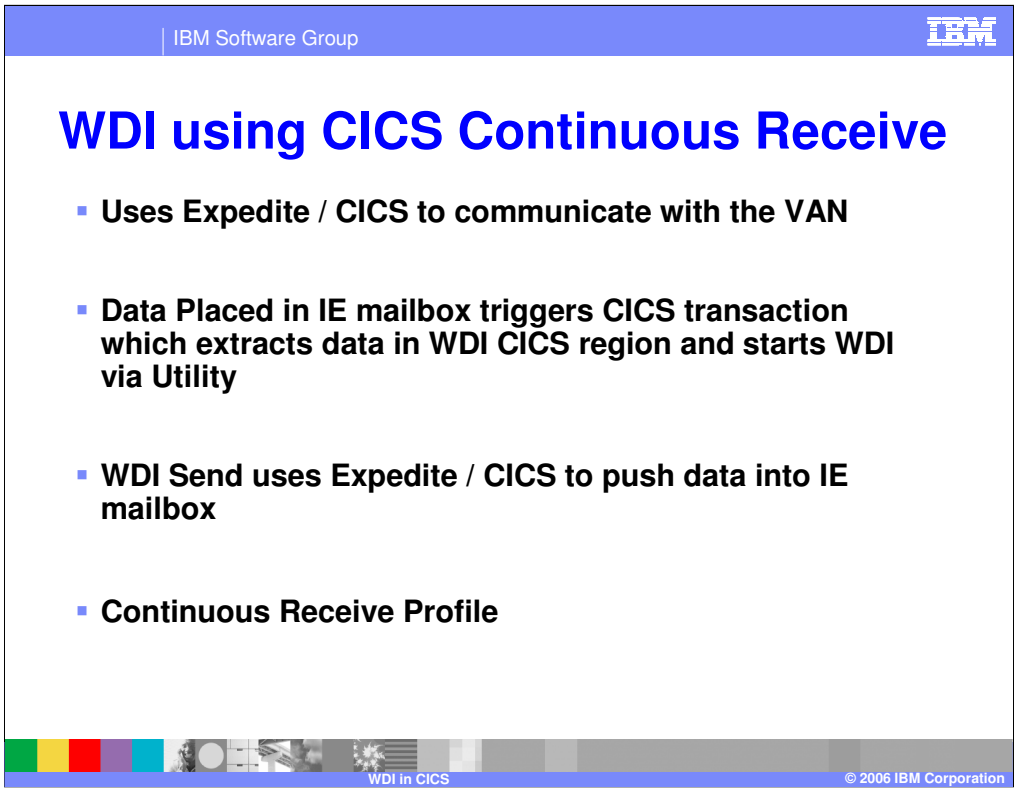
The WDI product has been around since 1984. Its primary purpose is to to convert or transform one type of data to another. It can convert flat file data to EDI format data and vice versa, it also can convert to and from XML formats.

Basic Flow



An enterprise deals with many trading partners (suppliers, distributors, clients, financiers)

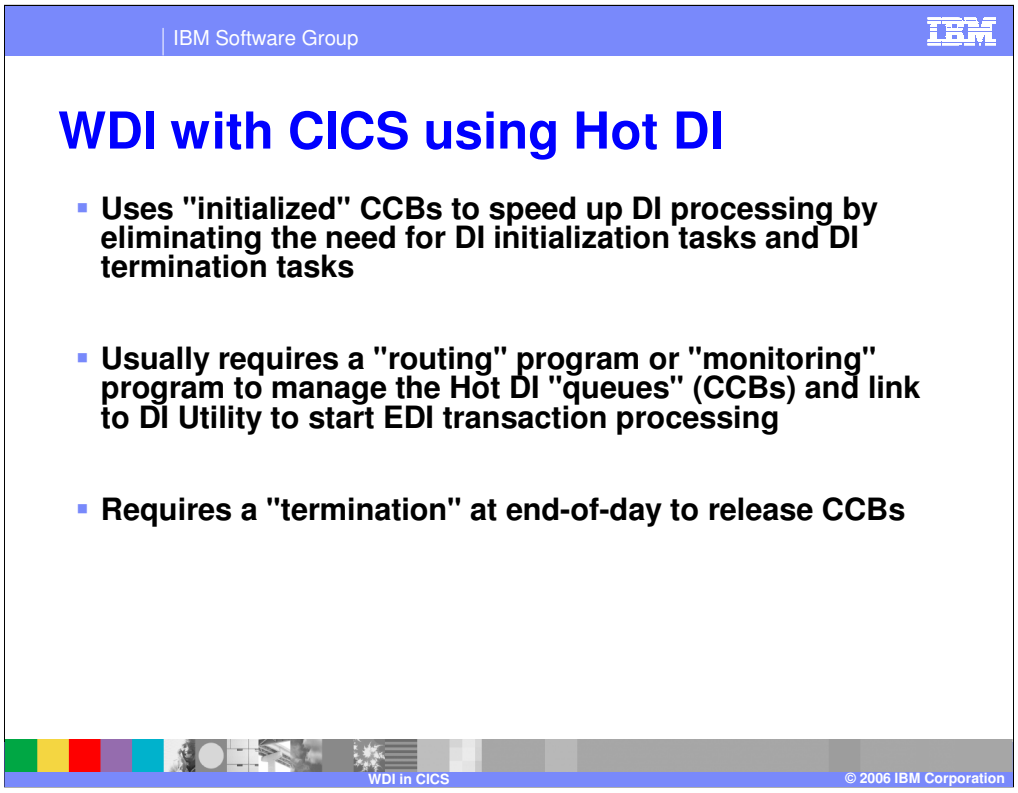
EDI processing is like a wheel with many spokes. Each spoke represents a Trading partner, the Hub is the business enterprise. WebSphere Data Interchange transforms data from the outside world (Trading Partners) using a server, like CICS, to data formats that can be used by internal applications. The flow can also go from applications back to the outside world.



Value-added networks (VANs) are vehicles used to communicate with the outside world. WebSphere MQ is another vehicle to do the same thing. VANs use a tool like Expedite (for the GXS Information Exchange VAN) to handle the communications. Mailboxes are used to separate data between trading partners.

Automatic triggerring of processes allows WDI and IE to extract and place data with mailboxes.

The WDI Continuous Receive profile is used to house information for use with the automatic triggering.

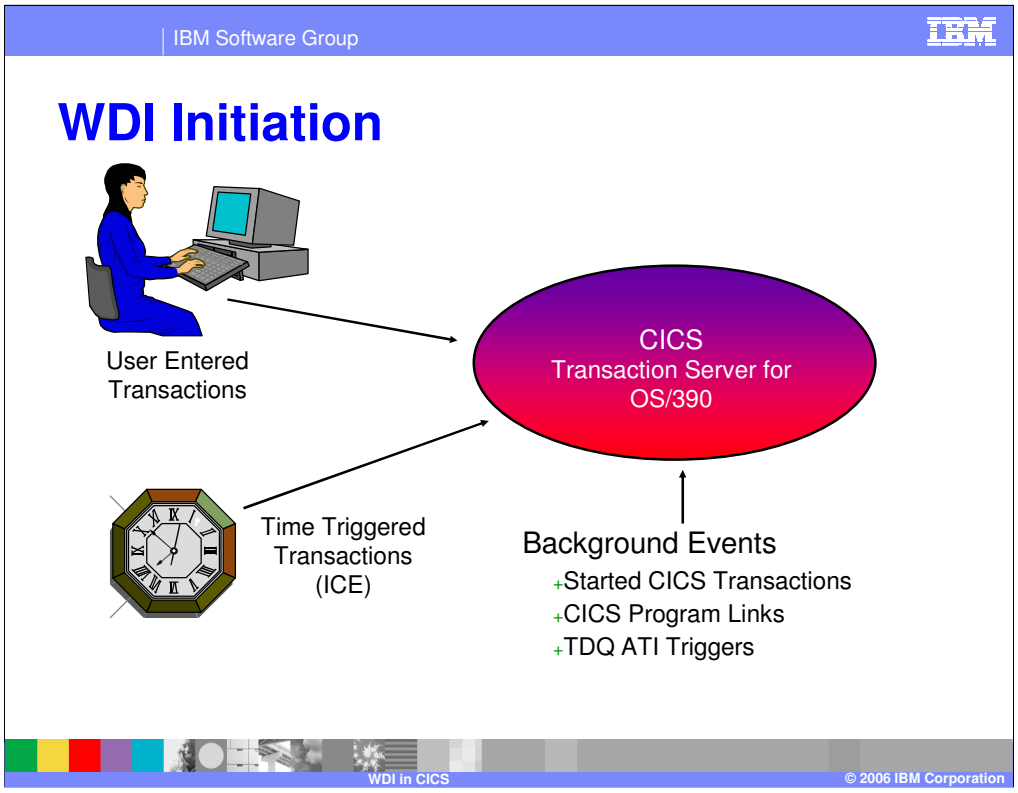


Hot DI is a WDI feature to improve the translation performance of WDI in CICS.

Hot DI uses "initialized" CCBs to speed up DI processing by eliminating the need for DI initialization tasks and DI termination tasks

It usually requires a "routing" program or "monitoring" program to manage the Hot DI "queues" (CCBs) and link to DI Utility to start EDI transaction processing.

A "termination" at end-of-day to release CCBs allocated is required.



WDI can be initialed in CICS either from a terminal or from some time-initiated transaction.

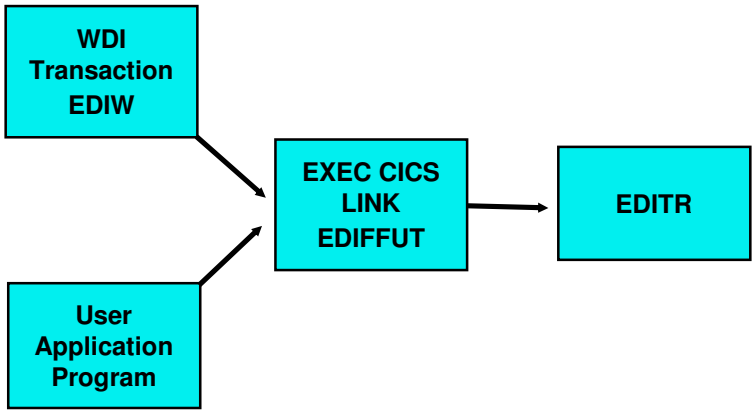
WDI Initiation

WDI can be started

- via a started task, START TRANSID('EDIB')
- via a link, LINK PROGRAM('EDIFFUT')
- via Automatic Task Initiation (CICS DCT)
- via EDIW

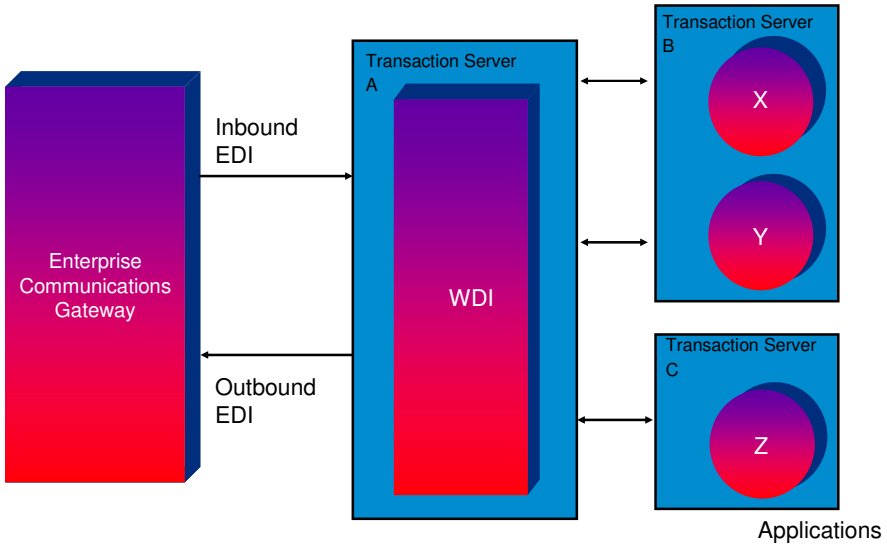
Technically, WDI can be initiated using the CICS START command, or a LINK. It can be initiated using CICS Interval Control (ATI) or even run manually with the WDI provided transaction EDIW.

WDI Interactive



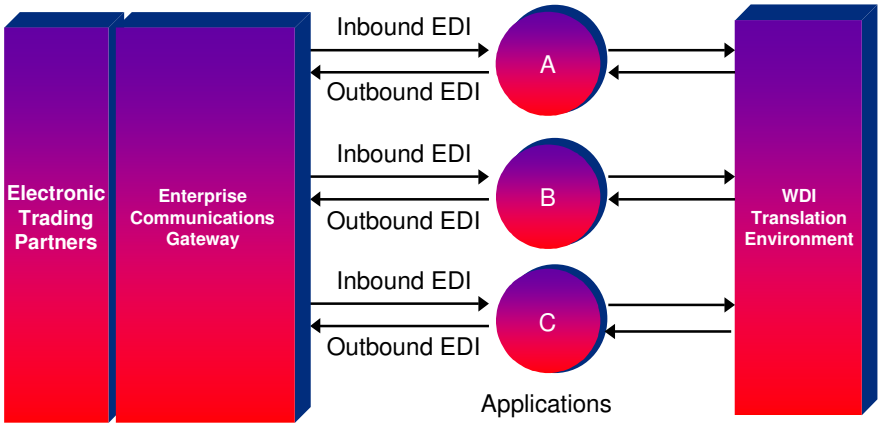
A user written program (even EDIW) typically uses EXEC CICS LINK PROGRAM('EDIFFUT') to transfer control to EDIFFUT, which eventually calls the EDITR module of WDI. EDITR is the internal name for the translator.

Multiple Processing Environments



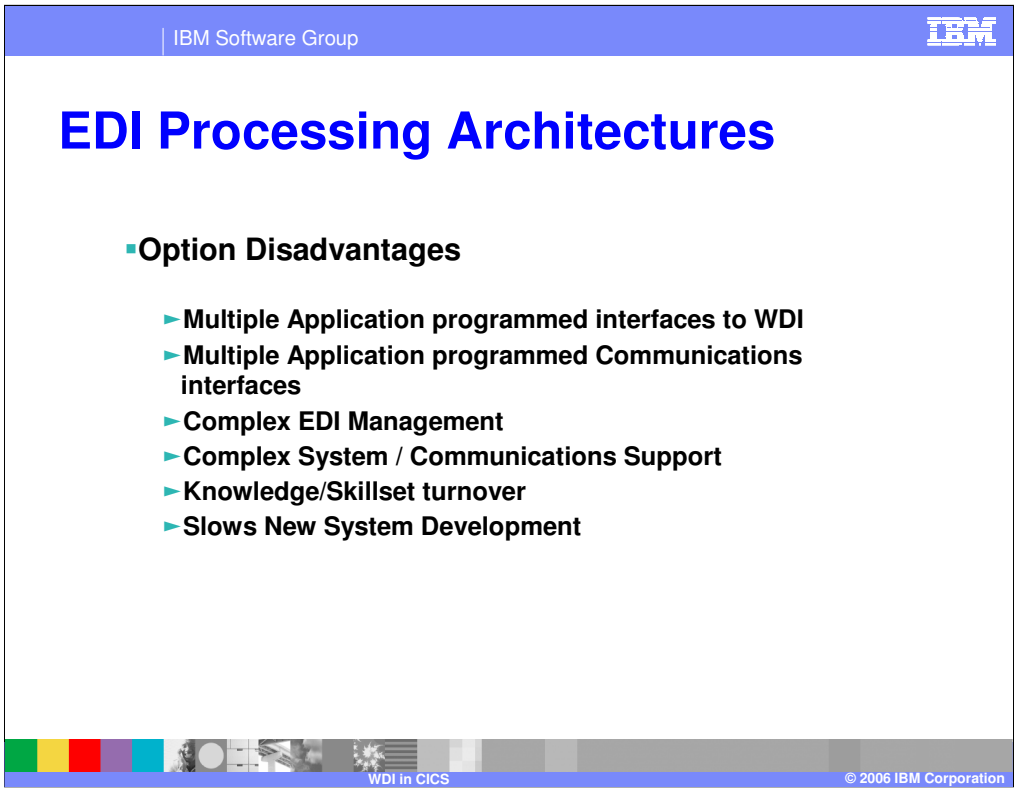
WDI works with both the CICS Multi-Region Option (MRO) environment and with CICS Inter System Communications (ISC) environment features.

EDI Processing Architectures



One EDI Processing architecture allows user-written applications to interface with the Gateways.

In this architecture, each application handles I/O from the network and links to resources (e.g. WDI for translation) that it needs.



Disadvantages with this option include:

Multiple Application programmed interfaces to WDI exist,

Multiple Application programmed Communications interfaces are needed

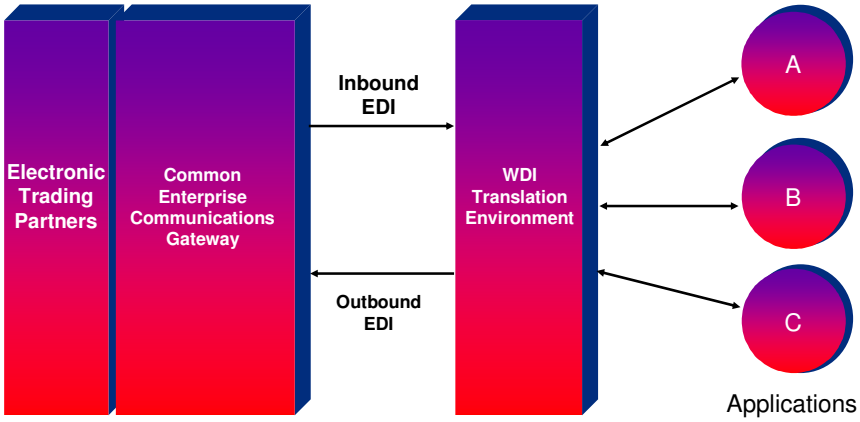
EDI Management nightmare, or at least highly complex,

System / Communications Support nightmare,

The Knowledge and / or Skillset turnover for new staff is extensive, and

The complications slow New System Development.

EDI Processing Architectures

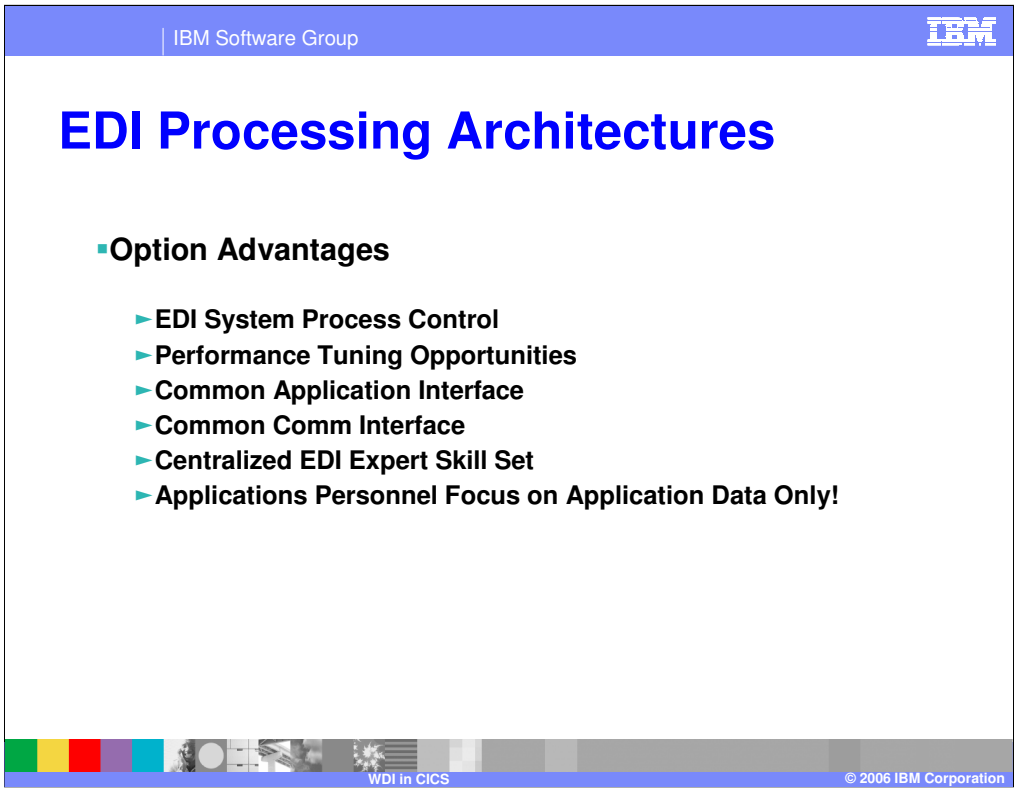


An alternate architecture is to let the processing resource (WDI) handle the data interface.

In this way the applications are removed from data acquisition handling considerations, leaving

an Application with a single way of receiving data.

WDI has a VAN interface, a MQ interface, uses TSQs as input, can read from VSAM files, etc. These components handle data acquisition for WDI and the architecture.



Advantages to using this architecture option are:

A central EDI System has Process Control of the applications

Performance Tuning Opportunities are enhanced by reducing interface points

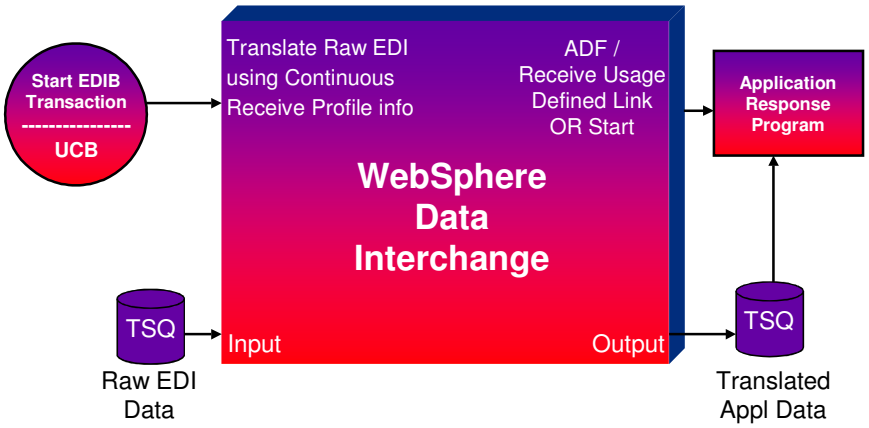
There is a Common Application Interface

There is a Common Communications Interface.

There exists a centralized EDI Expert Skill Set which is more easily transferred to new staff.

Applications development personnel focus on handling application data and the business process.

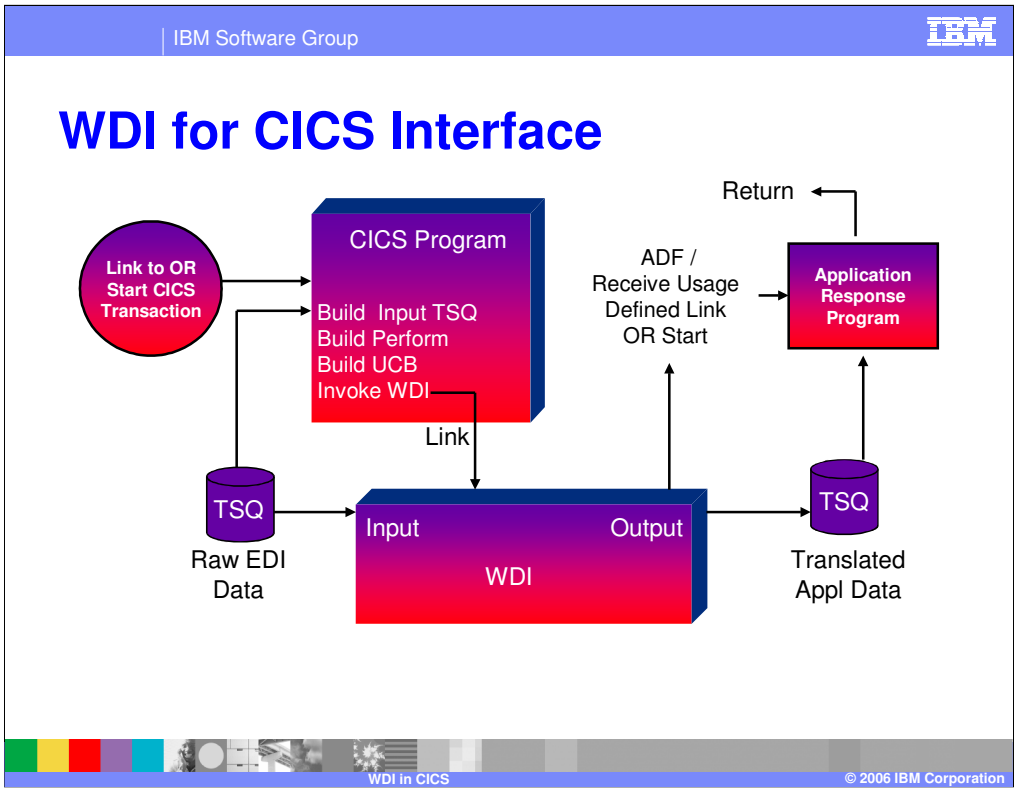
WDI for CICS Interface



A typical interface in CICS would be a START of EDIB as in Continuous Receive mode; EDIB is the WDI transaction ID for initiating the Utility

The UCB has a PERFORM statement and identifies the TSQs to be used

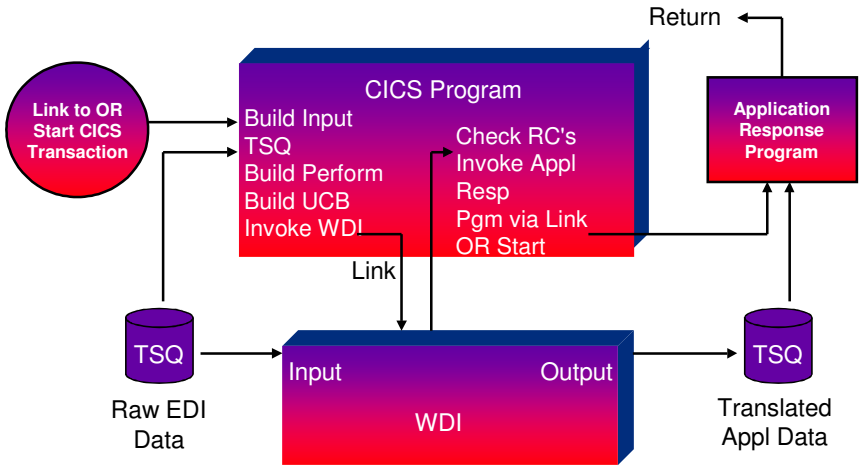
In this example WDI will initiate the Application Response Program after data is translated. An application response program is WDI's way of allowing business processing of the data to be written by the users.



In this architecture, a user program controls accesses application data and creates the initiation resources for WDI, then uses a CICS LINK to transfer control to WDI.

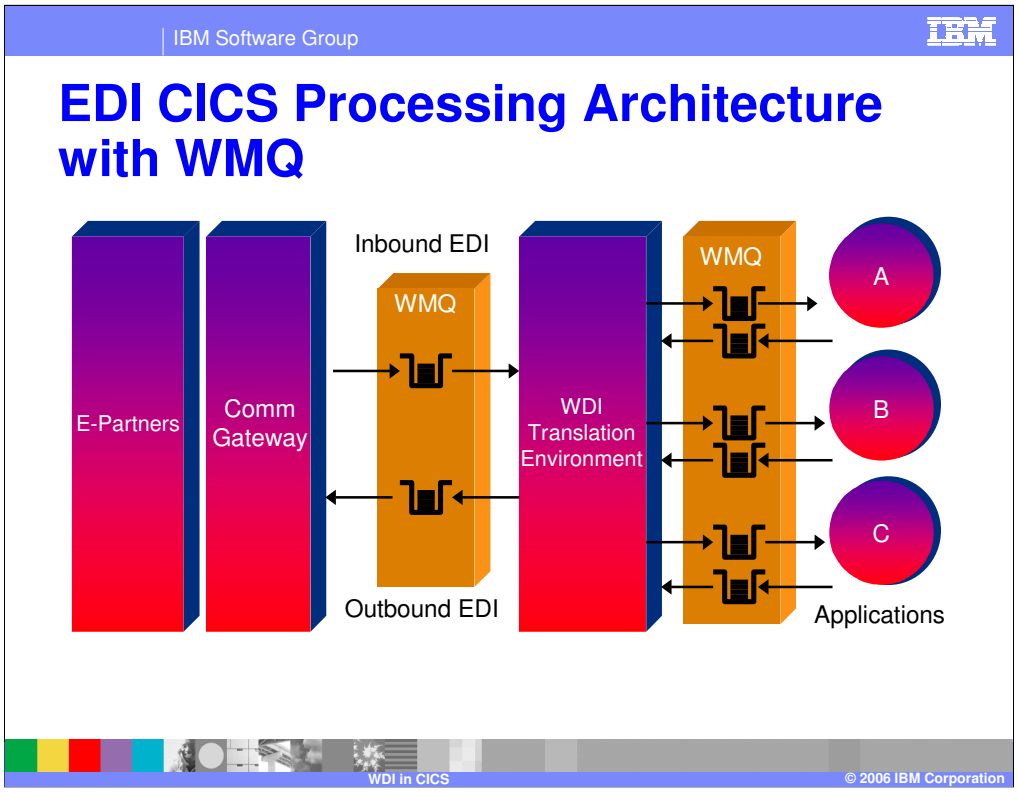
WDI then translates the data and uses the Map Usage to identify a response program to which WDI will return control.

WDI for CICS Interface

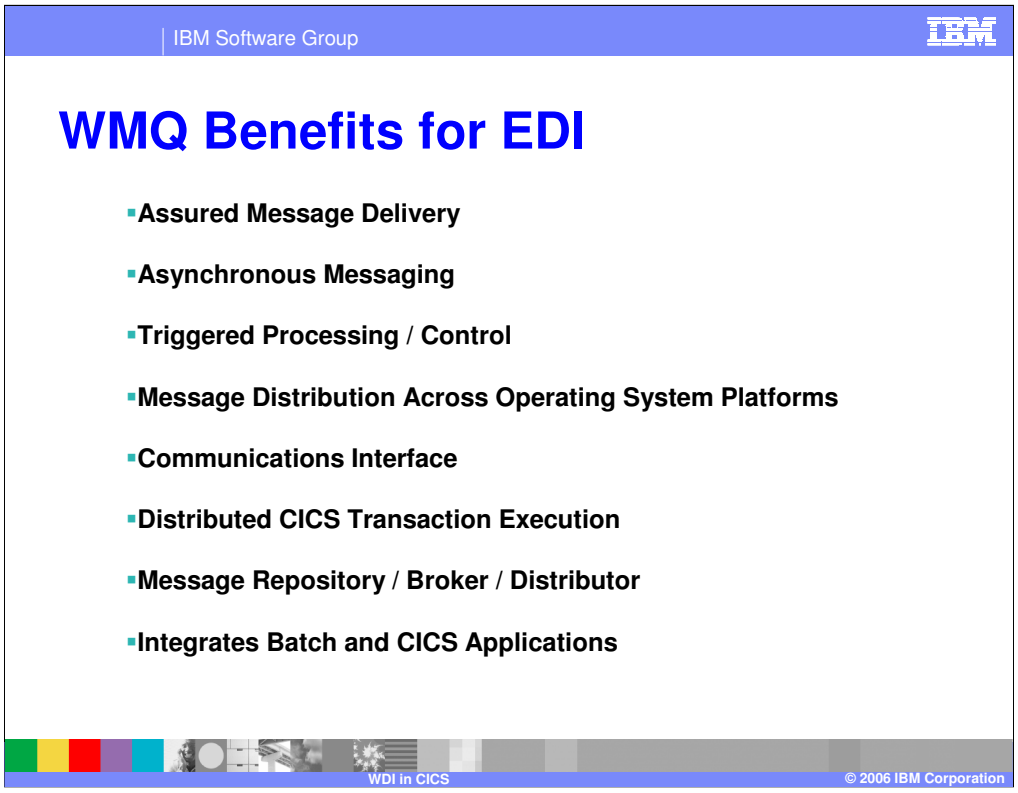


In this architecture, the user takes complete control of sequencing processes.

The user program uses a CICS LINK to transfer control to WDI, and then WDI returns to the calling program so that it can check return codes and determine the next process to invoke.



WebSphere MQ can also be used as the transport mechanism for an architecture. In this configuration, WMQ feeds WDI incoming messages and WDI feeds WMQ outgoing messages. Commonly, if WDI is being executed on a distributed system, WMQ can be used to feed user applications with translated messages and also WMQ can receive application messages, present them to WDI which transforms them to the format to be sent to external trading partners.



Benefits of MQSeries and the MQSeries Interface

- Assured Message Delivery – WMQ assures delivery or maintains the message on the queue

- Asynchronous Messaging – message transport is processed independently from the application

- Triggered Processing / Control – WMQ can automatically trigger processing, helping with a “hands off” operation

- Message Distribution Across Operating System Platforms – WMQ supports a number of different hardware platforms and operating systems

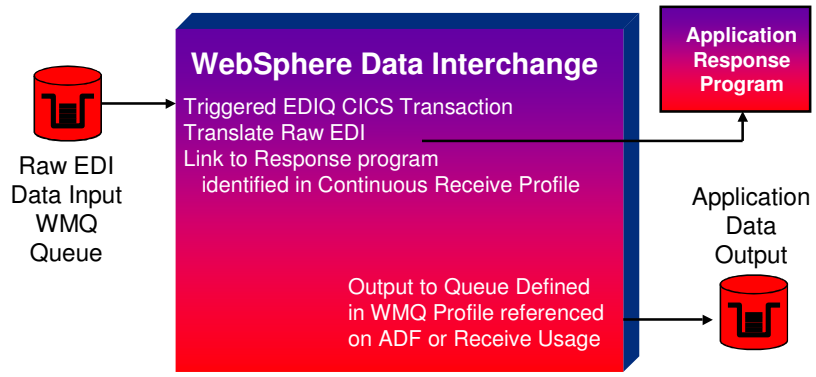
- Communications Interface – WMQ provides a common way of interfacing with distinctly different trading partner configurations

- Distributed CICS Transaction Execution – processing can be spread across multiple machines

- Message Repository / Broker / Distributor – these functions can be handled by WMQ

- Integrates Batch and CICS Applications – WMQ can be the interface between real-time and batch data and the systems that process the data

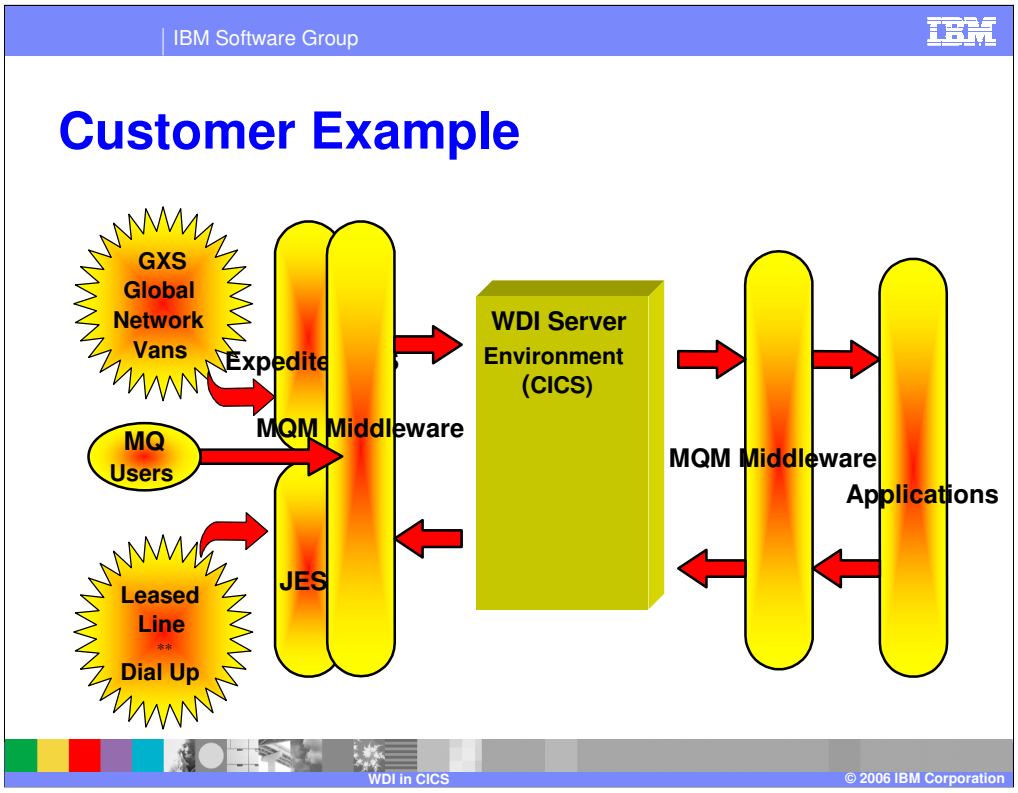
WDI WMQ Interface



WMQ works with WDI in the same manner as the Continuous Receive feature does with VANs.

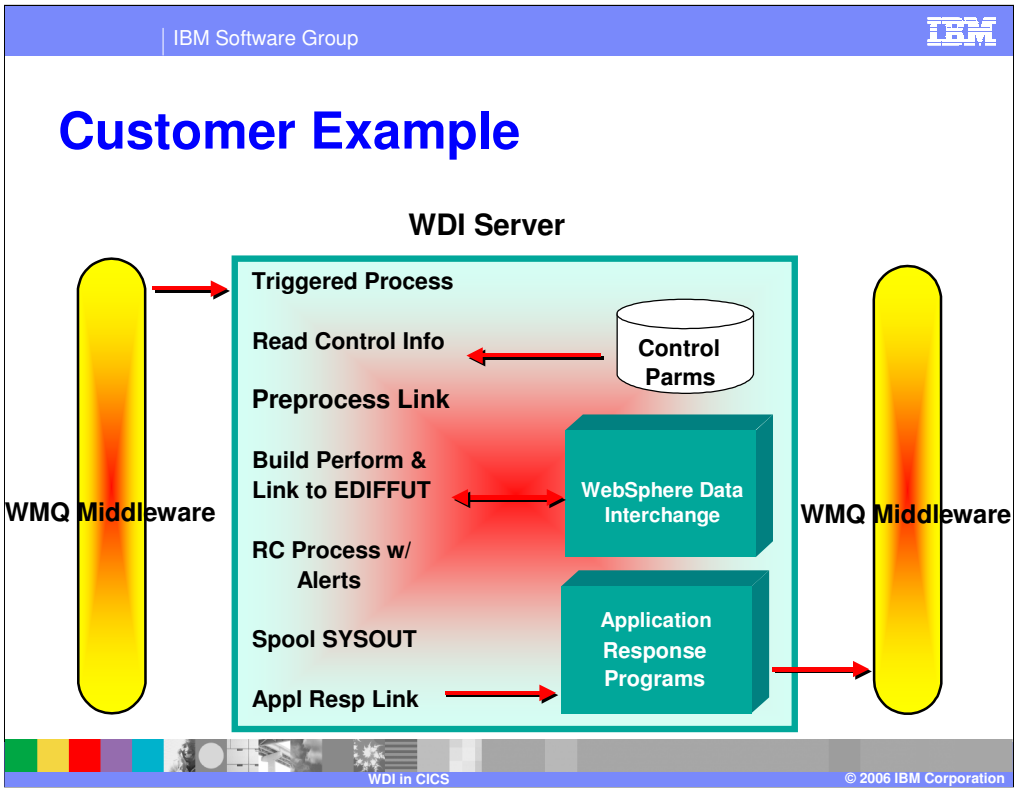
A WMQ trigger program monitors the WMQ Queue and initiates WDI.

Additional processing can be performed with a user-written Application Response Program.



By using WMQ to accept data from a variety of sources and platforms, the customer isolated the translation application

by using the WDI / WMQ interface. The customer then would sequence processes so that translated data could be routed to applications based on the translated data destinations.



A user program provided additional control and allowed for alert system interfaces and spool output retention.

Summary

- WDI is designed to operate in CICS and uses CICS capabilities
- WDI has established features specifically for execution in CICS, like Hot DI and Continuous Receive
- WDI has multiple ways it can be initiated to run in a “hands-off” mode or as a single, ad hoc transaction tool
- WDI’s ability to receive data from multiple sources allows it to be the hub in a processing architecture

In summary,

1) WDI is designed to operate in CICS and uses CICS capabilities,

2) WDI has established features specifically for execution in CICS, like Hot DI and Continuous Receive,

3) WDI has multiple ways it can be initiated to run in a “hands-off” mode or as a single, ad hoc transaction tool, and,

4) WDI’s ability to receive data from multiple sources allows it to be the hub in a processing architecture

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	MQSeries	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.