IBM

Implementing EDI Solutions

Introduction to EDI technologies and products

Multi-partner and multi-product implementation scenarios

Integration options with WebSphere Data Interchange and InterChange Server

Geert Van de Putte
Krishna Bathini
Kiran Chandu
Ronan Dalton
Arpit Doshi
Reza Ghorieshi
Bhushan Mahashabde

Redbooks

ibm.com/redbooks

International Technical Support Organization

Implementing EDI Solutions

October 2003

SG24-6906-00

Note: Before using this information and the product it supports, read the information in “Notices” on page vii.

First Edition (October 2003)

This edition applies to Version 3, Release 2 of WebSphere Data Interchange (product number 5724-C50).

Contents

Notices

Trademarks

Preface

This IBM® Redbook introduces the reader to the world of EDI. In addition to general terms about EDI, it also introduces a number of products in this area. WebSphere® Data Interchange is discussed as the translation engine to map EDI documents to and from documents in other formats. The redbook also introduces two communication products that use Internet technologies: iSoft's P2PAgent and Trading Partner Interchange.

In addition to product introductions, the redbook describes several implementation scenarios in a multi-partner and multi-product environment. Besides a network where trading partners only use iSoft's P2PAgent, we also look at a setup where trading partners use a combination of the two products.

For each communication product, we investigate several integration options with internal applications and other middleware. We discuss the integration options with the translation product WebSphere Data Interchange and with the process integration product WebSphere BI Interchange Server. The integration technique can be file-based or messaging-based.

Finally, we take a look at options to combine the flexibility of the Internet with the reliability of value-added networks. When Internet connectivity is temporarily not available, a trading partner can use Expedite to dial into IBM's network and send or receive EDI documents. By exploiting the recycle mechanics in iSoft's P2PAgent, we can implement a solution that provides a highly available connection between trading partners.

The team that wrote this redbook

This redbook was produced by a team of specialists from around the world working at the International Technical Support Organization, Raleigh Center. This redbook consists of materials built by several teams and portions of it were published earlier as two Redpapers.

Geert Van de Putte is an IT Specialist at the International Technical Support Organization, Raleigh Center. He is a subject matter expert in messaging and business integration and has seven years of experience in the design and implementation of WebSphere MQ-based application integration solutions. He has published several Redbooks™ about messaging and business integration solutions. Geert has also taught several classes about messaging, business integration and workflow. Before joining the ITSO, Geert worked at IBM Global Services, Belgium where he designed and implemented EAI solutions for customers in many industries. Geert holds a Master of Information Technology degree from the University of Ghent in Belgium.

Krishna Bathini is a certified Senior EAI Consultant at Miracle Software Systems, Inc, Detroit. He is an expert in system architecture, interface design and development using CrossWorlds®, CrossWorldsTPI, WebSphere MQ, SAP and Java™. Krishna also worked on WebSphere Data Interchange and iSoft's P2PAgent. Krishna holds a Master of Engineering degree from Andhra University of Visakhapatnam, India.

Kiran Chandu is a Senior EAI consultant and CrossWorlds Solutions Expert from Miracle Software Systems, Inc, Detroit. He has three years of experience in the design and implementation of EAI solutions and two years of experience in Web-based solutions. Kiran holds a Master in Information Technology degree from India.

Ronan Dalton is an e-Procurement Specialist with IBM in Ireland. He has two years of experience in IBM Procurement and has more recently worked as EDI Lead with the Dublin Procurement e-Services Team. Ronan holds a degree in Business and Legal Studies from University College, Dublin, a Post-Graduate Diploma in Computing from Griffith College, Dublin, and is currently studying for a Master's Degree in Computing.

Arpit Doshi has more than four years of experience in the analysis, design and development of banking and financial applications, primarily using C++, Java, EJB 1.1/2.0, Weblogic Application Server, EDI, ATG-Dynamo Application Server, Oracle, Rational® Rose®, and UML. Arpit has a very good knowledge of databases, is BEA Certified and at present preparing for his OCP certification. He received his bachelor's degree in Engineering from Thadomal Shahani Engineering College, Mumbai, India.

Reza Ghorieshi is a Global Technical Strategist and Senior Consulting IT Architect for the IBM Software Group. He has over ten years of experience in the IT industry and has been focusing on the services provider industry and the retail/distribution sector for the past four years. In his current position, he designs and formulates the IBM Software Group's business and technical strategy to drive industry-specific e-business solutions using IBM middleware and ensuring open standards and interoperability with industry leading organizations such as the Unified Code Council (UCC), Global Commerce Initiative (GCI), and many others. In addition, he has served as Solution Architect for the WebSphere Business Components product (formerly IBM San Francisco Framework). He came on board bringing strong Java architecture experience, object-oriented analysis, and design solutions to the San Francisco team. Prior to IBM, he co-founded Penumbra Software, a leading Java start-up focusing on Enterprise Java Development.

Bhushan Mahashabde is a Solutions Architect at Y-Point Inc, New Jersey, USA. His areas of expertise include e-commerce, J2EE, security, and biometrics. He has also worked extensively in developing telecommunications applications. Bhushan holds a degree in Computer Science from Pune University, India.

Thanks to the following people for their contributions to this project:

Shivendra Dubey, Sreevidhya Gnanasekaran, Ajit Mahajani, Daljeet Singh Sarna,
Y-Point Technologies

Nagaraju Goriparti, Gopal Krishna Nemani, Sunil Kundur, Meher Jyothi Kopparthi, Ravi Pydi,
Bhushan MahashabdePrasad Babu Vuppu, Murali Maka, Srinivas Ryali,
Miracle Software Systems

Pushkar Suri
Netcom Systems

John Hatfield
IBM

Become a published author

Join us for a two- to six-week residency program! Help write an IBM Redbook dealing with specific products or solutions, while getting hands-on experience with leading-edge technologies. You'll team with IBM technical professionals, Business Partners and/or customers.

Your efforts will help increase product acceptance and customer satisfaction. As a bonus, you'll develop a network of contacts in IBM development labs, and increase your productivity and marketability.

Find out more about the residency program, browse the residency index, and apply online at:

ibm.com/redbooks/residencies.html

Comments welcome

Your comments are important to us!

We want our Redbooks to be as helpful as possible. Send us your comments about this or other Redbooks in one of the following ways:

- Use the online **Contact us** review redbook form found at:

  ibm.com/redbooks

- Send your comments in an Internet note to:

  redbook@us.ibm.com

- Mail your comments to:

  IBM Corporation, International Technical Support Organization
  Dept. HZ8 Building 662
  P.O. Box 12195
  Research Triangle Park, NC 27709-2195

1

Introducing EDI technologies and products

This chapter provides a brief introduction to EDI technologies in general and how EDI solutions have evolved in the world of Internet technologies. The chapter goes on to introduce the reader to two business-to-business gateway products, iSoft's P2PAgent and Trading Partner Interchange (TPI), which both implement EDI technologies for the Internet world. We also introduce the reader to WebSphere Data Interchange, which is an EDI translation engine.

1.1 EDI terms and concepts

Electronic Data Interchange (EDI) is a concept that has been in commercial use for more than 30 years. It is widely accepted by companies all over the world as the way to electronically exchange business data.

Over the years, we have seen a variety of interpretations of the term EDI. A common and basic definition of EDI is: the transfer of business data between computer applications using a mutually agreed standard to describe the data contained in the message.

Typically, this means that business data is extracted from a company's internal application in an application-specific data format. This data extraction can be implemented in several ways. It can be a daily batch job reading information in a database and generating a file in an application-specific format. This user format data file is then translated into a standardized format such as EDI or XML and transmitted over a network to a trading partner.

An alternative technique is to generate a WebSphere MQ message from within the application. This WebSphere MQ message will again be in an application-specific format.

For both techniques, the message will end up in some way at a translator component, where the application-specific structure of the data will be mapped to an EDI standard format, as shown in Figure 1-1.

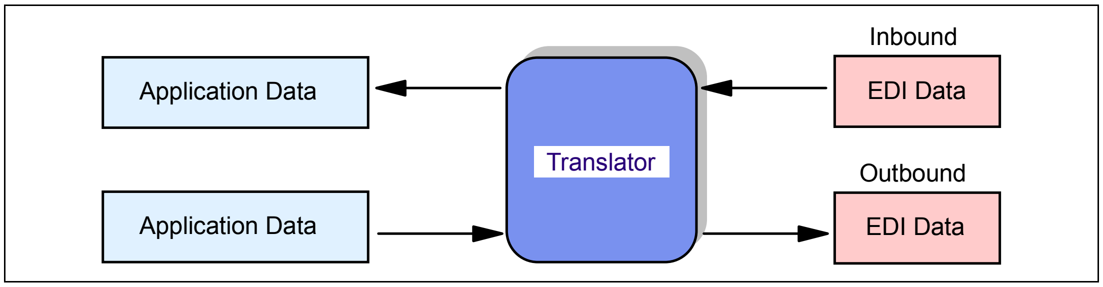


Figure 1-1 The role of the translator

The receiving trading partner then re-translates the received EDI/XML message back into an application-readable format that fits into their processes, as shown in Figure 1-2 on page 3.

An EDI-based information exchange is usually a two-way process. Thus, the translator component will also be used to translate incoming EDI messages into an application-specific format.

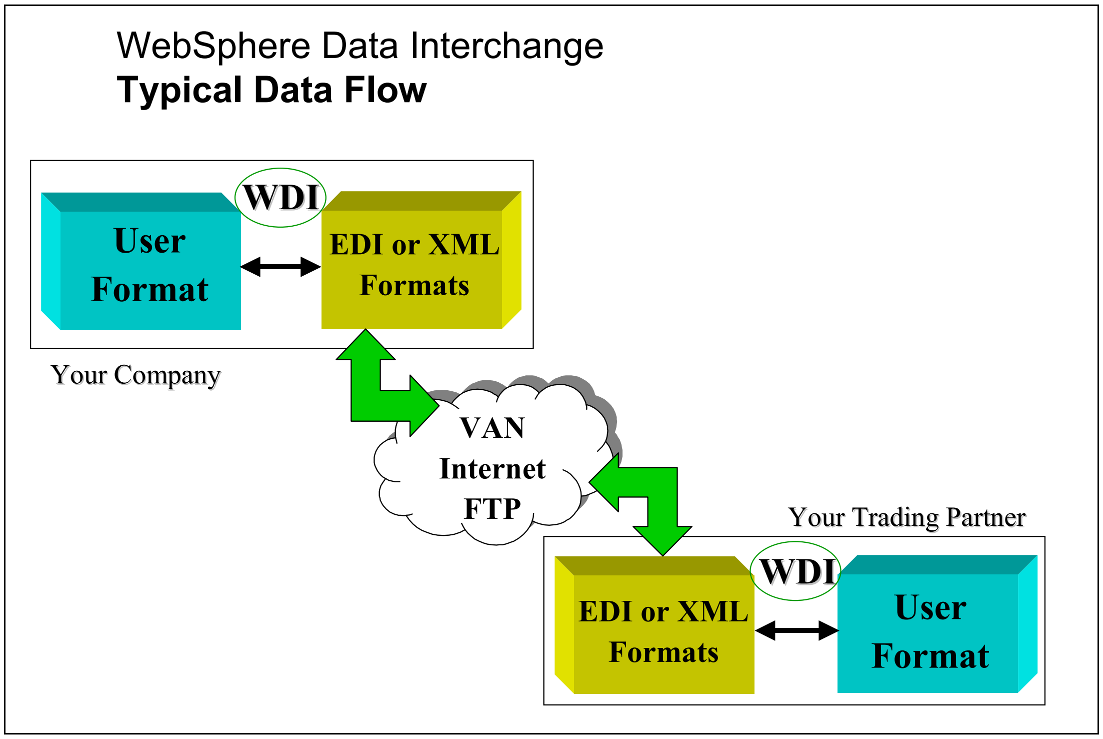


Figure 1-2 The very basic EDI message flow showing where WebSphere Data Interchange fits in

From a company perspective, the EDI concept means business integration and process automation. Business documents such as purchase orders, invoices, shipping notices, and price catalogs are exchanged between companies over a network in a structured and computer processable format.

Figure 1-3 on page 4 shows a typical flow of actions and data between a buyer and a seller. Usually, a buyer will ask for a quote and when a quote has been received, a purchase order request might be sent out by the buyer to the supplier. This information exchange is typically handled by a purchasing application at the buyer side and handled by a sales management application at the seller side.

When the goods are ready to be delivered, a shipping notice will be sent by the seller to the buyer. This time, the information exchange is very probably between different components of the IT infrastructure at both the seller side and the buyer side.

Thus, the use of EDI between two companies implies integration between the applications at each end. The application used in the warehouse or the accounting application needs to know about the purchase order generated by the purchase application.

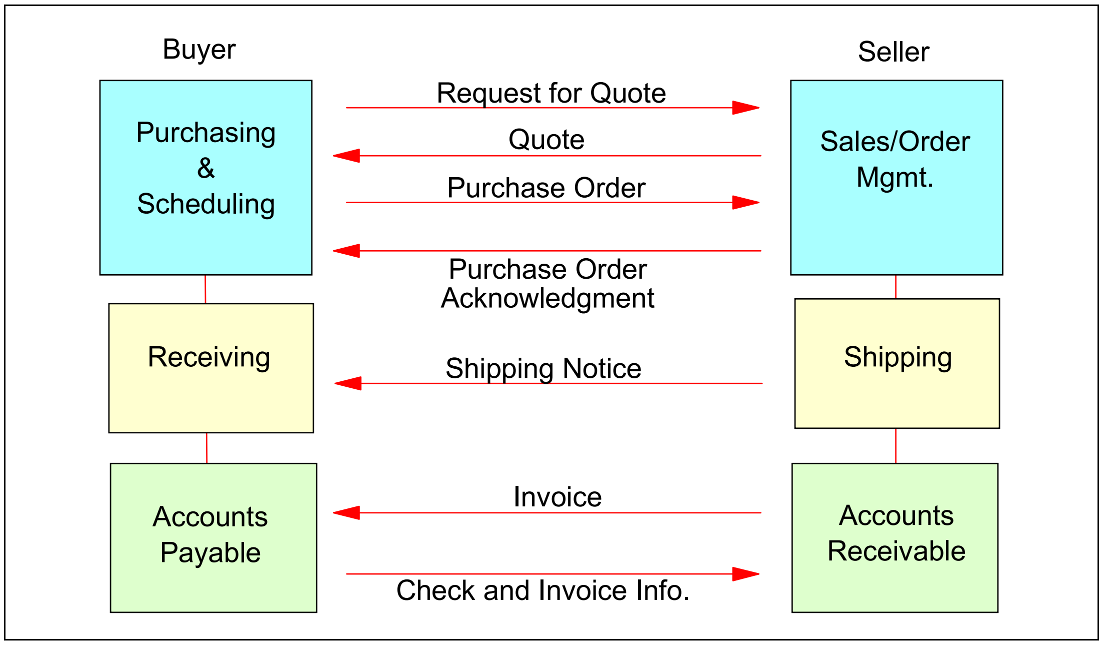


Figure 1-3 EDI and the business cycle

Since the early days of EDI, a lot of new initiatives and techniques have been adopted by the market. Terms that hardly existed at that time, such as the Web, XML, B2B and Business Process Management, are a natural part of today's realities. So the obvious question is, why is EDI still so important?

- EDI is a mission-critical part of companies' B2B strategies
- 95% of Fortune 500 companies use EDI
- 80% of business transactions are conducted via EDI value-added networks (VAN) today
- EDI continues to deliver significant return on investment
- EDI continues to evolve in response to new enterprise and industry requirements, as well as competitive pressures (for example, HIPAA, AS1, AS2)

1.2 Benefits of EDI

The market is driving every business to act smarter and more quickly and to be more visible. Much of this can be achieved using EDI. Additionally, EDI can give companies a better knowledge of their markets, because it opens up possibilities to collect and analyze information from the EDI transactions the companies are generating.

Among the most visible benefits of adopting EDI are:

- Reduction of data entry errors
- Reduced cycle time
- Minimization of paper use
- Improved relationships with your business partners
- Information in electronic form is more easily shared over the organization
- Improved inventory management

1.3 EDI components

The term EDI is a concept; it does not define any technique and does not point to any specified product or service. An EDI transmission can basically be divided into two logical parts: the message itself and the communication.

1.3.1 Message standards

Since the goal of EDI is to have a standardized message, a number of different standards have been developed and established over the years. The most commonly used message standards are:

- ANSI ASC X12 - US standard
- EDIFACT - standard recommended by the United Nations, used mainly in Europe
- UNTDI - UK retail standard
- ODETTE - European automotive industry
- Others such as HIPAA, VICS, VDA, UCS, etc.

The standardized messages are built by components such as elements, segments, and transactions/messages. Between all objects, there is a separator.

The *elements* are the individually defined fields such as Amount, Name or Quantity. Two or more elements can be grouped together, forming a *composite element*.

A *segment* is a set of elements or composite elements built to a logical entity, such as Name and Address or Pricing Information.

An *envelope* contains overall information about the transaction or message, such as sender and receiver, type, and control values.

A set of segments put together in a specified order all wrapped in an envelope make up a *transaction* or *message,* such as an Invoice or a Purchase Order. The envelope contains information about the sender and receiver, transaction/message type, and so on.

Figure 1-4 shows the structure of an EDI message in a graphical way.

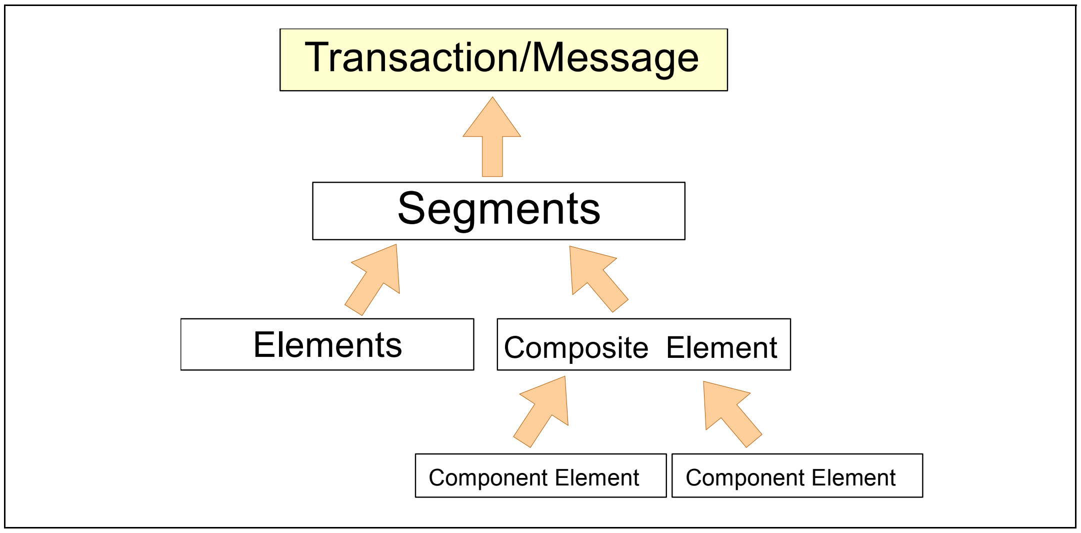


Figure 1-4 Components of an EDI message

EDI interchange messages have a control structure made up of functional groups nested within an EDI envelope, as shown in Figure 1-5. Functional groups, in turn, contain one or more transaction records. The envelope header contains information such as the source address, destination address, time stamp and return receipt (if any). A variety of addressing methods are supported, including Duns Numbers, Standard Carrier Alpha Code, Phone Number, CCITT X.121 Address, etc.

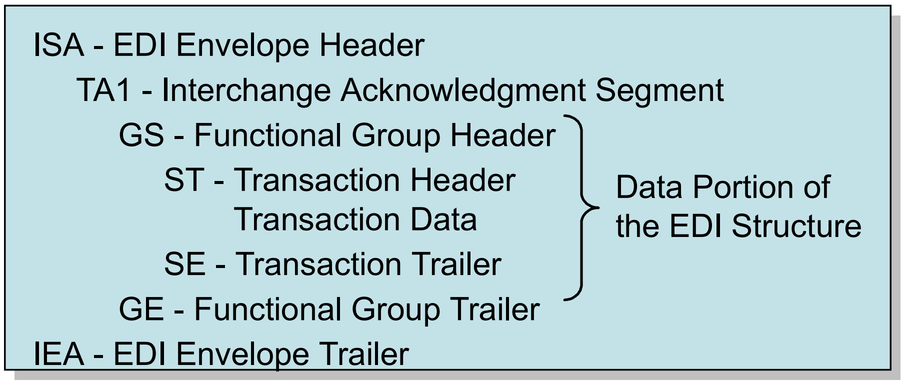

ISA - EDI Envelope Header
TA1 - Interchange Acknowledgment Segment
GS - Functional Group Header
ST - Transaction Header
Transaction Data
SE - Transaction Trailer
GE - Functional Group Trailer
IEA - EDI Envelope Trailer
Data Portion of the EDI Structure

Figure 1-5 Interchange control structure

Transaction records include transaction headers, transaction trailers and a transaction data segment. An important component of the transaction header is the transaction code. For example, the code for a purchase order is 850 and the code for an invoice is 859. The data segment of the transaction record itself is made up of multiple data elements separated by data element separators. An example data segment for a purchase order transaction is illustrated in Figure 1-6.

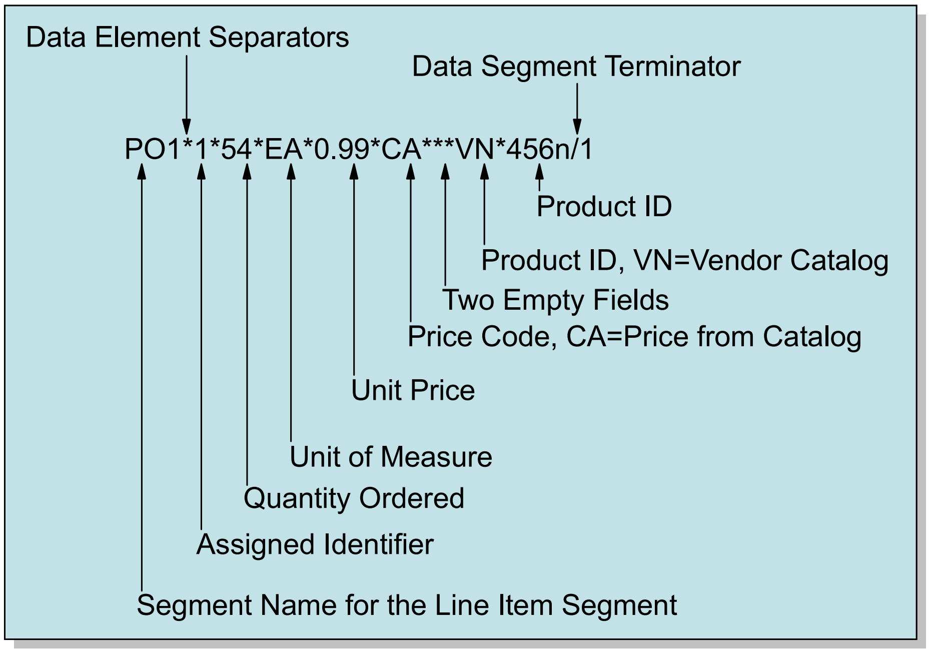


Figure 1-6 X12 encoding purchase order sample

Example 1-1 on page 7 shows a sample X12 transaction. The first three lines are part of the envelope. The line starting with `ST*810` is the start of the actual message. This time it is an 810 message, which is used to send invoices.

Example 1-1   X12 transaction (invoice)

```
ISA*00*          *00*          *ZZ*CELORGC02      *ZZ*IBMIRLPROD     *021018
*0229*U*00401*000008899*0*P*~¬
GS*IN*CELITALY*IBMIRELAND*20021018*0229*8899*X*004010¬
ST*810*88990001¬
BIG*20021017*0002146553**P350342***DR¬
CUR*SE*USD¬
REF*D2*0080118614¬
N1*SE*Celestica Italia S.r.l.*92*103015¬
N2*Celestica Italia S.r.l.¬
N3*Via Lecco 61¬N4* Vimercate - MI - IT**20059¬
REF*GT*IT03029690967¬
N1*BY*IBM INTERNATIONAL HOLDINGS¬
N2*IBM INTERNATIONAL HOLDINGS¬
N3*MULHUDDARTH¬
N4*DUBLIN*DB*15 ¬
REF*GT*IE6602632V¬
IT1*000001*4*EA*1767.87*PE*BP*00004N3524*VP*4N3524¬
TXI*VA*0*0¬
PID*F****BK C_F_CARDINAL¬
REF*ZZ*7071.48¬TDS*707148*707148¬
TXI*VA*0¬
CTT*1¬
SE*22*88990001¬
GE*1*8899¬
IEA*1*000008899¬
```

Both the X12 and the EDIFACT transactions in Example 1-1 and Example 1-2 are presented with one segment per row for easier viewing. Normally, a new segment follows directly after the previous segment to save space.

Example 1-2   EDIFACT message (Purchase Order)

```
UNB+UNOA:2+3568579005454:14+3015437860102:14+021003:0053+02018852760++ORDERS'
UNH+1+ORDERS:D:93A:UN:EAN007'
BGM+220::9+001779'
DTM+137:20021002:102'
DTM+2:20021005:102'
DTM+63:20021005:102'
NAD+BY+3568579005454::9'
NAD+SU+3015166100102::9'
NAD+DP+3568579005454::9'
CUX+2:EUR:9+3:EUR:4'
LIN+1++3560998032054:EN::9'
QTY+21:2'QTY+59:1'
PRI+AAA:798.33::NTP'
LIN+2++3560998032054:EN::9'
QTY+21:5'QTY+59:1'
PRI+AAA:34.6::NTP'
UNS+S'
UNT+17+1'
UNZ+1+02018852760'
```

From a message organization point of view, these look similar. Every segment starts with a three-letter word indicating the type of segment that follows. Each element within the segment is separated from the next one by a separator. Finally, the message structure uses a segment terminator. There are additional rules. Some elements are optional, other elements are conditional. Element A and B are labeled conditional when, for example, the appearance of element A implies the appearance of element B.

1.3.2 Communication

Transportation of the EDI file over a network can be done in many ways. Any network and any protocol can be used as long as it fits the needs. Three types of communication are discussed here:

- VAN communication
- Internet (AS1, AS2, FTP, etc.)
- WebSphere MQ

Note that we are focusing more on the communication aspect between two trading partners. There is also a communication aspect within the IT setup of a trading partner. The data has to be sent from the internal applications to the EDI translator software and after translation, the data has to be handed over to some communication software.

VAN communication

For connectivity and exchange of EDI data between enterprises, one option is a direct connection between the trading partners using the X.25 network or leased or dial-up lines. The direct communication method assumes that two partners communicate with a single data communication protocol out of over ten available options. This works well when only a few partners are involved or if one party can dictate to all their trading partners the single protocol to use. As the number of partners increases, so does the number of protocol options one must support, and therefore the management of the trading partner communications becomes more complex. This has been the primary driver for the advent of value-added networks (VANs). Unlike the Internet, which is public and free of charge except for connectivity to it, VANs are privately run; companies pay to be registered users and for services. VANs offer services such as EDI packet transportation, conversion between different EDI versions and standards, audit trails, security, trading partner identification, education, and consulting. Multiple VAN providers are in existence and bridges are in place between these in order to enable the subscribers of one to do business with the subscribers of another. Connectivity options to the VAN itself from any enterprise vary depending on the VAN provider and connectivity software vendor. Secure messaging-based connectivity using a messaging middleware solution such as WebSphere MQ is the usual choice.

Using a value-added network (VAN) for the transmission of files is traditionally seen as the most secure way of communication. Apart from pure communication, a VAN also provides value-adds such as:

- Built-in security features that help protect against unauthorized access to customer data
- Restart and recovery facilities that help to reduce or eliminate the impact of communications interruptions
- Archive capability for the online retention of data copies
- 24x7 availability
- Notification of message arrivals that meet predefined criteria, such as a message from a specific trading partner

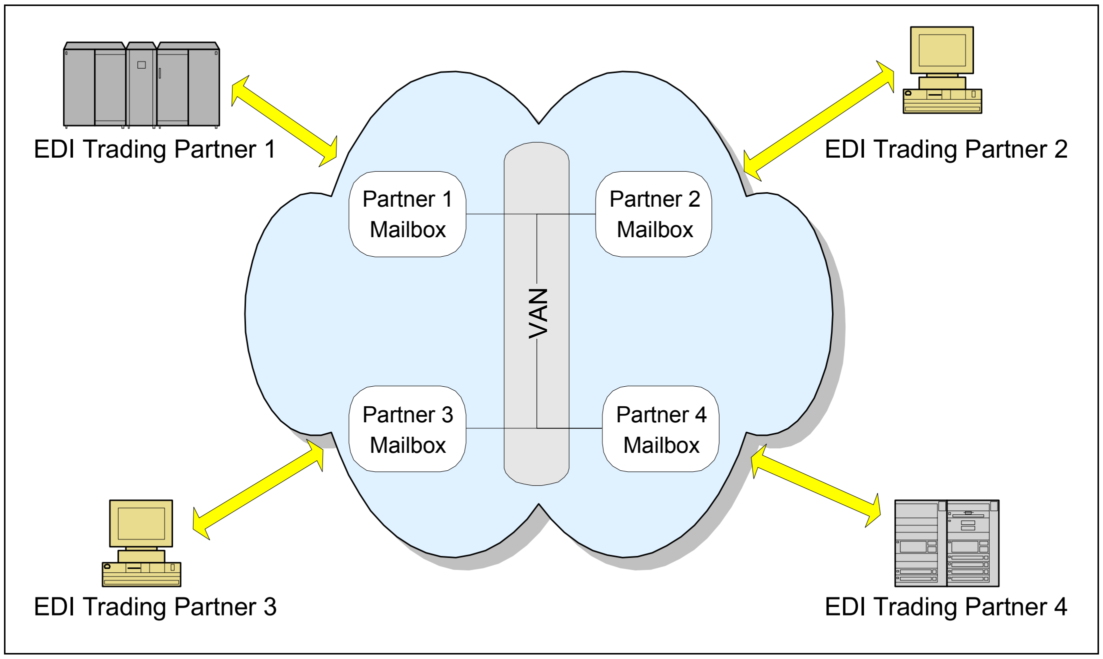


Figure 1-7 EDI VAN network

The VAN Gateway software will drop off and pick up EDI documents via the mailbox. The VAN provides store-and-forward mailbox services. The physical communication system between the VAN Gateway and the VAN network can vary from dial-up to FTP, or some proprietary communication technology.

IBM Information Exchange (IIN) is an example of such a value-added network.

EDI over the Internet

Although the VAN has been a viable proposition for many of the big corporations, its high cost has been a deterrent for its wider adoption by medium-sized and small businesses. Adoption by the latter has been driven largely by the dictates of the big corporations with which they do business. There has been a growing desire among businesses to explore means for driving down the cost of electronic data exchange.

In 1996, a working group called EDI-INT was created by the IETF (Internet Engineering Task Force) to create a set of secure protocols for conducting highly structured inter-enterprise exchanges over the Internet. The requirement was to create a method for packaging EDI/X12, UN/EDIFACT and mutually agreeable transaction sets in a MIME envelope. Several additional requirements were included for obtaining multi-vendor, interoperable service beyond how the EDI transactions are packed. These revolved around security issues, such as EDI transaction integrity, privacy, and confirmation of source and destination. Currently, there are two main EDI-INT initiatives, known as applicability statements AS1 and AS2, which describe how current Internet standards can be used to achieve VAN functionality.

- AS1 uses MIME (Multipurpose Internet Mail Extensions) and SMTP (Simple Mail Transfer Protocol).
- AS2 uses MIME and HTTP (Hypertext Transfer Protocol) for process-to-process real-time EDI.

Although created originally for transporting EDI formatted data, AS1 and AS2 can be used to transport a variety of data types, including XML documents.

The Uniform Code Council (UCC) and Drummond Group, Inc. have partnered to sponsor an interoperability testing program for software vendors providing AS2 connectivity solutions. For more information about the interoperability tests of the Drummond Group, visit their Web site at:

`http://www.drummondgroup.com`

Message queuing

Message queueing (MQ) connects commercial systems in today's business. It provides assured, once-only delivery of data in any format.

IBM WebSphere MQ is an example of this.

1.4 The evolution of EDI

In today's economy, market dynamics have converged on a business model that provides for the integration of different trading partners in a value chain. Depending heavily on Internet technologies, this model can enable highly coordinated trading communities, each with the ability to operate as a virtual enterprise.

In the virtual marketplace, business relationships are formed electronically. Buyers and sellers come together without the benefit of paper contracts, fee schedules, or sales people to close the deal. This Web economy requires an agile enterprise, one that can work more directly with suppliers and customers and respond more rapidly and intelligently to change. The need for flexibility and lower costs, such as VAN charges, are driving the evolution of EDI.

Organizations are recognizing the value of many years of investments in EDI. Rather than replace the present solution, they plan to extend and evolve the EDI transactions. This existing EDI solution is considered as a part of a multi-modal B2B gateway or hub alongside XML, Web solutions, and portals. By integrating B2B and EDI technologies, event-driven or process-driven integration models can be supported using the existing EDI solution.

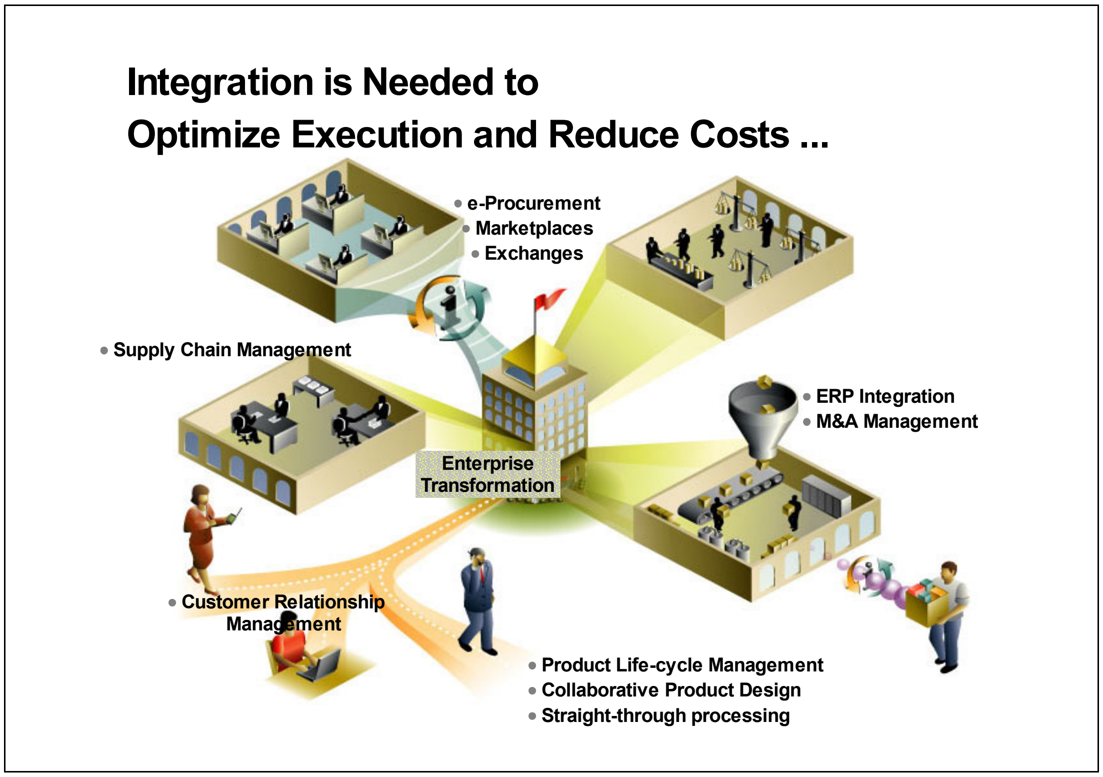

Integration is Needed to
Optimize Execution and Reduce Costs ...
e-Procurement
Marketplaces
Exchanges
Supply Chain Management
ERP Integration
M&A Management
Enterprise Transformation
Customer Relationship Management
Product Life-cycle Management
Collaborative Product Design
Straight-through processing

Figure 1-8 Business Initiatives = Business Integration

The Internet is widely perceived as being much less expensive than a VAN, but this is not necessarily the case. VANs generally provide valuable services, such as TPA management, service-level administration, security, and store-and-forward capability. The Internet requires you to manage these elements yourself, which means the total costs are not always lower than those of a VAN.

EDI users cannot realize the full value of the Internet in e-commerce applications until the entire underlying business process is optimized. Business process management (BPM) is the automation, optimization, and management of end-to-end business process flows. In this case, it is accomplished by integrating front-end Web applications to back-end legacy applications and to existing EDI trading partners.

Earlier phases of EDI achieved efficiency by automating manual processes. Now, however, the focus is on business process integration and optimizing business operations. EDI steps are tied to the full value-chain processes by the ability to share information throughout.

The result of these trends is that traditional EDI customers are facing increasing challenges to remain competitive. To grow or even preserve your business, you need to integrate your existing EDI applications with core business processes by distributing transactions or information to and from various back-end applications. Situations in which this kind of integration can help are as follows:

- You have typically spent tens or hundreds of thousands of dollars on your current EDI solutions and you want to leverage that investment.
- Internal departments lack timely information about EDI transactions and make costly mistakes or provide poor service.

- Competitiveness suffers from an inability to track or manage the distribution of EDI messages within the business.
- Manual processing of EDI messages is slow, error-prone and consumes valuable resources.

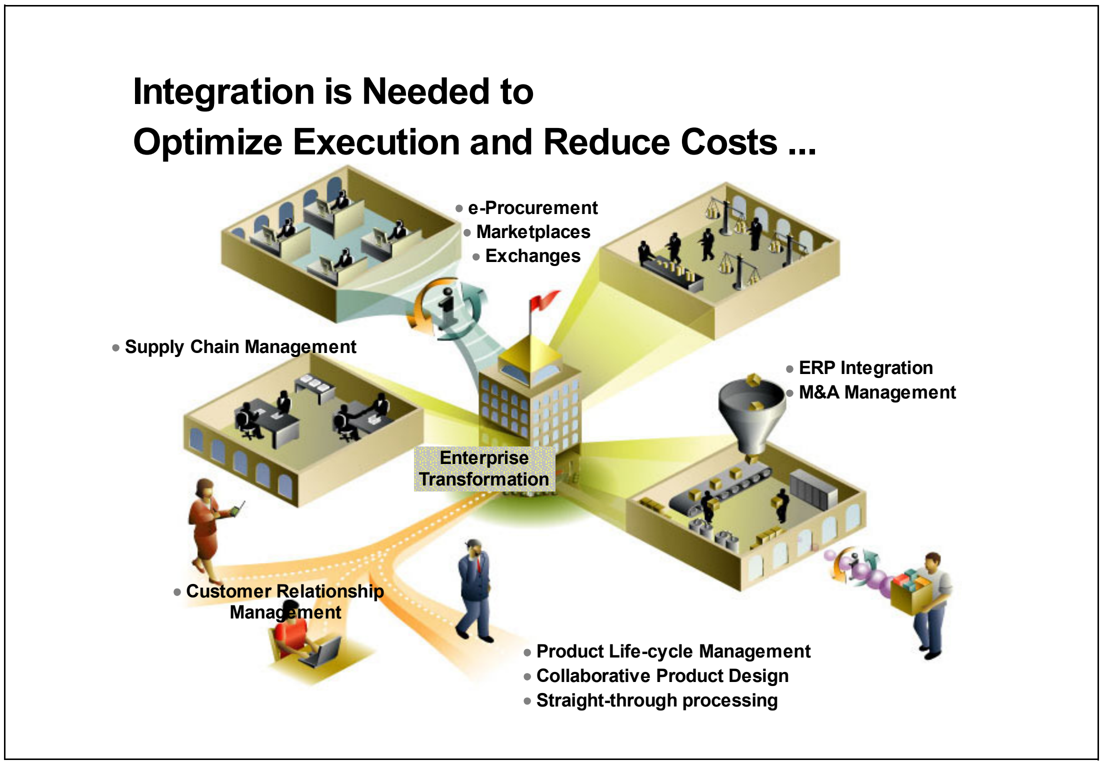

Integration is Needed to
Optimize Execution and Reduce Costs ...
• e-Procurement
• Marketplaces
• Exchanges
• Supply Chain Management
• ERP Integration
• M&A Management
Enterprise Transformation
• Customer Relationship Management
• Product Life-cycle Management
• Collaborative Product Design
• Straight-through processing

Figure 1-9 The industry view

1.4.1 Elements of an EDI solution

In addition to obvious components of an EDI solution, such as application programs and systems, VANs and trading partners, a complete and flexible solution should include the following important elements.

Translators

A universal problem in integration of applications is the conversion of shared data from one format to another. Common data fields, such as names, addresses, and numbers, often have different formats across disparate systems. The traditional approach to EDI implementation is to place the function that converts application data to the EDI standard directly into the business application. This approach is less effective because a separate program is required for each transaction as well as for each trading partner. In addition, it is difficult to keep up with new versions of standards because programs must be modified every time a trading partner adopts a newer standard or version of the standard.

This approach has changed with the introduction of third-party translation software, also known as mappers. The translator is responsible for mapping application data to the specific EDI format and vice versa. This translation software is implemented in either a centralized engine or in an adapter. It must handle primary EDI standards as well as different and evolving versions of each standard.

Batch enveloper/deenveloper

Typically, because VAN charging is based on each sent transaction, enterprises have been driven to find ways to reduce the number of transactions and to compress more information into each. Consequently, EDI messages are sent in large batches, which can then be grouped from, or split out to, several divisions or areas of an enterprise.

Enveloping batch messages involves placing the EDI standard header and trailer around transactions in preparation for sending. When the envelope is complete, the package can then be sent to a trading partner through a VAN. Similarly, batch transactions must be deenveloped when they are received from the VAN.

Message router

Once the EDI message is deenveloped, it can be divided into function groups. Each function group may relate to a different division or area of the business. A mechanism is needed to sort messages destined for different groups and deliver them to the appropriate target applications. This means there is a requirement to fan in and fan out messages. Message transformation may also be required to get the message into the correct format for the end applications.

Trading Partner Agreements

A TPA is an agreement related to the exchange of information in electronic transactions. The term includes a particular agreed-upon standard for business documents as well as communications and business protocols, the service-level agreement, and more. TPAs can also be extended to include business events. For example, if an event occurs in one organization that might affect processes in a second organization, the TPA can specify that the second organization be alerted to the event.

1.4.2 The IBM EDI solution

The key to the IBM EDI solution, shown in Figure 1-10, is IBM WebSphere Data Interchange for Multiplatforms. It is the core of the solution and handles key EDI solution elements, such as translation and enveloping/deenveloping.

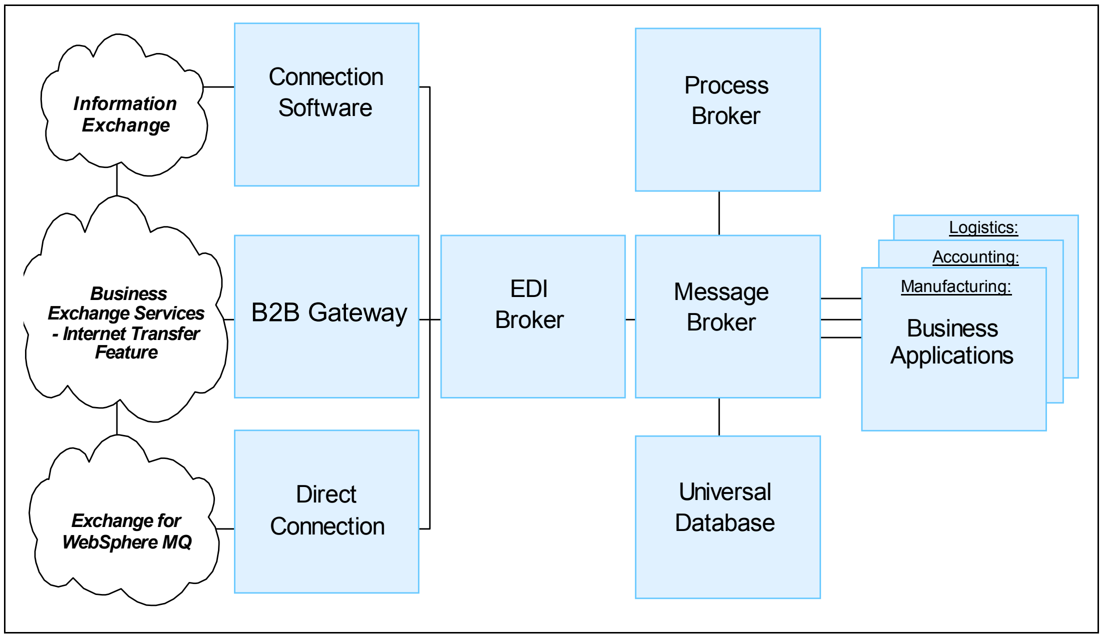


Figure 1-10 The IBM EDI solution

Working closely in partnership with IBM WebSphere MQ Integrator Broker, WebSphere Data Interchange can:

- Automate the distribution of EDI messages to and from all departments and trading partners.
- Transform EDI messages, on the fly, to the proper format for each existing application.
- Automatically redirect messages based on message content and system state.
- Track the flow of messages through systems for offline analysis and data mining.
- Reconfigure the system to respond to changing circumstances by adding or deleting applications.

1.5 Introducing WebSphere Data Interchange

This section provides an introduction to WebSphere Data Interchange. We discuss its main features and functions and describe the usage and role of its main components.

1.5.1 Features of WebSphere Data Interchange

WebSphere Data Interchange for Multiplatforms V3.2 (WDI V3.2) provides advanced translation, validation, and batched information exchange capabilities for Electronic Data Interchange (EDI) standards and for XML. WebSphere Data Interchange V3.2 electronically translates EDI format data, such as invoices, purchase orders, and billing forms, for exchange with trading partners. WebSphere Data Interchange V3.2 supports industry implementations of the ANSI X12, EDIFACT, VICS, UCS and Rail standards. Translation can take place between any combination of EDI, XML, or structured Application Data Format, which is a feature that is sometimes called any-to-any transformation. WebSphere Data Interchange V3.2 provides advanced data validation and standards compliance functions that allow the functional acknowledgments, defined by some standards, to be generated in response to inconsistencies in the data content. WebSphere Data Interchange V3.2 can be configured to both construct and deconstruct envelopes of EDI format data that contain batches of related EDI items such as invoices or purchase orders.

WebSphere Data Interchange V3.2 provides a dedicated GUI mapping tool, the WebSphere Data Interchange Client, which is optimized to build EDI, XML, and Application Data Format transformations. The WebSphere Data Interchange Client allows direct import of EDI standards definitions, Application Data Format structures and industry standards or user-defined XML DTDs for mapping and translation.

The WebSphere Data Interchange Client provides configuration and administration capabilities. Network profiles and Trading Partner profiles can all be managed via the client interface of WebSphere Data Interchange.

In addition to mapping and configuration, the client interface can be used for auditing and runtime support tasks. The client offers an interface to create reports on transactions and messages. Event logs and activity logs can be created and used as a way to analyze system behavior.

WebSphere Data Interchange V3.2 is available on the Windows® 2000, AIX®, and z/OS® platforms. The server component is available in two shapes. One way to use the translation engine is to start in a batch-type of mode driven by a command file. The command file typically contains actions that the translation engine has to perform on a number of files containing EDI messages. Typical actions include batching, enveloping, and deenveloping, sending and receiving files. Usually, the startup of the translation engine is controlled by some automation or scheduling tool.

Another way of launching the translation engine is to use the WDIAdapter program. You can configure WebSphere MQ in such a way that the WDIAdapter program is launched when a message arrives at a queue. The adapter program will then read this message and perform the translations that are required, as configured in WebSphere Data Interchange. The translated message can then be written to another WebSphere MQ queue.

WebSphere Data Interchange V3.2 supports integration with WebSphere MQ enabling inter-operation with a wide range of enterprise applications, business process engines such as the InterChange Server, information brokers such as WebSphere MQ Integrator, and ERP systems such as SAP R3. The reading and writing of WebSphere MQ messages can be performed in three different ways:

- Standard MQ messages with only a message descriptor.
- MQ messages with an MQRFH2 header. That header can contain an mcd folder, to indicate the message set, type and format, and it can contain additional information in the user folder to indicate receiver and sender information.
- MQ messages destined for JMS API clients.

WebSphere Data Interchange V3.2 provides for communication with trading partners via both value-added networks (VANs) or Internet B2B gateways by provision of an easy-to-use configurable interface which enables connection to leading VAN and Internet gateways. The WebSphere Business Integration - Connect offerings that provide AS1 and AS2 support and the IBM e-business hosting Expedite VAN gateway are two examples of supported gateways from IBM.

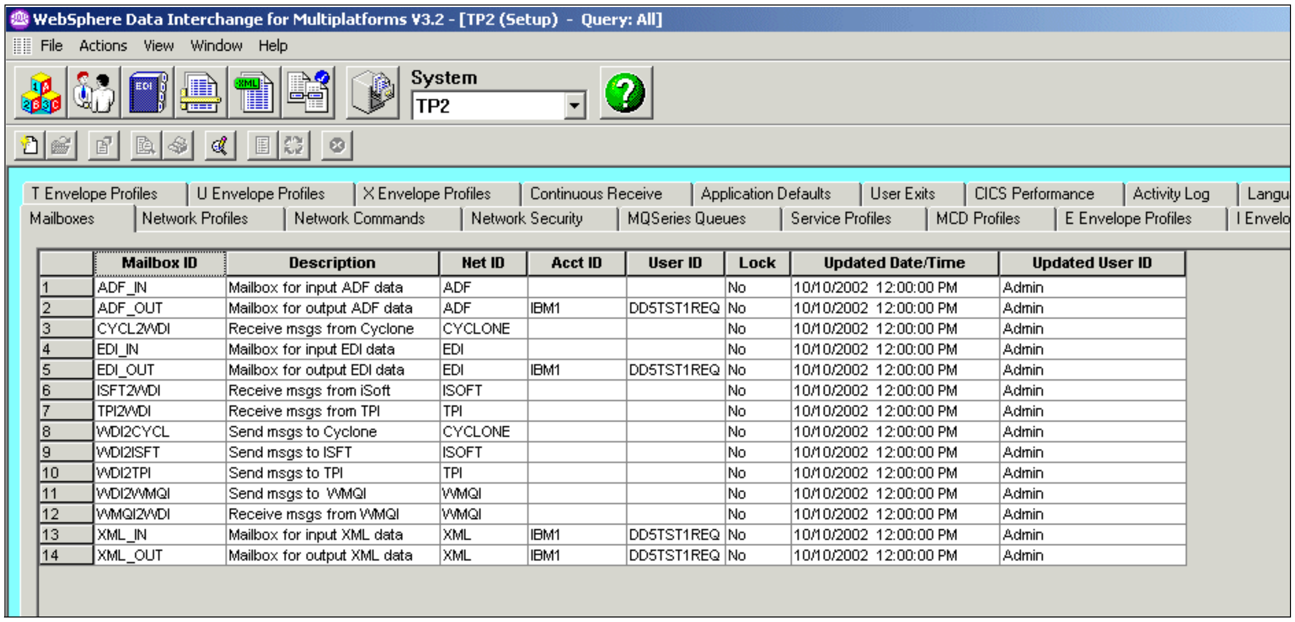
In the context of a typical enterprise integration architecture, WebSphere Data Interchange fulfills the role of an EDI broker that performs the specialist EDI validation, transformation, and exchange functions, and propagates the resulting transformed information either internally or externally. Internal propagation of transformed information may be via a message broker, a process broker, direct to the business applications, or through any combination of those, depending on the needs of enterprise. External propagation of transformed information or receipt of information may be through a specialized, dedicated VAN gateway, an Internet B2B gateway, directly to a trading partner, or through any combination of those interfaces, depending on the nature of the trading relationship between the enterprise and its trading partner.

There are certain concepts you should become familiar with before attempting to understand how a message is processed by WebSphere Data Interchange. Described below are the components of particular relevance:

- Mailbox profiles
- Network profiles
- WebSphere MQ-related artifacts
- Service profiles
- Trading Partners
- Maps
- Rules

1.5.2 Mailbox profiles

Mailbox profiles contain the information that WebSphere Data Interchange needs to identify the individuals and groups in your organization that receive documents to be translated. Each individual or group requires its own Mailbox profile. Figure 1-11 illustrates the default Mailbox profiles shipped with WebSphere Data Interchange.


WebSphere Data Interchange for Multiplatforms V3.2 - [TP2 (Setup) - Query: All]
File Actions View Window Help
System
TP2
T Envelope Profiles | U Envelope Profiles | X Envelope Profiles | Continuous Receive | Application Defaults | User Exits | CICS Performance | Activity Log | Langu
Mailboxes | Network Profiles | Network Commands | Network Security | MQSeries Queues | Service Profiles | MCD Profiles | E Envelope Profiles | I Envelo

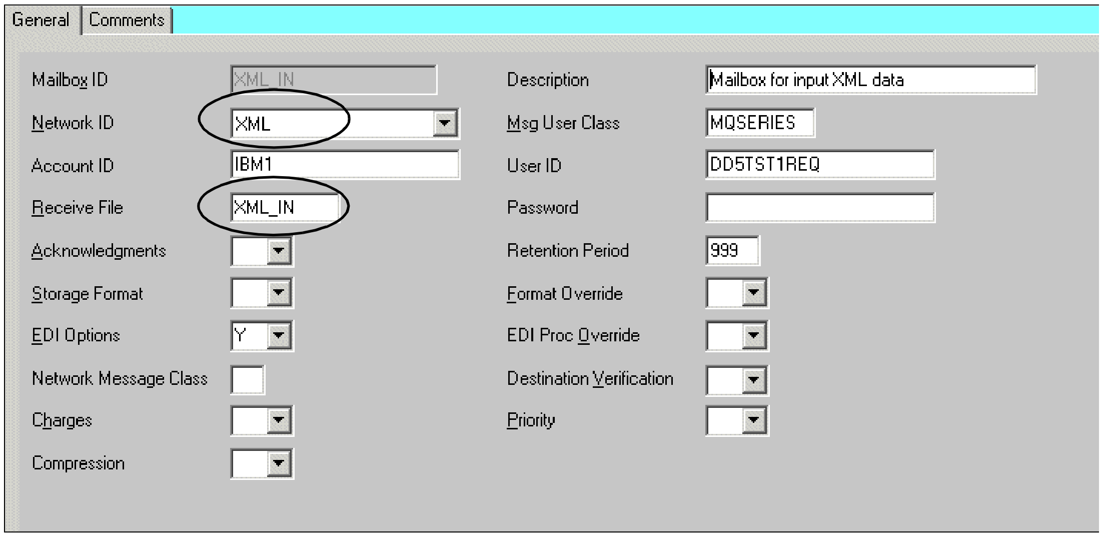
	Mailbox ID	Description	Net ID	Acct ID	User ID	Lock	Updated Date/Time	Updated User ID
1	ADF_IN	Mailbox for input ADF data	ADF			No	10/10/2002 12:00:00 PM	Admin
2	ADF_OUT	Mailbox for output ADF data	ADF	IBM1	DD5TST1REQ	No	10/10/2002 12:00:00 PM	Admin
3	CYCL2WDI	Receive msgs from Cyclone	CYCLONE			No	10/10/2002 12:00:00 PM	Admin
4	EDI_IN	Mailbox for input EDI data	EDI			No	10/10/2002 12:00:00 PM	Admin
5	EDI_OUT	Mailbox for output EDI data	EDI	IBM1	DD5TST1REQ	No	10/10/2002 12:00:00 PM	Admin
6	ISFT2WDI	Receive msgs from iSoft	ISOFT			No	10/10/2002 12:00:00 PM	Admin
7	TPI2WDI	Receive msgs from TPI	TPI			No	10/10/2002 12:00:00 PM	Admin
8	WDI2CYCL	Send msgs to Cyclone	CYCLONE			No	10/10/2002 12:00:00 PM	Admin
9	WDI2ISFT	Send msgs to ISFT	ISOFT			No	10/10/2002 12:00:00 PM	Admin
10	WDI2TPI	Send msgs to TPI	TPI			No	10/10/2002 12:00:00 PM	Admin
11	WDI2WMQI	Send msgs to WMQI	WMQI			No	10/10/2002 12:00:00 PM	Admin
12	WMQI2WDI	Receive msgs from WMQI	WMQI			No	10/10/2002 12:00:00 PM	Admin
13	XML_IN	Mailbox for input XML data	XML	IBM1	DD5TST1REQ	No	10/10/2002 12:00:00 PM	Admin
14	XML_OUT	Mailbox for output XML data	XML	IBM1	DD5TST1REQ	No	10/10/2002 12:00:00 PM	Admin

Figure 1-11 Mailbox configuration window

Of particular importance in the Mailbox profile settings are the Network ID and Receive File details. The Network ID identifies which logical network within WebSphere Data Interchange is to be used to send or receive information. The Network ID is selected from the list of available Network profiles available in WebSphere Data Interchange. The Receive File field defines the logical file name expected to be received by this Mailbox profile.

A mailbox can be something logical, referring to a file or an MQ queue, or it can refer to a mailbox provided by a VAN. In that case, the attributes Account ID, User ID and Password are the actual account ID, user ID, and password associated with your VAN account.

Figure 1-12 illustrates the default settings of the XML_IN Mailbox profile.



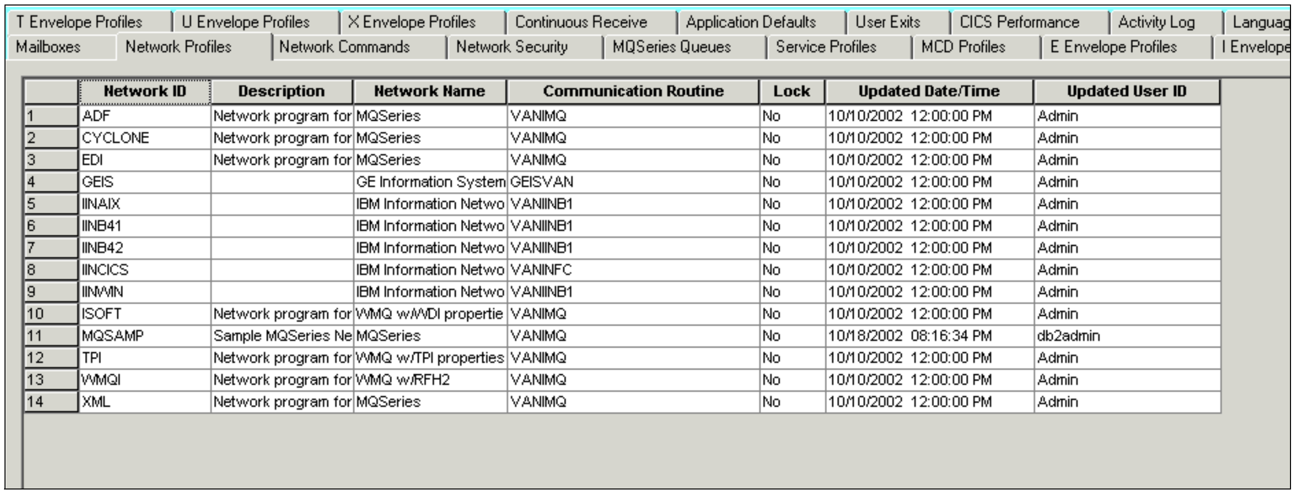
Mailbox ID	XML_IN	Description	Mailbox for input XML data
Network ID	XML	Msg User Class	MQSERIES
Account ID	IBM1	User ID	DD5TST1REQ
Receive File	XML_IN	Password	
Acknowledgments		Retention Period	999
Storage Format		Format Override	
EDI Options	Y	EDI Proc Override	
Network Message Class		Destination Verification	
Charges		Priority	
Compression			

Figure 1-12 Settings of a Mailbox profile

Here we see that the selected Network ID is `XML` and Receive File is set to `XML_IN`. When using WebSphere MQ as the communication between trading partners or applications, all other details on this window are unused. The Account ID, User ID, Password and Msg User Class fields come into play when required by the network, for example, when using IBM Information Exchange.

1.5.3 Network profiles

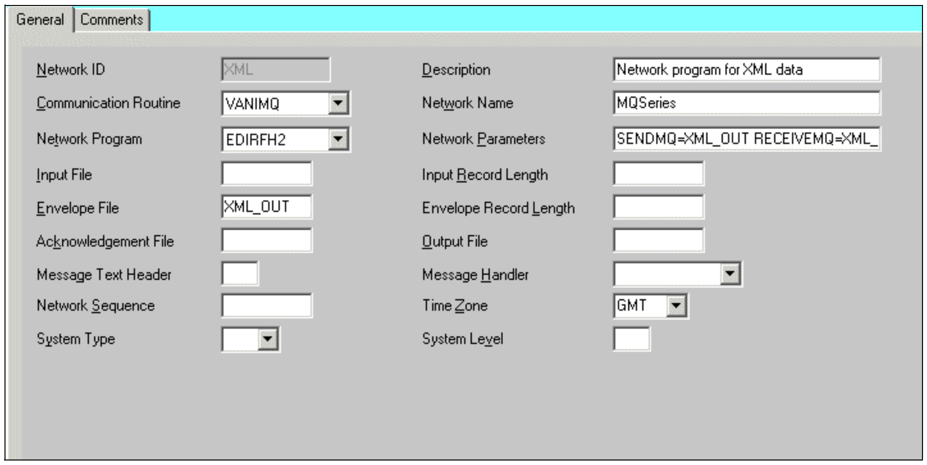
Network profiles define for WebSphere Data Interchange the characteristics of the networks you use for communications with trading partners. WebSphere Data Interchange is shipped with the Network profiles required to communicate with several major networks. In Figure 1-13, we see the default Network profiles shipped with WebSphere Data Interchange.



	Network ID	Description	Network Name	Communication Routine	Lock	Updated Date/Time	Updated User ID
1	ADF	Network program for	MQSeries	VANIMQ	No	10/10/2002 12:00:00 PM	Admin
2	CYCLONE	Network program for	MQSeries	VANIMQ	No	10/10/2002 12:00:00 PM	Admin
3	EDI	Network program for	MQSeries	VANIMQ	No	10/10/2002 12:00:00 PM	Admin
4	GEIS		GE Information System	GEISVAN	No	10/10/2002 12:00:00 PM	Admin
5	IINAIX		IBM Information Netwo	VANIINB1	No	10/10/2002 12:00:00 PM	Admin
6	IINB41		IBM Information Netwo	VANIINB1	No	10/10/2002 12:00:00 PM	Admin
7	IINB42		IBM Information Netwo	VANIINB1	No	10/10/2002 12:00:00 PM	Admin
8	IINCICS		IBM Information Netwo	VANIINFC	No	10/10/2002 12:00:00 PM	Admin
9	IINWIN		IBM Information Netwo	VANIINB1	No	10/10/2002 12:00:00 PM	Admin
10	ISOFT	Network program for	WMQ w/WDI propertie	VANIMQ	No	10/10/2002 12:00:00 PM	Admin
11	MQSAMP	Sample MQSeries Ne	MQSeries	VANIMQ	No	10/18/2002 08:16:34 PM	db2admin
12	TPI	Network program for	WMQ w/TPI properties	VANIMQ	No	10/10/2002 12:00:00 PM	Admin
13	WMQI	Network program for	WMQ w/RFH2	VANIMQ	No	10/10/2002 12:00:00 PM	Admin
14	XML	Network program for	MQSeries	VANIMQ	No	10/10/2002 12:00:00 PM	Admin

Figure 1-13 Default Network profiles

If we look at the XML profile in more detail, we can examine the details of importance. Figure 1-14 shows the default details of the XML Network profile.



Network ID	XML	Description	Network program for XML data
Communication Routine	VANIMQ	Network Name	MQSeries
Network Program	EDIRFH2	Network Parameters	SENDMQ=XML_OUT RECEIVEMQ=XML_
Input File		Input Record Length	
Envelope File	XML_OUT	Envelope Record Length	
Acknowledgement File		Output File	
Message Text Header		Message Handler	
Network Sequence		Time Zone	GMT
System Type		System Level	

Figure 1-14 Settings of a Network profile

Network program `EDIRFH2` is specified along with communications routine `VANIMQ`, so the VANIMQ program will be called to actually process the WebSphere MQ queue specified by the Network Parameters. This network program and communications routine are shipped with WebSphere Data Interchange and are used when processing from an WebSphere MQ queue. The network program EDIRFH2 is used when you expect to process or generate an

MQRFH2 header. If you do not have this requirement and you plan on using standard MQ messages, you should use the network program EDIMQSR. There is a third MQ-oriented network program, called EDICYCL, which is used when interacting with JMS clients. Examples of these different network programs for WebSphere MQ can be found in Chapter 2, “Implementing iSoft P2PAgent” on page 49 and in Chapter 3, “Implementing multi-product AS/2 communication with trading partners” on page 101.

The logical names of the inbound and outbound queues to be processed are kept in the field named Network Parameters. The format of the field is:

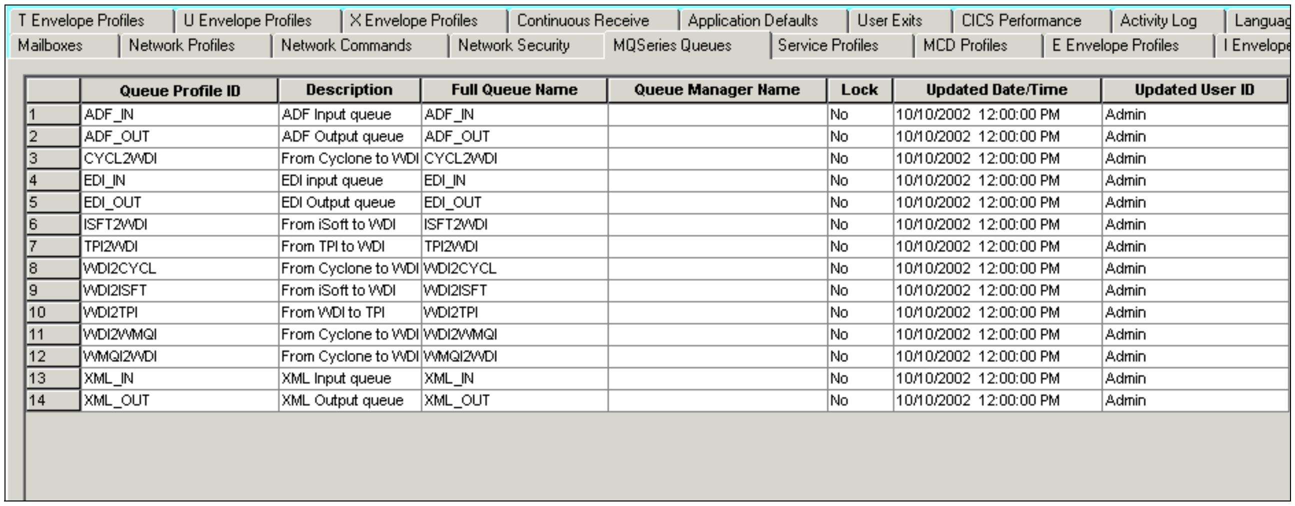
```
SENDMQ = name_of_the_queue_for_outbound_data RECEIVEMQ =
name_of_the_queue_for_inbound_data
```

The actual details of the queue, such as the queue manager name, full queue name, whether destructive MQGET operations should be performed, and so on are specified in a different profile called the MQSeries® Queue profile. Every WebSphere MQ queue that WebSphere Data Interchange accesses must be described within WebSphere Data Interchange by an MQSeries Queue profile.

The Envelope File field is an optional one. When translating documents, the output documents are written to this file, here XML_IN. Documents are then read from this file when sending to a trading partner. When receiving documents from a trading partner, the documents will be written to this file. The WebSphere Data Interchange will read the documents from this file to translate them.

1.5.4 WebSphere MQ-related artifacts

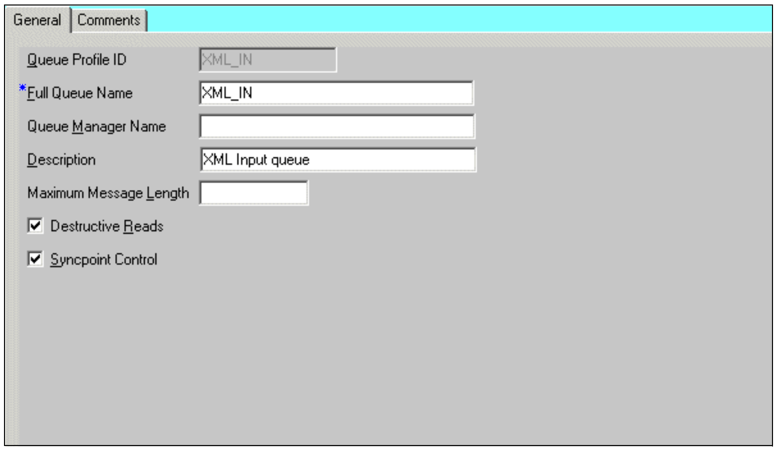
Of particular importance to the user is an understanding of the WebSphere MQ-related artifacts that exist in WebSphere Data Interchange and how they are related to MQ concepts such as queues and queue managers. The interface with each of these MQ-related artifacts can be found in the Setup area of the WebSphere Data Interchange client. The MQSeries Queues tab selected below contains WebSphere Data Interchange’s definitions of actual queues defined in WebSphere MQ.



	Queue Profile ID	Description	Full Queue Name	Queue Manager Name	Lock	Updated Date/Time	Updated User ID
1	ADF_IN	ADF Input queue	ADF_IN		No	10/10/2002 12:00:00 PM	Admin
2	ADF_OUT	ADF Output queue	ADF_OUT		No	10/10/2002 12:00:00 PM	Admin
3	CYCL2WDI	From Cyclone to WDI	CYCL2WDI		No	10/10/2002 12:00:00 PM	Admin
4	EDI_IN	EDI input queue	EDI_IN		No	10/10/2002 12:00:00 PM	Admin
5	EDI_OUT	EDI Output queue	EDI_OUT		No	10/10/2002 12:00:00 PM	Admin
6	ISFT2WDI	From iSoft to WDI	ISFT2WDI		No	10/10/2002 12:00:00 PM	Admin
7	TPI2WDI	From TPI to WDI	TPI2WDI		No	10/10/2002 12:00:00 PM	Admin
8	WDI2CYCL	From Cyclone to WDI	WDI2CYCL		No	10/10/2002 12:00:00 PM	Admin
9	WDI2ISFT	From iSoft to WDI	WDI2ISFT		No	10/10/2002 12:00:00 PM	Admin
10	WDI2TPI	From WDI to TPI	WDI2TPI		No	10/10/2002 12:00:00 PM	Admin
11	WDI2WMQI	From Cyclone to WDI	WDI2WMQI		No	10/10/2002 12:00:00 PM	Admin
12	WMQI2WDI	From Cyclone to WDI	WMQI2WDI		No	10/10/2002 12:00:00 PM	Admin
13	XML_IN	XML Input queue	XML_IN		No	10/10/2002 12:00:00 PM	Admin
14	XML_OUT	XML Output queue	XML_OUT		No	10/10/2002 12:00:00 PM	Admin

Figure 1-15 MQSeries Queue profiles

When we look at the details of one of these queue definitions, we can see that WebSphere Data Interchange has assigned a logical queue name to the physical WebSphere MQ queue and defined the queue manager where this queue resides.



Queue Profile ID	XML_IN
*Full Queue Name	XML_IN
Queue Manager Name	
Description	XML Input queue
Maximum Message Length	
☑ Destructive Reads	
☑ Syncpoint Control	

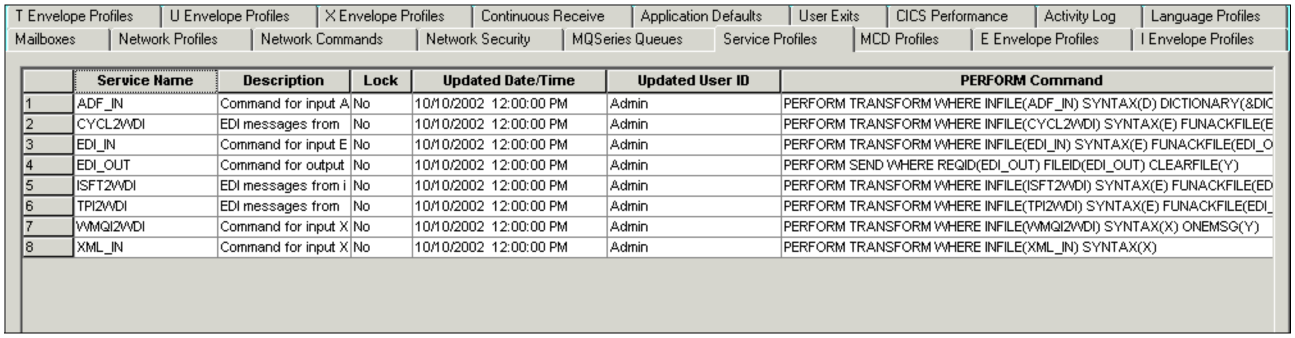
Figure 1-16 Details of an MQSeries Queue profile

In Figure 1-16, the Queue Profile ID XML_IN corresponds to Full Queue Name XML_IN (the actual WebSphere MQ queue). The Queue Manager field is not specified since this queue resides on the default queue manger. If the queue isn't on the default queue manager then you would specify the name of the queue manager in the Queue Manager field above.

WebSphere Data Interchange uses these MQSeries Queue profiles once called from the Network Parameters field in the Network profile.

1.5.5 Service profiles

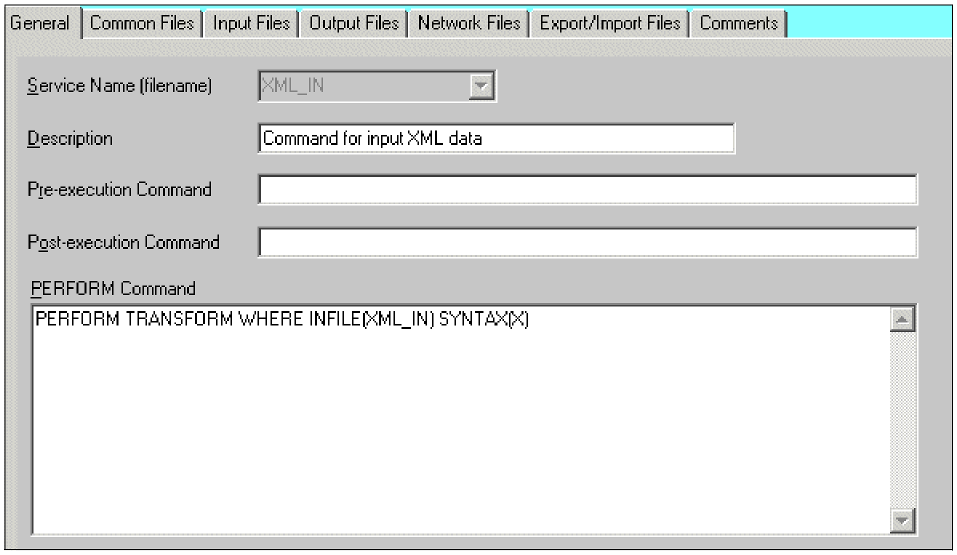
The purpose of the Service profile is to allow you to enter a utility command and all the files that will be used during execution of that command. There are specific fields for fixed names, such as the print file (PRTFILE), and short name and long name pairs for times when both the short and long names are user-defined, such as input and output files. Below are the default Service profiles shipped with WebSphere Data Interchange.



	Service Name	Description	Lock	Updated Date/Time	Updated User ID	PERFORM Command
1	ADF_IN	Command for input A	No	10/10/2002 12:00:00 PM	Admin	PERFORM TRANSFORM WHERE INFILE(ADF_IN) SYNTAX(D) DICTIONARY(&DIC
2	CYCL2WDI	EDI messages from	No	10/10/2002 12:00:00 PM	Admin	PERFORM TRANSFORM WHERE INFILE(CYCL2WDI) SYNTAX(E) FUNACKFILE(E
3	EDI_IN	Command for input E	No	10/10/2002 12:00:00 PM	Admin	PERFORM TRANSFORM WHERE INFILE(EDI_IN) SYNTAX(E) FUNACKFILE(EDI_O
4	EDI_OUT	Command for output	No	10/10/2002 12:00:00 PM	Admin	PERFORM SEND WHERE REQID(EDI_OUT) FILEID(EDI_OUT) CLEARFILE(Y)
5	ISFT2WDI	EDI messages from i	No	10/10/2002 12:00:00 PM	Admin	PERFORM TRANSFORM WHERE INFILE(ISFT2WDI) SYNTAX(E) FUNACKFILE(ED
6	TPI2WDI	EDI messages from	No	10/10/2002 12:00:00 PM	Admin	PERFORM TRANSFORM WHERE INFILE(TPI2WDI) SYNTAX(E) FUNACKFILE(EDI_
7	WMQI2WDI	Command for input X	No	10/10/2002 12:00:00 PM	Admin	PERFORM TRANSFORM WHERE INFILE(WMQI2WDI) SYNTAX(X) ONEMSG(Y)
8	XML_IN	Command for input X	No	10/10/2002 12:00:00 PM	Admin	PERFORM TRANSFORM WHERE INFILE(XML_IN) SYNTAX(X)

Figure 1-17 Default Service profiles

In the General tab of the XML_IN Service profile, we can see the default utility command provided by WebSphere Data Interchange.



General | Common Files | Input Files | Output Files | Network Files | Export/Import Files | Comments

Service Name (filename)	XML_IN
Description	Command for input XML data
Pre-execution Command	
Post-execution Command	

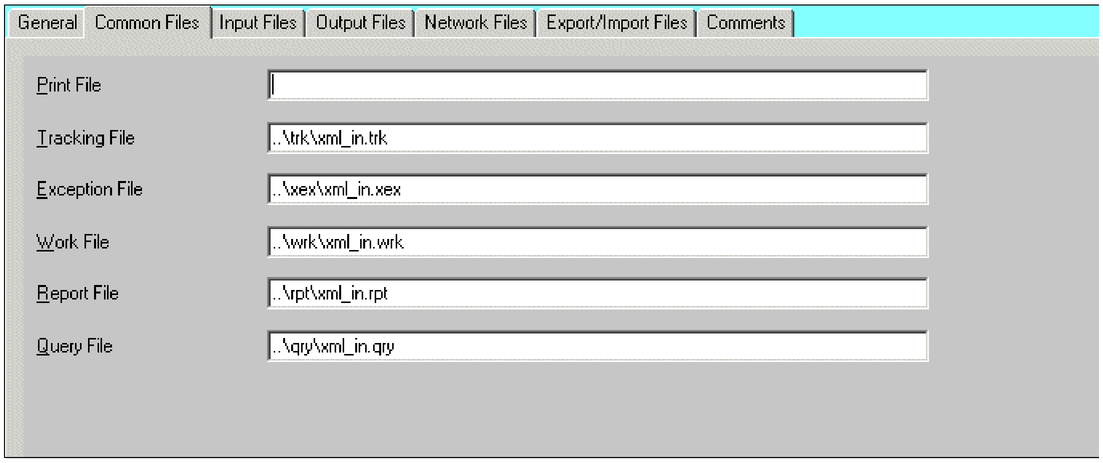
PERFORM Command

```
PERFORM TRANSFORM WHERE INFILE(XML_IN) SYNTAX(X)
```

Figure 1-18   Detailed view of the settings of a Service profile

The XML_IN parameters passed to INFILE tell WebSphere Data Interchange that this is the file to perform translation on (to TRANSFORM). This corresponds to the Receive File detail we specified earlier in the Mailbox profile. The X parameter passed to SYNTAX tells WebSphere Data Interchange to expect XML_IN to be in an XML format.

The Common Files tab outlines the default locations for each of the file structures used by WebSphere Data Interchange to provide the user with information on the TRANSFORM process. These are discussed in more detail in Chapter 2, “Implementing iSoft P2PAgent” on page 49.



General | Common Files | Input Files | Output Files | Network Files | Export/Import Files | Comments

Print File	
Tracking File	..\trk\xml_in.trk
Exception File	..\xex\xml_in.xex
Work File	..\wrk\xml_in.wrk
Report File	..\rpt\xml_in.rpt
Query File	..\qry\xml_in.qry

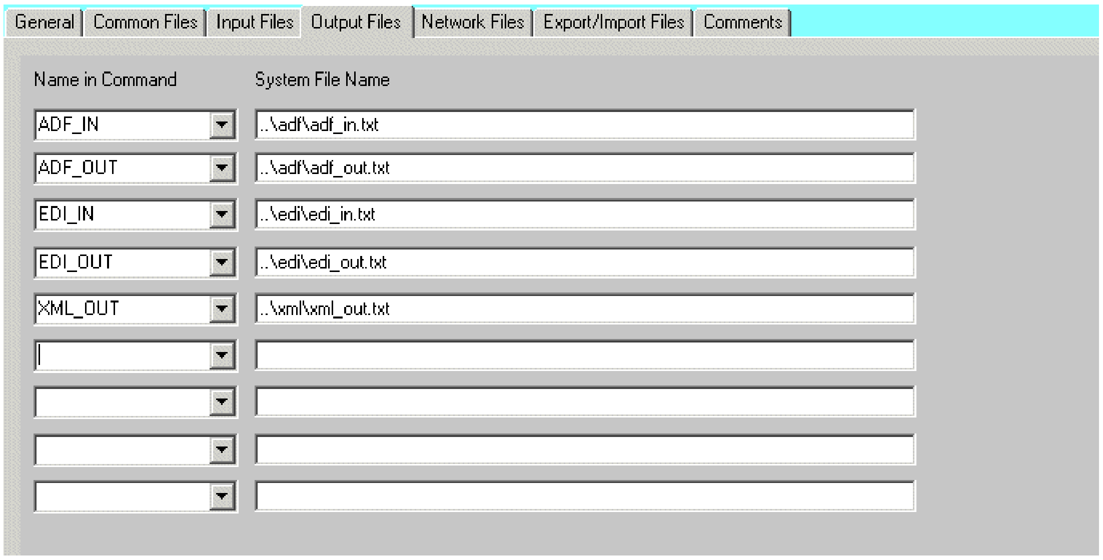
Figure 1-19   The Common Files tab for a Service profile

The Input Files tab is typically left blank. This is only used if the input for the utility command is different from the file that initially triggered the process. This is not a common scenario.

The Output Files tab associates a physical file location for the logical output of the utility command. For example, let’s issue the following command in the General tab:

```
PERFORM TRANSFORM WHERE INFILE(XML_IN) SYNTAX(X) OUTFILE(XML_OUT)
```

The output of the utility command will reside in the physical file location associated with XML_OUT in the Output Files tab.



Name in Command	System File Name
ADF_IN	..\adf\adf_in.txt
ADF_OUT	..\adf\adf_out.txt
EDI_IN	..\edi\edi_in.txt
EDI_OUT	..\edi\edi_out.txt
XML_OUT	..\xml\xml_out.txt

Figure 1-20 The Output Files tab of a Service profile

In Figure 1-20, we see that the logical file name XML_OUT is mapped to the physical file name ..\xml\xml_out.txt.

Note that the name you give to the Service profile is its logical name. If another command writes information to the file associated with this logical name, the **PERFORM** command is executed after that command completes, connecting the commands together. This is known as *command chaining*.

The Network Files tab allows the user to enter details of files required for communication by the network program. Typically, these are used if Expedite is being used as the communication channel between WebSphere Data Interchange and Information Exchange. If using WebSphere MQ-to-WebSphere MQ communications, these fields can remain unused.

1.5.6 Trading Partner profiles

Trading Partner profiles are maintained in WebSphere Data Interchange under the Trading Partner icon. This icon is shown in Figure 1-21.

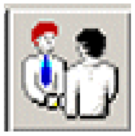
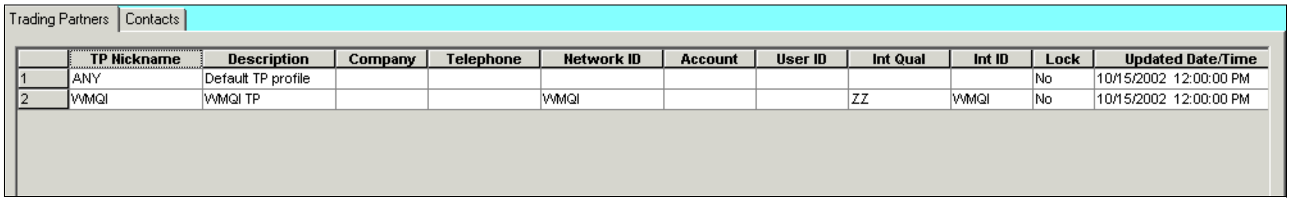


Figure 1-21 Icon to access Trading Partner profiles

WebSphere Data Interchange ships with two sample Trading Partner profiles.

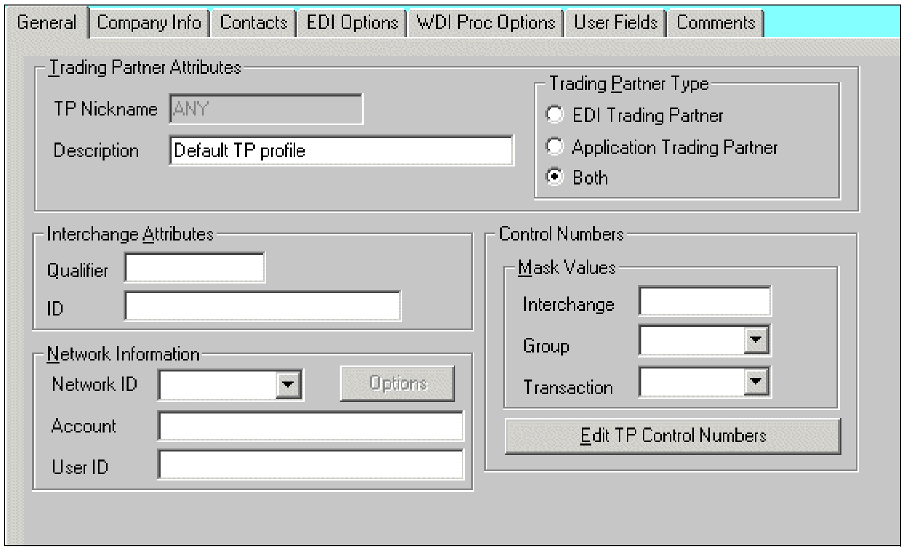


	TP Nickname	Description	Company	Telephone	Network ID	Account	User ID	Int Qual	Int ID	Lock	Updated Date/Time
1	ANY	Default TP profile								No	10/15/2002 12:00:00 PM
2	WMQI	WMQI TP			WMQI			ZZ	WMQI	No	10/15/2002 12:00:00 PM

Figure 1-22 Default Trading Partner profiles

The ANY trading partner is a useful template that can be used when simulating a data transformation scenario where the sender and receiver details are of little importance.

The General tab of the ANY Trading Partner profile can be seen in Figure 1-23.



Trading Partner Attributes

TP Nickname	ANY
Description	Default TP profile

Trading Partner Type

EDI Trading Partner / Application Trading Partner / [✓ Both]

Interchange Attributes

Qualifier	
ID	

Network Information

Network ID	
Account	
User ID	

Control Numbers

Mask Values

Interchange	
Group	
Transaction	

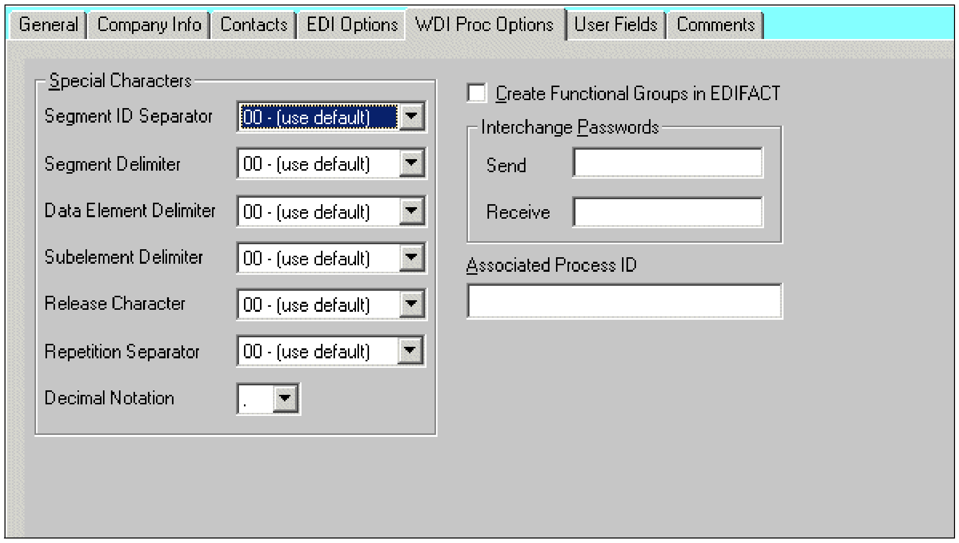
Figure 1-23 Settings of a Trading Partner profile

In a simple ANY to ANY scenario over WebSphere MQ, the only field of importance is the Network ID field. This determines which WebSphere Data Interchange Network profile is to be used with this trading partner. This should correspond to the network ID selected in the Mailbox profile as illustrated earlier. If Information Exchange is being used, the user is required to enter an Information Exchange Account and User ID. It will also be necessary to identify the Interchange Attributes by entering the Trading Partner's Qualifier and ID. The ID in this instance could be an alias, depending on what you have defined in Information Exchange.

In WebSphere Data Interchange's view of the world, there are two type of trading partners: application (or internal) trading partners and EDI (or external) trading partners. An application trading partner represents a business entity within the customer's enterprise. An external trading partner is a business entity that the user's enterprise does business with via EDI. Both are represented by a Trading Partner profile. What differentiates the two of them is the trading partner type field on the General tab of the Trading Partner Profile editor. In Figure 1-23, we see that ANY is defined as being both an EDI and application trading partner.

WebSphere Data Interchange provides tabs to store more specific details on the trading partner. For example, space is provided for company information and contacts. These are not required, however.

The only other tab of importance in this quick beginnings setup is the WDI Proc Options tab. Here the user is allowed to specify the delimiters used by WebSphere Data Interchange. Figure 1-24 shows the default settings of ANY.



Special Characters

Segment ID Separator	00 - (use default)
Segment Delimiter	00 - (use default)
Data Element Delimiter	00 - (use default)
Subelement Delimiter	00 - (use default)
Release Character	00 - (use default)
Repetition Separator	00 - (use default)
Decimal Notation	.

☐ Create Functional Groups in EDIFACT

Interchange Passwords

Send

Receive

Associated Process ID

Figure 1-24 The WDI Proc Options tab of a Trading Partner profile

1.5.7 Concepts of the mapping editor

A data transformation map is a set of mapping instructions that describes how to translate data from a source document into a target document. The order in which the mapping instructions occur can be based on the source document (source-based mapping) or the target document (target-based mapping). Both the source and target documents can be one of several supported document types.

In this section, we discuss a number of techniques available in WebSphere Data Interchange when creating a data transformation map. To open the Map Editor, use the button shown in Figure 1-25 on the tool bar.

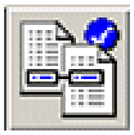


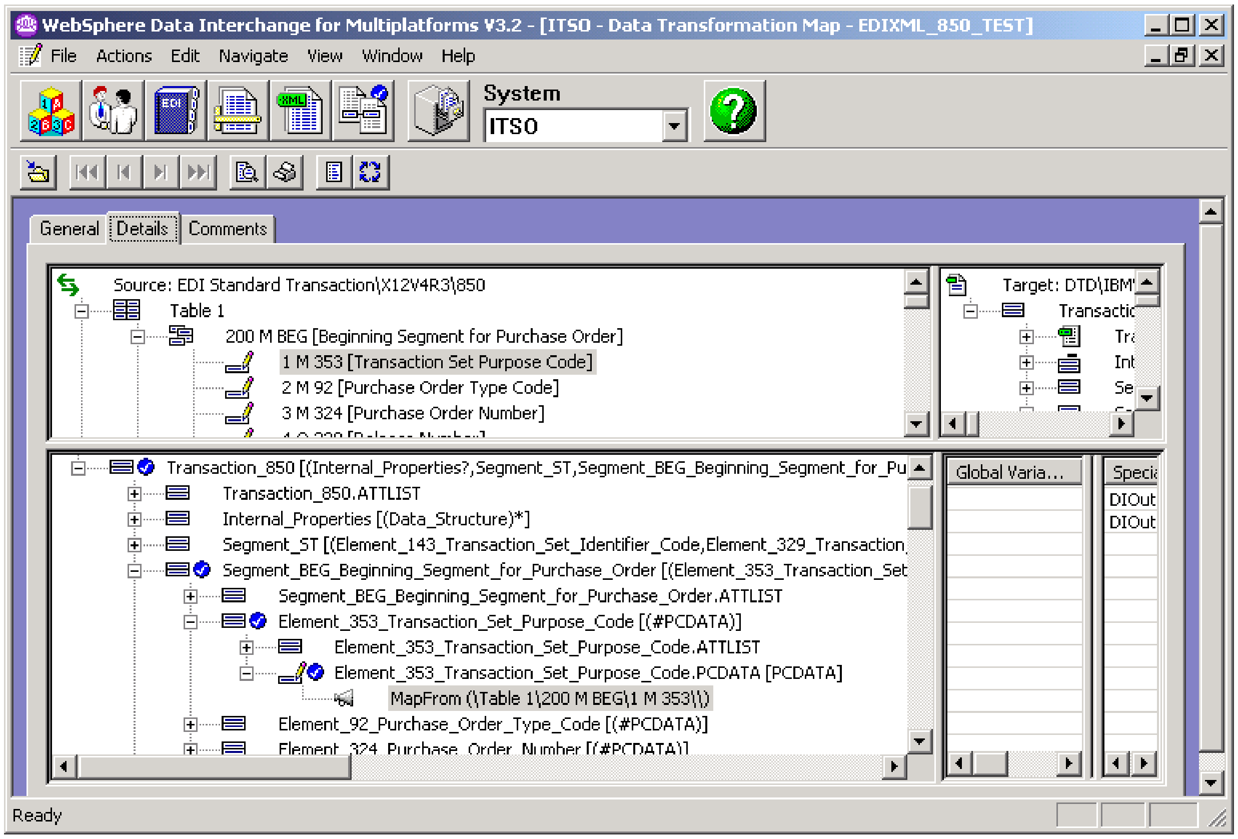
Figure 1-25 Tool bar button to open the Map Editor

Basic mapping by drag-and-drop

The Details tab in the Map Editor (Figure 1-26 on page 24) allows you to perform drag and drop mapping on your documents. The top left pane in the window displays the source document definition, and the top right pane displays the target document definition. The lower left pane is the Mapping Command window pane, and the lower right pane is the Variables window pane, which includes the lists for Global, Local, and Special Variables.

Click the element you want to map on the top left side of the window. While holding down the mouse button, drag it to the corresponding element in the target document definition on the top right pane. When you have dragged the element to the right side of the window over the element with which you want to associate it, that component becomes highlighted. Release the mouse button.

In a target-based map, the result of this action will be a MapFrom() command. Refer to the example in the Mapping Command window below.


WebSphere Data Interchange for Multiplatforms V3.2 - [ITSO - Data Transformation Map - EDIXML_850_TEST]
File Actions Edit Navigate View Window Help
System
ITSO
General
Details
Comments
Ready

```
Source: EDI Standard Transaction\X12V4R3\850
  Table 1
    200 M BEG [Beginning Segment for Purchase Order]
      1 M 353 [Transaction Set Purpose Code]
      2 M 92 [Purchase Order Type Code]
      3 M 324 [Purchase Order Number]

Target: DTD\IBM'
  Transactic
    Tra
    Int
    Se

Transaction_850 [(Internal_Properties?,Segment_ST,Segment_BEG_Beginning_Segment_for_Pu
  Transaction_850.ATTLIST
  Internal_Properties [(Data_Structure)*]
  Segment_ST [(Element_143_Transaction_Set_Identifier_Code,Element_329_Transaction_
  Segment_BEG_Beginning_Segment_for_Purchase_Order [(Element_353_Transaction_Set
    Segment_BEG_Beginning_Segment_for_Purchase_Order.ATTLIST
    Element_353_Transaction_Set_Purpose_Code [(#PCDATA)]
      Element_353_Transaction_Set_Purpose_Code.ATTLIST
      Element_353_Transaction_Set_Purpose_Code.PCDATA [PCDATA]
        MapFrom (\Table 1\200 M BEG\1 M 353\)
    Element_92_Purchase_Order_Type_Code [(#PCDATA)]

Global Varia...
Speci
DIOut
DIOut
```

Figure 1-26 The different panes of the Mapping Editor

Mapping elements by assignment

A value can be assigned to any variable or any simple element in the target document definition. This is accomplished using an assignment statement.

One method of creating an assignment statement is to right-click a node where you want to insert the statement. In the context menu that is displayed, select **Insert within -> Command -> Assignment**.

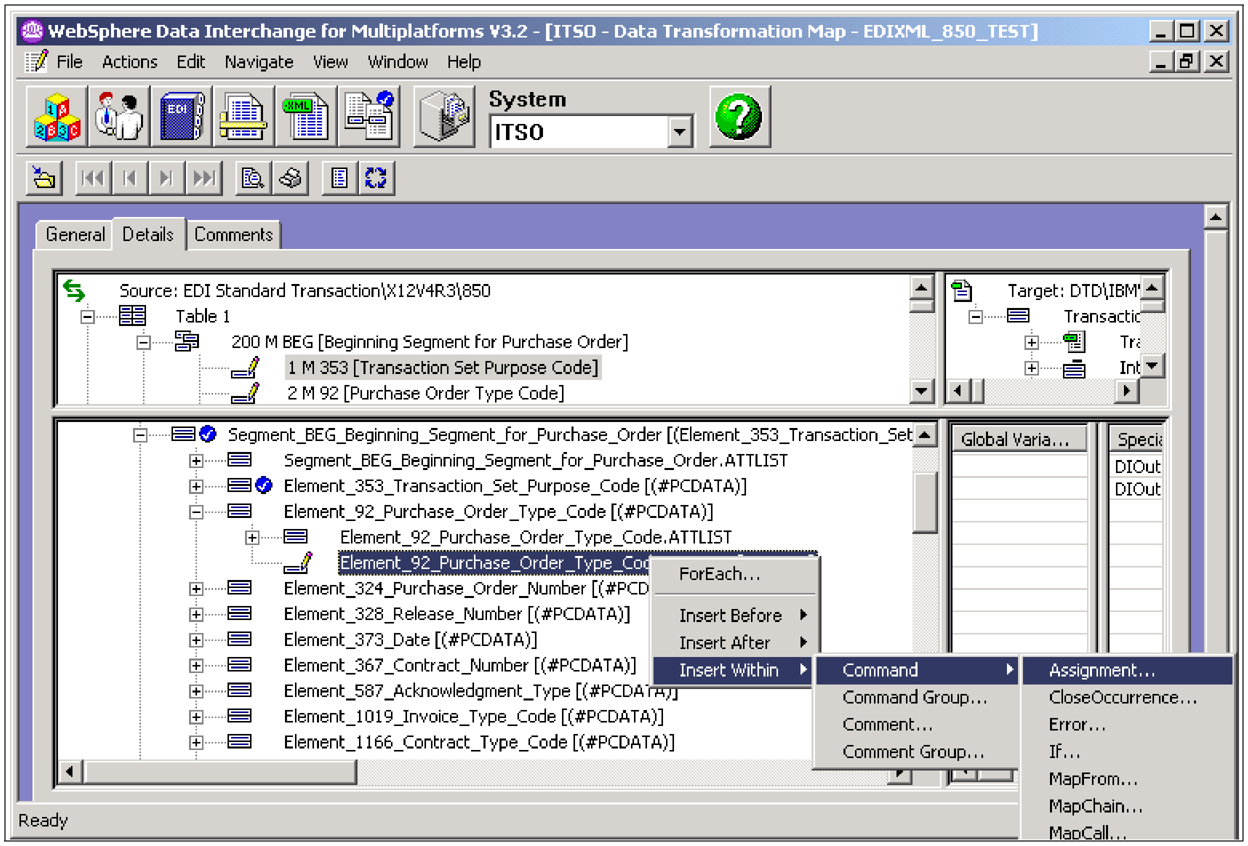

WebSphere Data Interchange for Multiplatforms V3.2 - [ITSO - Data Transformation Map - EDIXML_850_TEST]
File Actions Edit Navigate View Window Help
System
ITSO
General Details Comments
Source: EDI Standard Transaction\X12V4R3\850
Table 1
200 M BEG [Beginning Segment for Purchase Order]
1 M 353 [Transaction Set Purpose Code]
2 M 92 [Purchase Order Type Code]
Target: DTD\IBM
Segment_BEG_Beginning_Segment_for_Purchase_Order [(Element_353_Transaction_Set
Segment_BEG_Beginning_Segment_for_Purchase_Order.ATTLIST
Element_353_Transaction_Set_Purpose_Code [(#PCDATA)]
Element_92_Purchase_Order_Type_Code [(#PCDATA)]
Element_92_Purchase_Order_Type_Code.ATTLIST
Element_324_Purchase_Order_Number [(#PCD
Element_328_Release_Number [(#PCDATA)]
Element_373_Date [(#PCDATA)]
Element_367_Contract_Number [(#PCDATA)]
Element_587_Acknowledgment_Type [(#PCDATA)]
Element_1019_Invoice_Type_Code [(#PCDATA)]
Element_1166_Contract_Type_Code [(#PCDATA)]
ForEach...
Insert Before
Insert After
Insert Within
Command
Command Group...
Comment...
Comment Group...
Assignment...
CloseOccurrence...
Error...
If...
MapFrom...
MapChain...
MapCall...
Global Varia...
Speci
DIOut
DIOut
Ready

Figure 1-27 Adding an assignment to a map

The WebSphere Data Interchange Client then displays the Mapping Command Editor (Figure 1-28).

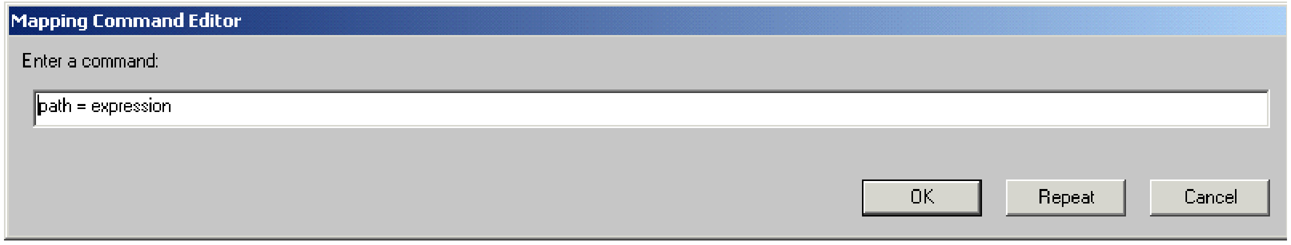
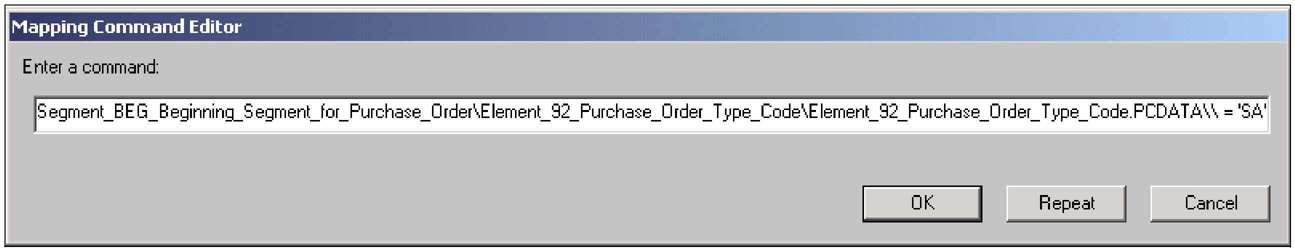


Figure 1-28 Mapping command editor

Click the element to which you want to assign a value on the top right side of the window shown in Figure 1-27. While holding down the mouse button, drag it over the path in the Mapping Command Editor shown in Figure 1-28. To assign a value to the target element, replace `expression` with a value. Be sure to place this value in single quotes. An example of an assignment completed using this method is shown in Figure 1-29 on page 26.



```
Segment_BEG_Beginning_Segment_for_Purchase_Order\Element_92_Purchase_Order_Type_Code\Element_92_Purchase_Order_Type_Code.PCDATA\\ = 'SA'
```

Figure 1-29 Completed assignment in the Mapping Command Editor

Another method of creating an assignment statement is to drag a simple element from the source document definition or the target document definition and drop it on a variable. This will create an assignment command at the appropriate position within the Mapping Command window. The variable will then hold the value of the source element.

Using variables when building maps

Map variables are used like variables in any programming language. They are an integral part of the WebSphere Data Interchange mapping command language. Variables are used to hold and manipulate values assigned to them during translation. They are used in mapping commands and functions, commonly within expressions. WebSphere Data Interchange supports three types of variables: Local Variables, Global Variables, and Special Variables.

Local Variables are unique to the map in which they are defined. A Local Variable must be defined to a map before it can be used in that map. Local Variables have a scope of document or loop. During translation, Local Variables defined with a scope of document will be created at the start of every document and deleted at the end of the document. Variables defined with a scope of loop will be created and initialized whenever a new loop iteration is started, and destroyed at the end of each loop iteration.

Global Variables are similar to Local Variables, but they are not unique to any map; they are shared across data transformation maps, validation maps, and functional acknowledgement maps.

Special Variables are a group of predefined variables used by WebSphere Data Interchange. They function much like Local Variables or Global Variables, but they each have a special purpose. A user can view properties of a Special Variable, but no changes can be made. Special Variable names will always start with “DI”.

An example of such a variable is DIPROLOG, which holds the XML declaration for an incoming XML document.

A variable is created in WebSphere Data Interchange by right-clicking below the variable type in the variable window. In Figure 1-30 on page 27, we are creating a new local variable.

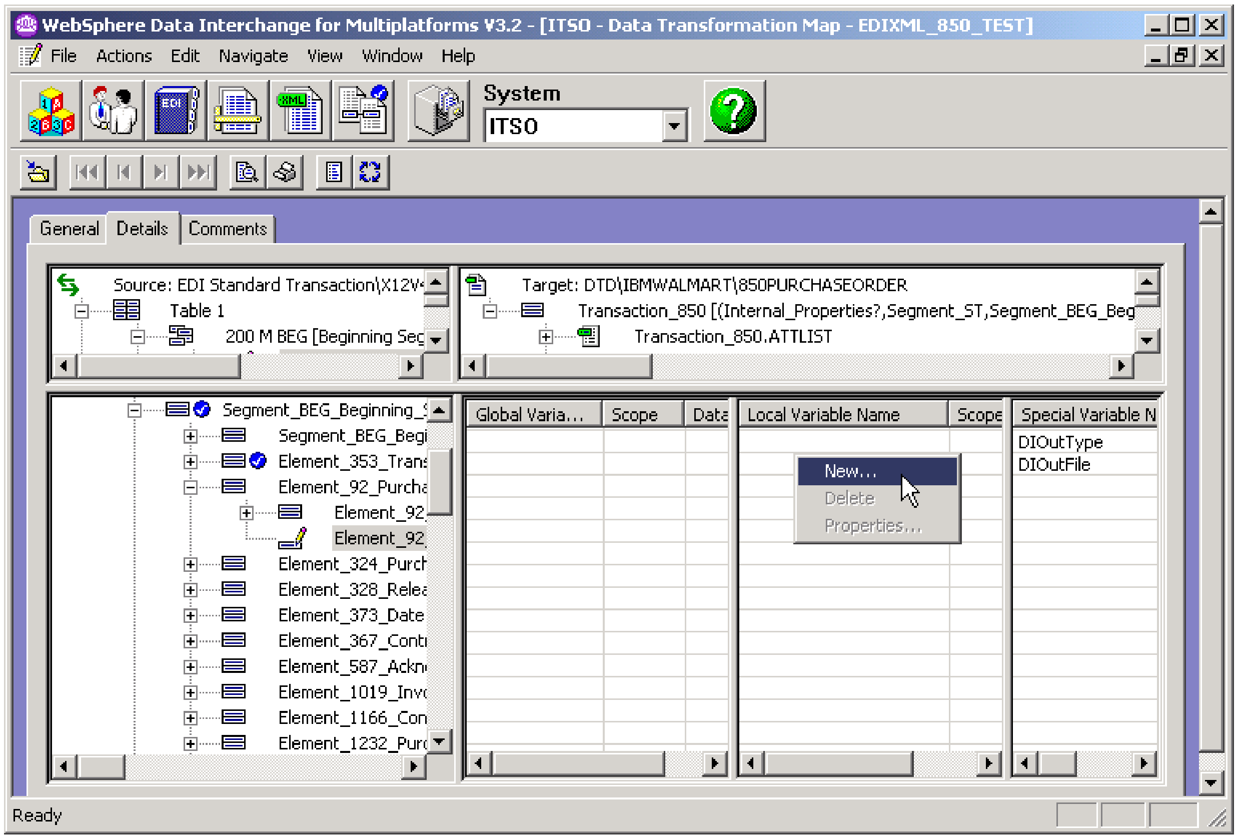


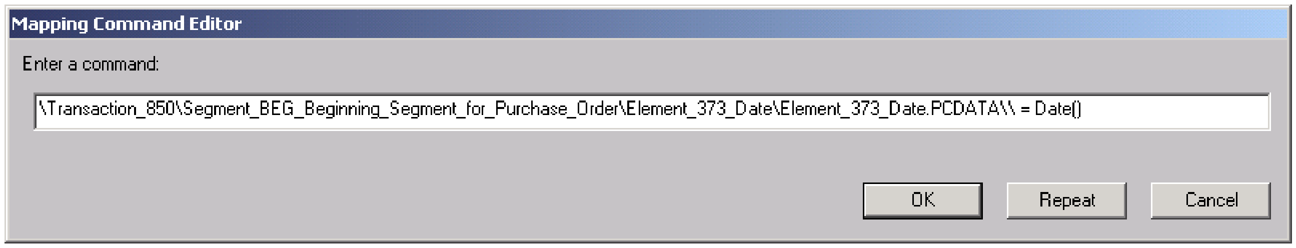
Figure 1-30 Creating a new variable

Using built-in functions

The WebSphere Data Interchange User Guide provides an extensive list of all functions available when creating a data transformation map. Functions perform an action and return a result within an expression or assignment statement. Functions can take zero or more parameters as input. The number of parameters and the data type of the return value vary from one function to the next.

In the example below, we use the Date() function. The Date() function returns the system date as a character string in the format yyyymmdd. Here we use an assignment statement to invoke the function; the target element will be assigned the return value.

Create an assignment statement as before. Right-click the **Mapping Command** window at the node where the command will appear and select **Insert Within -> Command -> Assignment**. Click the element you want to assign the system date on the top right side of the screen. While holding down the mouse button, drag it over the path in the Mapping Command Editor. Replace `expression` with **`Date()`**. The **`Date()`** function takes no parameters. Complete this command by clicking **OK**.



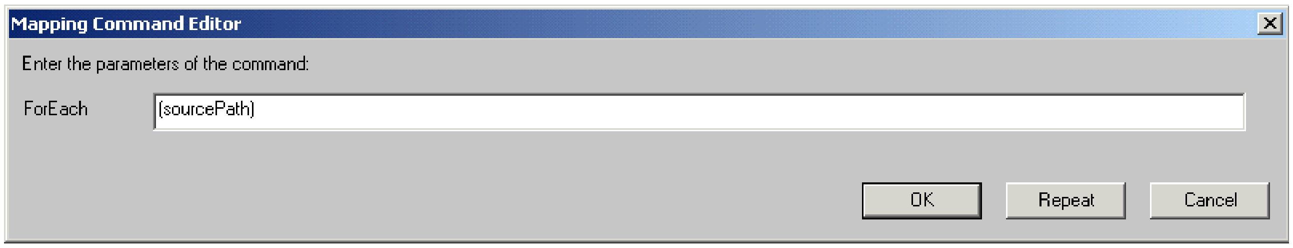
```
\Transaction_850\Segment_BEG_Beginning_Segment_for_Purchase_Order\Element_373_Date\Element_373_Date.PCDATA\\ = Date()
```

Figure 1-31   Mapping an element to the system date

Handling multiple occurrences of an element

The **`ForEach()`** command is used in target-based maps to indicate that subsequent nested mapping commands are to be executed for each occurrence of an element in the source document. Each occurrence of the element in the source document results in a new occurrence of the current element in the target document, unless other mapping commands, such as **`Qualify()`** or **`If()`**, limit execution of the nested commands.

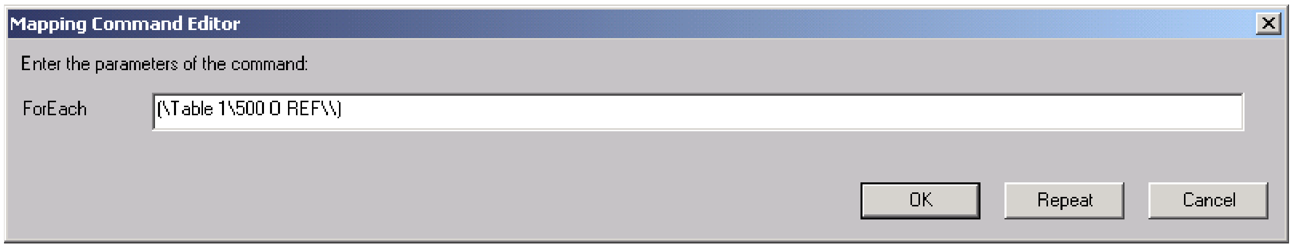
Create a **`ForEach()`** command by right-clicking the node where the command will occur in the Mapping Command window and selecting **ForEach**.



ForEach (sourcePath)

Figure 1-32   Using the ForEach command

Replace `sourcePath` in the Mapping Command Editor by clicking the repeating element in the top left side of the main window (see Figure 1-26 on page 24), where the source document is listed and expanded. While holding down the mouse button, drag the repeating element over sourcePath, resulting in a command as shown in Figure 1-33.



ForEach (\Table 1\500 O REF\\)

Figure 1-33   Completed ForEach() command

Mappings commands that occur within a **`ForEach()`** command are performed each time the associated element in the source document is encountered. **`ForEach()`** commands can only be used in target-based maps. They are inserted within simple elements or compound elements that occur in the target document definition. You can include multiple **`ForEach()`** commands within a single element in the target document definition. The element in the target document definition can be either a repeating element or a non-repeating element. If the target element is not repeating, you may need to use the **`Qualify()`** command or conditional logic (**`If()`** / **`ElseIf()`** / **`Else()`** commands) so that the mapping commands within the **`ForEach()`** command are executed only once. If a simple element in the target document is non-repeating and multiple values are written to it, the last value will overwrite the earlier values.

The **Qualify()** command should only be performed when specific conditions are satisfied. For instance, the first occurrence of a compound element may need to be mapped differently from all other occurrences of the compound element.

To qualify a **ForEach()** command by occurrence, right-click the **ForEach()** command in the Mapping Command window and select **Qualify -> By Occurrence**.

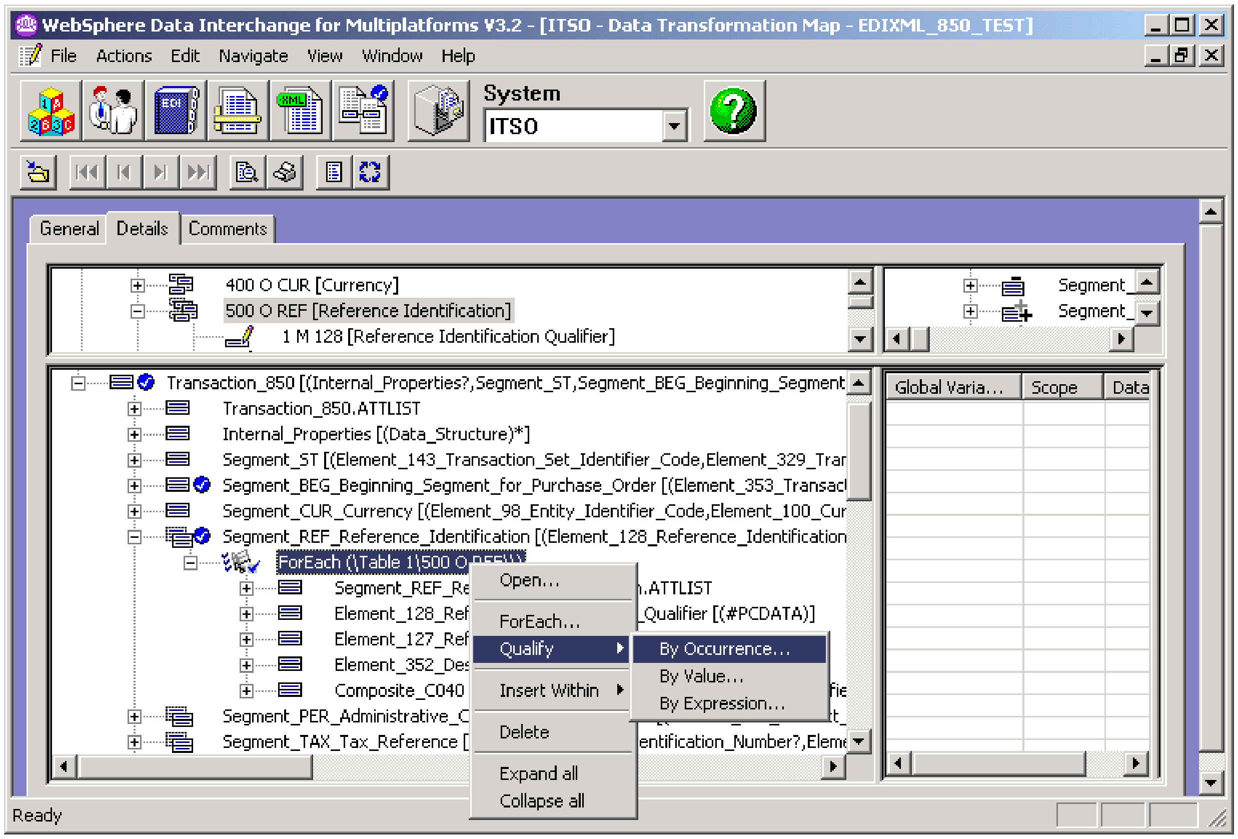
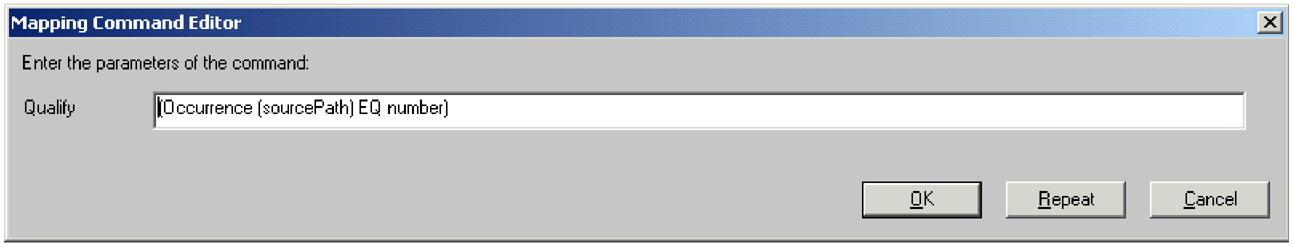

WebSphere Data Interchange for Multiplatforms V3.2 - [ITSO - Data Transformation Map - EDIXML_850_TEST]
File Actions Edit Navigate View Window Help
System
ITSO
General Details Comments
400 O CUR [Currency]
500 O REF [Reference Identification]
1 M 128 [Reference Identification Qualifier]
Transaction_850 [(Internal_Properties?,Segment_ST,Segment_BEG_Beginning_Segment
Transaction_850.ATTLIST
Internal_Properties [(Data_Structure)*]
Segment_ST [(Element_143_Transaction_Set_Identifier_Code,Element_329_Trar
Segment_BEG_Beginning_Segment_for_Purchase_Order [(Element_353_Transacl
Segment_CUR_Currency [(Element_98_Entity_Identifier_Code,Element_100_Cur
Segment_REF_Reference_Identification [(Element_128_Reference_Identification
ForEach (\Table 1\500 O REF\\)
Segment_REF_Re
Element_128_Ref
Element_127_Ref
Element_352_Des
Composite_C040
Segment_PER_Administrative_C
Segment_TAX_Tax_Reference [
Open...
ForEach...
Qualify
Insert Within
Delete
Expand all
Collapse all
By Occurrence...
By Value...
By Expression...
Segment
Segment
Global Varia... Scope Data
Ready

Figure 1-34 Adding a Qualify By Occurrence command

The Mapping Command Editor is displayed with default parameters which must be changed.

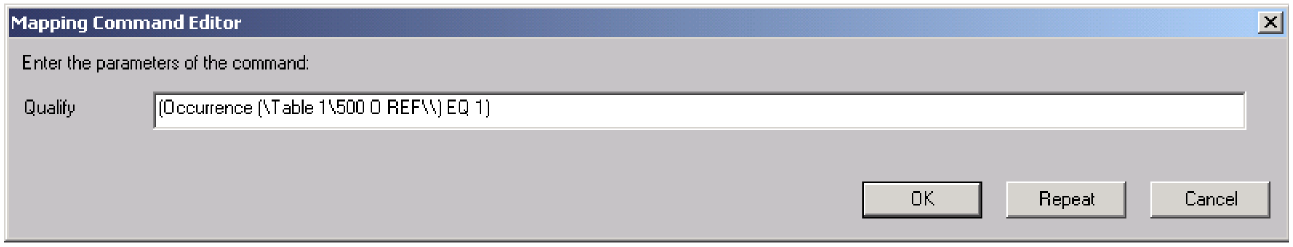


Enter the parameters of the command:

Qualify: Occurrence (sourcePath) EQ number)

Figure 1-35 Building a Qualify By Occurrence command

Specify the sourcePath as before. Next, overwrite `number` with the numeric value of the occurrence to be mapped. For example, EQ 1 is the first occurrence of the sourcePath element EQ. When you have completed the **Qualify()** command, the Mapping Command Editor should look as shown in Figure 1-36 on page 30.



Qualify: (Occurrence (\Table 1\500 0 REF\\) EQ 1)

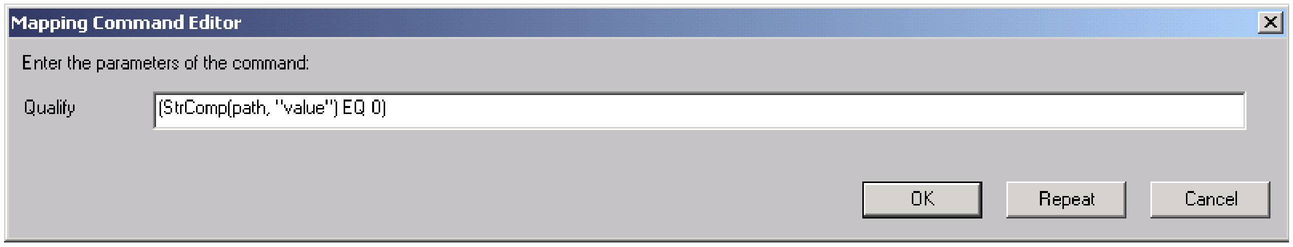
Figure 1-36 Completed Qualify() command

WebSphere Data Interchange allows for qualifications to be repeated. By doing so, a new set of mapping commands is created for each occurrence of the repeating source element as specified by the user.

`ForEach()` commands can also be qualified by value. Qualifying by value allows us to map an element differently if a condition exists to test the value of an element in the source document.

To qualify a **`ForEach()`** command by value, simply right-click the command as before and select **Qualify -> By Value**.

Qualifying by value utilizes the functionality available through the **`StrComp()`** function. The Mapping Command Editor appears with a default set of parameters.



Qualify: (StrComp(path, "value") EQ 0)

Figure 1-37 Building a Qualify By Value command

Replace `path` by clicking an element in the top left window and dragging it onto the Mapping Command Editor. It is the value in this element that will determine the qualification. Replace `value` with the value to test against the element, remembering not to remove the double quotes. No other detail needs to be changed. If the **`StrComp()`** function returns zero, the condition will be true and the subsequent mapping commands executed.

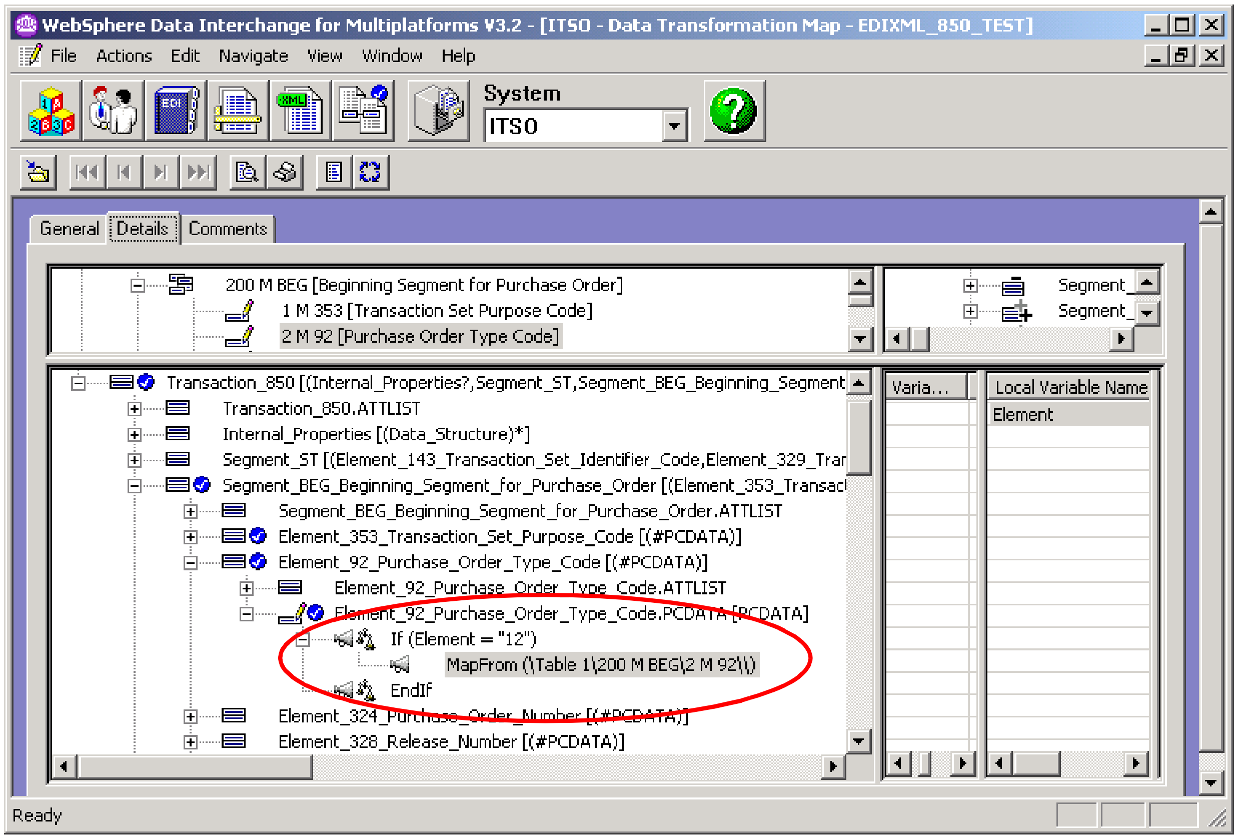
Using conditions in the transformation map

The **`If()`** / **`ElseIf()`** / **`Else()`** / **`Endif()`** commands are used to conditionally perform one or more mapping commands. The **`If()`** command marks the beginning of the If() condition block. The **`EndIf()`** command is used to mark the end of the If() condition block.

When the **`If()`** command is encountered, the expression will be evaluated. When the expression evaluates to True, the mapping commands within the **`If()`** command will be executed. If the expression evaluates to False, the WebSphere Data Interchange server will look for an **`ElseIf()`** command within the If() condition block. If an **`ElseIf()`** command is found, its associated expression will be evaluated. When the expression evaluates to True, the mapping commands within the **`ElseIf()`** command will be executed. If the expression evaluates to False, the server will look for the next **`ElseIf()`** command within the If() condition block. When all **`ElseIf()`** commands have been tested and evaluated to False, the server will look for an **`Else`** command within the If() condition block. If an **`Else`** command is found, the mapping commands within the **`Else`** command will be executed.

To create an **If()** command, simply right-click the node where the command is to occur in the Mapping Command window and select **Insert Within -> Command -> If**. The Mapping Command Editor prompts the user to enter an expression. An expression in this instance could take the form of a **StrComp()** function as seen previously or could perhaps test the numeric value of an element using a conditional operator.

Figure 1-38 illustrates a simple **If()** /**EndIf** command. Here, we test the value of a variable, called Element. If Element is equal to the number 12, then we execute a **MapFrom()** command as specified within the **If()** / **EndIf** command. Since the variable Element is an integer, we can use classic conditional operators. To test string variables or string elements, we can use the **StrComp()** function.


WebSphere Data Interchange for Multiplatforms V3.2 - [ITSO - Data Transformation Map - EDIXML_850_TEST]
File Actions Edit Navigate View Window Help
System
ITSO
General Details Comments
Segment
Segment
Varia...
Local Variable Name
Element
Ready

```
200 M BEG [Beginning Segment for Purchase Order]
    1 M 353 [Transaction Set Purpose Code]
    2 M 92 [Purchase Order Type Code]
```

```
Transaction_850 [(Internal_Properties?,Segment_ST,Segment_BEG_Beginning_Segment
    Transaction_850.ATTLIST
    Internal_Properties [(Data_Structure)*]
    Segment_ST [(Element_143_Transaction_Set_Identifier_Code,Element_329_Trar
    Segment_BEG_Beginning_Segment_for_Purchase_Order [(Element_353_Transacl
        Segment_BEG_Beginning_Segment_for_Purchase_Order.ATTLIST
        Element_353_Transaction_Set_Purpose_Code [(#PCDATA)]
        Element_92_Purchase_Order_Type_Code [(#PCDATA)]
            Element_92_Purchase_Order_Type_Code.ATTLIST
            Element_92_Purchase_Order_Type_Code.PCDATA [PCDATA]
                If (Element = "12")
                    MapFrom (\Table 1\200 M BEG\2 M 92\\)
                EndIf
        Element_324_Purchase_Order_Number [(#PCDATA)]
        Element_328_Release_Number [(#PCDATA)]
```

Figure 1-38 Using conditional mapping

Using MapCall and MapSwitch commands

The **MapCall()** command is used to indicate that a new map must be used to process the data within the current source element (for source-based maps) or the specified target element (for target-based maps). This is commonly called an imbedded map.

When this command is encountered, the data from the source element is translated using an imbedded map. The imbedded map receives the data from the source element to use as its input document. The element can be a simple element (for example, the BIN02 element in a BIN segment) or it can be a compound element (a subtree or subset).

When translation within the imbedded map has completed, translation resumes in the original map if the translation with the imbedded map completed successfully. If a significant translation error occurs in the imbedded map, translation will not resume in the original map.

Using the **MapCall()** command and imbedded maps can be compared to calling a subroutine in an application programming language. You can, for example, build a single map to transform a data segment that occurs in more than one EDI document. Using imbedded maps then results in less development effort and in more consistent mapping across EDI documents.

The **MapSwitch()** command is used to indicate that a document needs to be translated by another map instead of the current map. Any translation performed by the current map is terminated. The document is translated by the map identified in the **MapSwitch()** command.

This command can be used when data in the document must be inspected before it can be determined which map should be used. The command allows you to switch the map dynamically based on the data that is contained in the document. You can create a map that will initially examine the data in the document. Only the compound and simple elements necessary to make a decision are mapped. The map that should be used to translate the document is determined based on the mapped elements. Then use conditional mapping commands and the **MapSwitch()** command to begin translating the document with an alternate map.

Before mapping can start

Given that creating a map is similar to building an application program, it comes as no surprise that you cannot just jump into the Map Editor without spending some time to design and to formally specify the map. The first step is to build a mapping specification. A mapping specification could take many forms, but a good way to start is to create a table that lists all elements of both the source and target documents. If a target element needs to have a value but the source document does not contain an equivalent element, you can use the column Comments and special instructions to specify how this element is going to be populated. Usually, this is going to be via an assignment of a constant or calculated value.

If the source document contains information that is not required to build the target document, you should still list that element in the column Source element, while the field target element is empty. This leaves no doubt about the fact that this element contains unmapped information.

The column Comments and special instructions can also be used to describe additional constraints such as valid values or range or correlation information. Often, when a certain element has a specific value, it has an impact on other elements and it requires that those elements have valid values. In these cases, it would be wise to have several variations of the same table for the same two documents: a version where that element has value A and another version of the table where that element has value B.

By analyzing the requirements and building the specification in this way, you will find it easy to design the actual map. For example, the example where the value of an element results in a variation of the specification table can easily be implemented by using a **MapCall()**. Build a map for that section of the document that is impacted by the value of that element. If that element has three different kinds of values then this leads to three different sub-maps. The parent map will then contain an If() statement to branch to the correct sub-map.

The following table provides a simple mapping specification to build an EDI 855 document when the source document is XML. Later in this book, we will build this map.

Table 1-1 Mapping specification for an EDI 855 document

Position	Target element	Source Element	Comments and special instructions
1	353 Transaction Set Purpose Code		Mandatory element Filled in by assignment to value ‘06’
2	587 Acknowledgment Type	Header.Response	
3	324 Purchase Order Number	Header.PONumber	
4	373 Date		Mandatory element Filled in by calling the system function Date()
5	330 Quantity Ordered	Detail.Quantity	
6	235 Product/Service ID Qualifier		Filled in by assignment to value ‘ID’
7	234 Product/Service ID	Detail.ItemNumber	
8	349 Item Description Type		Filled in by assignment to value ‘F’
9	352 Description	Detail.Description	

1.5.8 Mapping rules

How does WebSphere Data Interchange know which map to use and which queue to put the output data on? WebSphere Data Interchange makes this decision based on the usage (rules) defined by the user for a particular map. This area is known as “rules” in WebSphere Data Interchange because conceptually it is simplest to think that whenever a message arrives on a queue monitored by WebSphere Data Interchange, there is a “rule” (created by the user) that defines what WebSphere Data Interchange should do with that message.

Each rule relates to a particular map and so a map must first be created before we can add a rule to determine its usage. We can see all maps in our system by clicking the map icon. The map icon is shown in Figure 1-39.

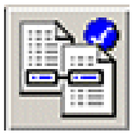


Figure 1-39 Icon to access the mapping editor

If we select the map by clicking it, we can either see existing usages or create new usages by clicking the usage icon (Figure 1-40).

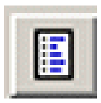
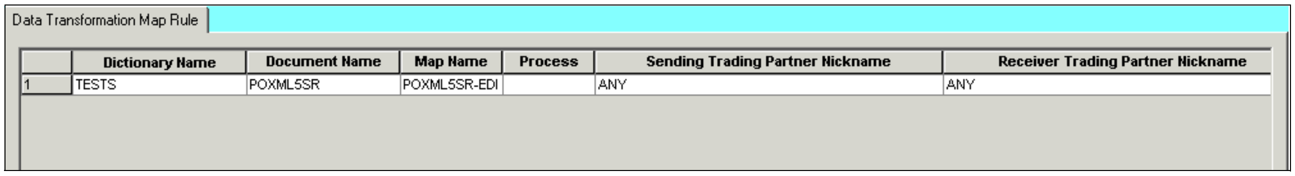


Figure 1-40 Icon to access the map usage editor

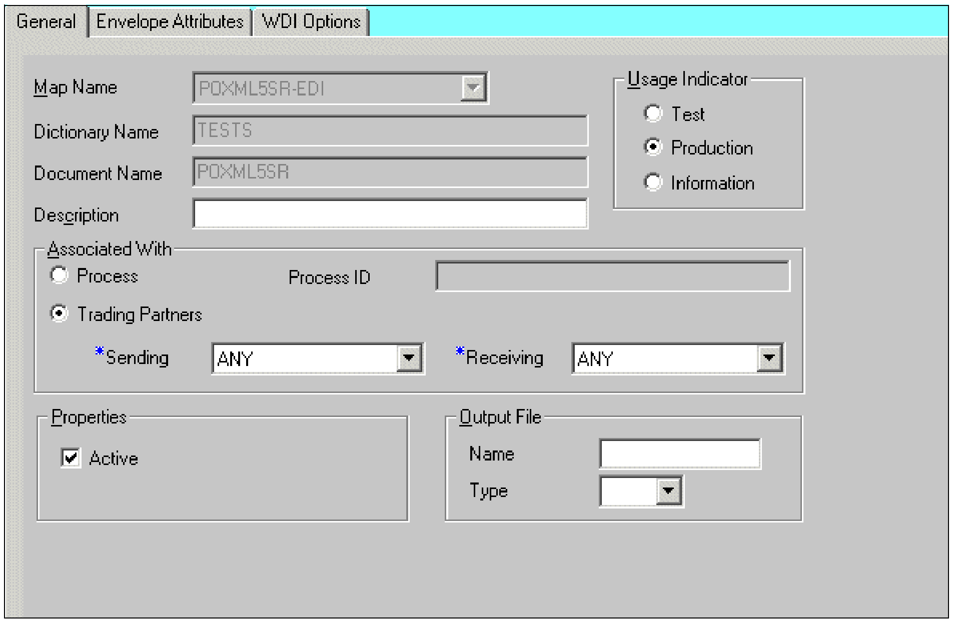
In Figure 1-41, we can see the sample data transformation map shipped with WebSphere Data Interchange; it has also been shipped with a sample rule for usage.



	Dictionary Name	Document Name	Map Name	Process	Sending Trading Partner Nickname	Receiver Trading Partner Nickname
1	TESTS	POXML5SR	POXML5SR-EDI		ANY	ANY

Figure 1-41   Sample rule for a data transformation map

If we look at the details of this rule, we can identify the areas of importance.



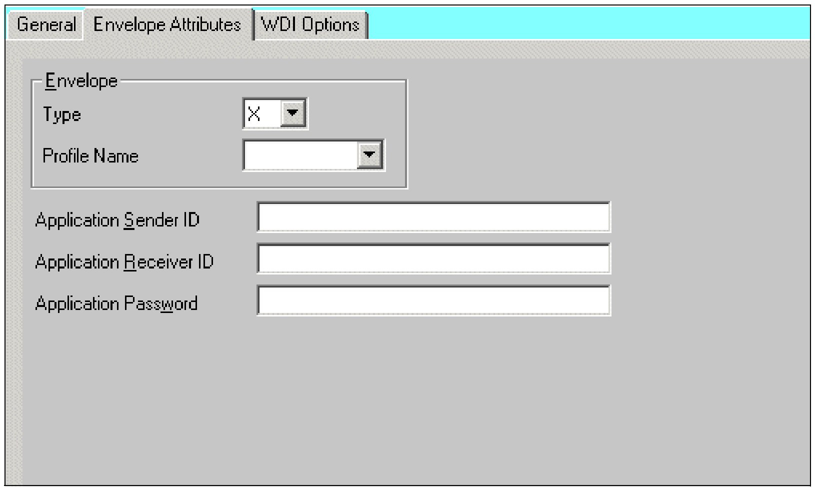
Map Name: POXML5SR-EDI
Dictionary Name: TESTS
Document Name: POXML5SR
Description:
Usage Indicator: Test [✓ Production] Information
Associated With: Process [✓ Trading Partners] Process ID:
*Sending: ANY
*Receiving: ANY
Properties: [✓ Active]
Output File: Name: Type:

Figure 1-42   Details of the usage of a map

The Map Name, Dictionary Name and Document Name are all in grey and cannot be changed since they are pulled from the map detail. In the Associated With frame, the user has the opportunity to associate the map with either a particular process or a set of Trading Partners. If selected, the process field allows the map to become associated with a particular business process, for example 850, 855, etc. If the Trading Partner field is selected, the user has the opportunity to specify a sender or receiver. These drop-down menus are populated by the list of trading partners available in the Trading Partner profile area described earlier.

To activate a rule to allow a map to perform data transformation, select **Active** in the Properties frame and set the Usage Indicator to `Production`. An Output File Name and Type can also be specified here. This detail in this field will be overwritten if an output file is specified in the PERFORM statement of the Service profile as described earlier.

The Envelope Attributes tab defines the type of envelope to be used on the EDI message once it has been created. In Figure 1-43 on page 35, we see that **X** has been selected. X in this instance defines an ANSI X12 standard envelope.


General | Envelope Attributes | WDI Options

Envelope
Type: X
Profile Name:

Application Sender ID:
Application Receiver ID:
Application Password:

Figure 1-43 The Envelope Attributes tab of a usage of a map

The WDI Options tab allows the user to specify varying levels of validation on a translation taking place using this usage.

1.6 Usage patterns for WebSphere Data Interchange

This section discusses some scenarios where WebSphere Data Interchange is being used in conjunction with other components of a typical enterprise IT infrastructure.

1.6.1 A point-to-point solution

A first implementation of WebSphere Data Interchange in an enterprise environment consists of a direct link between an ERP system or other internal application that manages your business, and an EDI broker, such as WebSphere Data Interchange. The communication between those two components can be MQ-based or file-based. The organization of the data that is being passed between the ERP and the EDI broker can be XML documents or data that can be modeled as C structures or COBOL copybooks.

Once the information has been translated into the appropriate EDI standard, it is handed over to some communication product that interacts with the network or VAN. Note that WebSphere Data Interchange is separate from the actual communications infrastructure and hence can work with a number of software products. WebSphere Data Interchange does provide configuration support for a number of software products, such as IBM Information Exchange, iSoft AS2 client, or TPI. The interaction between the EDI broker and the communication software can be file-based or MQ-based.

While this scenario is labeled as point-to-point, it still works perfectly well for communication with a number of partners. The point-to-point label refers more to the direct link between a single internal application and the EDI broker instance.

For inbound communication, the EDI information is received by the communication software and handed over the EDI broker via files or queues. The EDI broker will then translate the information into the appropriate format for the back-end system.

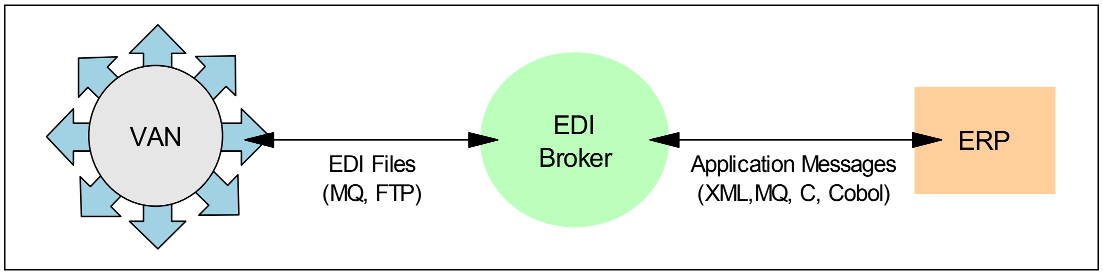


Figure 1-44 Point-to-point EDI solution

1.6.2 An integration broker solution

Typically, there will be more than a single application system. Referring to Figure 1-3 on page 4, a typical business cycle will involve a number of application systems, such as order management, scheduling systems, and so on. You can, of course, apply the point-to-point solution in this environment, too. However, as discussed before, application integration between the different components of an IT infrastructure is almost a prerequisite for a successful EDI implementation. Hence, the integration of the EDI broker into the existing EAI infrastructure is an important advantage.

Figure 1-45 shows a schematic view of the integration between a message broker and an EDI broker. The message broker is responsible for distributing information between the different applications. When the broker receives information from one system, it can pass it over to other systems, including the EDI broker. When, for example, the EDI broker receives a purchase order document, it can translate it into an XML document and hand it over to the message broker. The message broker, in turn, will fan out to the different internal systems, based on message content and system state. The contents of each message generated by the message broker can be set with information out of the incoming EDI message but can also be enriched by the broker.

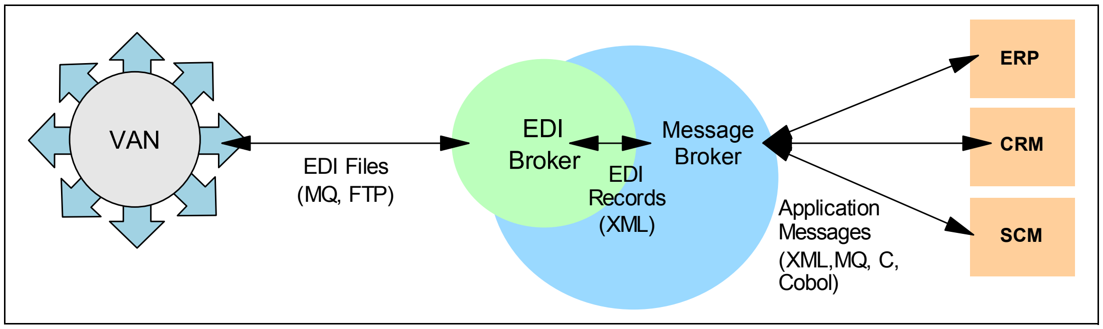


Figure 1-45 Integration broker EDI solution

1.6.3 A B2B gateway solution

While the use of EDI technology is widespread, technology changes and evolution have resulted in the use of many types of B2B communication infrastructures. Besides the traditional VAN-based EDI communication, Internet-based techniques have become available, too. AS1 and AS2 have been mentioned before and those protocols are still tied more or less to traditional EDI communications. More recently, Web services-based technologies also became available for use in the B2B area. While this technology is still maturing, it is clear that a flexible B2B solution should handle multiple communication techniques.

A B2B gateway solution, such as WebSphere Business Integration - Connect, offers an answer to these challenges. As shown in Figure 1-46, an EDI broker works next to an AS2 solution and a Web services solution. This offers trading partners a wide range of technology options for interacting and at the same time there is a single point of control and management for all technologies.

Note that the B2B gateway solution can be integrated with the integration broker solution.

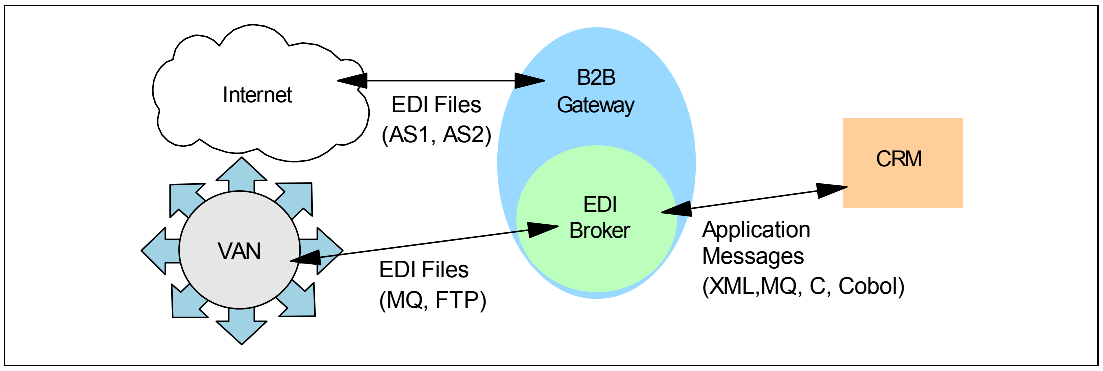


Figure 1-46 The B2B gateway solution

1.7 Introducing the iSoft Peer-to-Peer Agent

iSoft's Peer-to-Peer Agent, hereafter referred to as *P2PAgent*, enables you to exchange documents between trading partners over the Internet in a secure and reliable way. In this section, we describe the main features and components of the P2PAgent program.

1.7.1 Communication features

Since the main task of the P2PAgent is to move data from one company to an external trading partner, the communication features are an important aspect of the product. The P2PAgent program can accept data from internal applications in a number of ways.

A more traditional way of passing data to the agent is by delivering files in a given directory. The agent can filter through these files using a number of selection criteria to determine what to do with a given file. Also, when a file has been sent, you can choose to rename the file or delete it. Simply preserving the file could result in the file being sent multiple times. The file system can also be used to store received files. You can configure the agent in such a way that the original file name (as it was named by the sender) is preserved, or that the file name is being generated. This last option can avoid files being overwritten accidentally.

A more recent addition to the product is support for WebSphere MQ. The P2PAgent can retrieve messages from an inbox queue. It considers each message as a separate entity that should be sent to the correct destination trading partner. Also, when the agent receives documents from trading partners, the agent can store the received document as a single message in a queue. By default, such a message will be prefixed by an MQRFH2 header that contains metadata information, such as the trading partner that had sent the document and the target trading partner ID. The MQRFH2 header is constructed in such a way that this information is also available to JMS clients in the form of message properties.

Further internal data delivery mechanisms include support for SMTP and HTTP. A received document or a received receipt can be delivered as an e-mail to a configured e-mail address using an SMTP server. HTTP communication for sending documents is used, for example, in a multi-machine setup of iSoft's P2PAgent. However, if your internal applications can hand over EDI documents using HTTP, then they can hook into the P2PAgent directly.

Since the P2PAgent program is an AS2 client program, the agent supports HTTP and HTTPS for sending and receiving documents through the Internet. The agent is also an AS1 client, which means that it needs to support SMTP for sending and receiving documents as an e-mail attachment.

Figure 1-47 summarizes the support for the different techniques to move data to and from the agent.

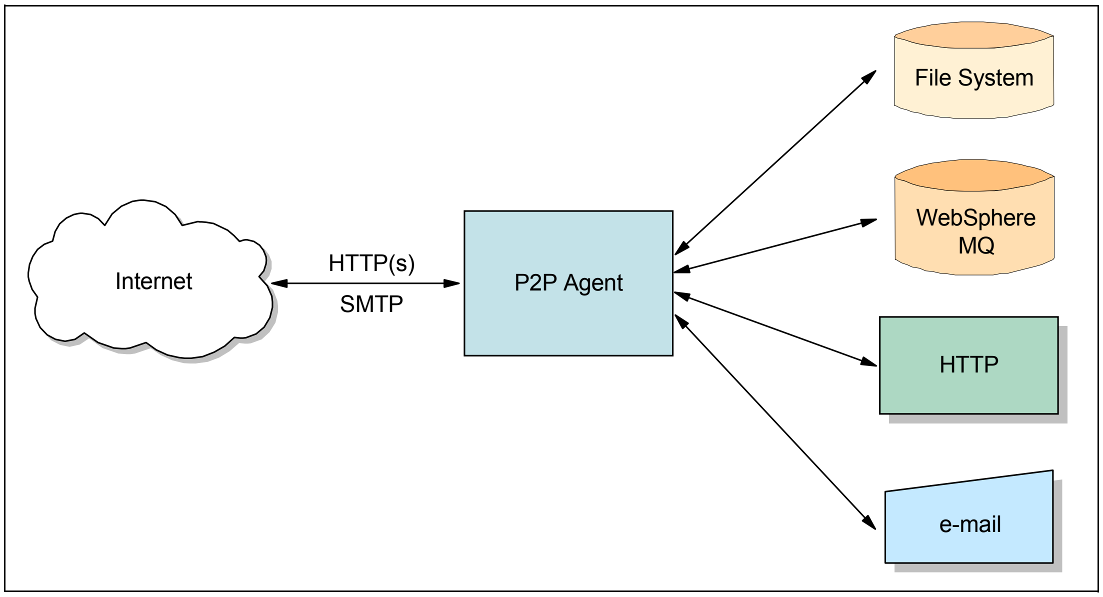


Figure 1-47 Inbound and outbound communication options

1.7.2 Data integrity and security characteristics

Since the P2PAgent program is designed to move business critical data, such as purchase orders, over a public network, it is required that the agent support several techniques to protect your data.

If configured to do so, the agent will encrypt the data using the certificate of the target partner. This means that only the target partner can read the document, since only the target partner is supposed to have access to the private key to decrypt the document. As such, this guarantees confidentiality. The agent supports the encryption algorithms RC2 and Triple DES. For additional data integrity, you can request to use base64 encoding.

As a receiver of a document, you want to be sure that the document has been sent by the trading partner that you expect. The sending partner can digitally sign the document using its private key. As a receiver, you can verify the signature using the certificate of the sender. The agent supports two digital signature algorithms: SHA-1 and MD5. Both can be combined with base64 encoding, if requested.

As a sender of a document, you want to have proof that the receiver has received the document and that only he has received it. The sender can request a receipt, sometimes referred to as an *MDN* (message disposition notification). Receipts can be delivered asynchronously or synchronously and can be signed and encrypted using the same technologies that are available for sending documents.

The status of a document, including several statistics, can be generated by the P2PAgent in the form of a notice. Again, this notice can be a message in a queue or a file in a named directory. Any errors can be reported in a daily log file. Alternatively, an error for a given document can result in an error file or message specifically tied to that document. The error document can contain, for example, HTTP error codes. The P2PAgent can also be configured to try to send a document several times, for example three times at an interval of one minute. If the agent has tried to send a document the maximum allowed number of times, the document can be stored in a separate location: a given directory or queue. This feature is sometimes referred to as the *recycle* feature. The message or file contains nothing but the original document, while the error itself is stored in the error message or file. The advantage of this separation is that the recycled document can be used without any alteration in a back-up transmission system, such as a VAN. The recycle feature will be exploited and described in Chapter 5, “Implementing a back-up solution using IBM Expedite” on page 199.

1.7.3 Administration features

Configuring and managing a P2PAgent configuration can be done in a variety of ways. The simplest form is to use the interactive session when the agent is started. The agent runs in a command window and commands can be entered at the prompt.

Figure 1-48 on page 40 shows the console view of the P2PAgent. After start-up, the user has executed the status command for which you can see the output in the lower part of the figure. This output also shows us what different options can be configured and tuned. We will introduce and use some of them throughout this redbook. For a complete reference, please consult the product documentation.

```
C:\iSoft_Advanced>p2pagent_odbc_ibm_unlimited.exe
iSoft(R) Peer-to-Peer Agent(TM) for MQSeries(R)
(C) Copyright 2001-2002 iSoft Corp.
Build: 3.1.2002.10.30.1 [Nov 27 2002 15:06:36]
IBM Unlimited Edition
Authorized License
2003.02.05 15:30:09.521       POPT OK  Error path set to [error]
2003.02.05 15:30:09.521       POPT OK  Inbound errant will be stored
2003.02.05 15:30:09.531       POPT OK  Log path set to [log]
2003.02.05 15:30:09.531       POPT OK  Trace set to WRITE_FILE
2003.02.05 15:30:09.541       POPT OK  Notice path set to [mq://FMCQM/notices]
2003.02.05 15:30:09.541       POPT OK  Notices will be written to file
2003.02.05 15:30:09.561       POPT OK  Work-order path set to [mq://FMCQM/workorders]
2003.02.05 15:30:09.561       POPT OK  Work-order searching enabled
2003.02.05 15:30:09.571       POPT OK  Work-order file-spec set to [wo]
2003.02.05 15:30:09.581       POPT OK  PKI path set to [pki]
2003.02.05 15:30:09.591       POPT OK  Async. receipt path set to [mq://FMCQM/receipts]
2003.02.05 15:30:09.611       POPT OK  First-receive interval set to [300000ms]
2003.02.05 15:30:09.621       POPT OK  Mailbox host set to [mq://FMCQM]
2003.02.05 15:30:09.621       POPT OK  Mailbox address set to [0.0.0.0]
2003.02.05 15:30:09.631       POPT OK  Mailbox port set to [0]
2003.02.05 15:30:09.711       HPIM OK  HTTP inbound service started
status
ok

 Build:        3.1.2002.10.30.1             Data Source:                NONE
 Host:         vdputteg                     SMTP Host:                  NONE
 Control IP:   9.24.104.115:3501            SMTP User:                  NONE

 --Services--------   --Config-------------   --Timeouts----------------------
 Serialize:    ON     Partner-Pairs:      2   Connect:                 30.000s
 Outbound:     ON     Key-Pairs:          3   First-Receive:          300.000s
 Control:      OFF    Inbound Ctlrs:      1   Next-Receive:            90.000s
 Work-Orders:  ON     Outbound Txns:      0   First-Send:              30.000s
 Beacon:       OFF    Transports:         0   Next-Send:               90.000s
 Router:       OFF    Trace Level:        3   Resend Wait:             60.000s
 Web-UI:       OFF    Buffer Size:     4096   Beacon Wait:             20.000s
 PKI Admin:    OFF    Peer Group:         0   Work Order Interval:     10.000s
                      Role:                   Stop Thread:             10.000s

 --Options---------   --Locations---------------------------------------------
 Local Config: YES    Error Path:             error
 Show Trace:   YES    Log Path:               log
 Write Trace:  YES    Notice Path:            mq://FMCQM/notices
 Fast Write:   NO     PKI Path:               pki
 Notices:      FILE   Receipt  Path:          mq://FMCQM/receipts
                      Work-Order Path:        mq://FMCQM/workorders
                      Work-Order Extension:   wo
```

Figure 1-48 Console view of the P2PAgent

For settings and commands that you need to execute every time at start-up of the agent, you should use the configuration file, which is named by default p2pagent.cfg. We will use the configuration file a lot when implementing an iSoft solution in this redbook. Some packages of the iSoft product provide a command line utility, called buildcfg, that will quickly generate such a configuration file by asking the end user to answer some simple questions. This utility only builds a starting configuration file and it is very probable that you will need to edit this file to optimize the final configuration.

Series of commands that you execute from time to time but are not required to be executed during the start-up of the agent can be stored in a work order file. You can then use the **batch** command to execute the commands whenever this is appropriate. We will use this technique to generate keys for a new trading partner in 2.2, "Basic implementation of iSoft" on page 51. Work order files can also be executed automatically. The agent can be configured to monitor a directory for files with a given extension (the default extension is .wo). When such a file is dropped in the correct directory, the agent will read it and execute the commands.

Example 1-3 shows a sample work order file. It has an XML format where individual commands are the data values of a command XML element. The actual command can be any command that is supported by the agent. The actual command does not look any different from what you would have typed in an interactive session.

Example 1-3 Sample work order file

```
<xml>
<command> send http QA_a QA_b -fNedi\edifile.x12 -r1 -cE</command>
</xml>
```

In addition to file-based work orders, you can use a message queue. However, the contents of the message should not be in the XML format that is shown in Example 1-3. The contents of the message should be the command text only, without the `xml` and `command` tags. Unfortunately, the response or output of the command is not sent back as a reply message. The output of the command is shown in the console view and in the log file, if activated.

Web-based status reporting can be turned on by executing the command **start GUI**. The P2PAgent will then listen on port 80 for any HTTP requests. If a request is received, it responds with the output of the **status** command. The standard product does not offer a complete Web-based administration tool. However, this functionality is available as an add-on product.

1.7.4 Load-balancing and multi-machine setup

For environments where many documents need to be sent and received, a single machine might not be sufficient to handle all the workload. iSoft's P2PAgent provides a solution to this problem. The task of sending and receiving documents can be split into three different roles:

- The Transport role: this is the task of encrypting, decrypting, signing and verifying signatures, as well as sending and receiving documents. Encryption and digital signature algorithms are mathematically intensive operations and it is therefore useful if these tasks can be spread over multiple machines.
- The Router role: this is the role that can perform software-based load sharing between multiple computers that provide the Transport role. The Router provides a single point of entry for data of a given Internet protocol. The actual processing of the incoming data is then delegated to one of the computers performing the Transport role. Figure 1-49 on page 42 graphically shows this inbound load sharing technique.

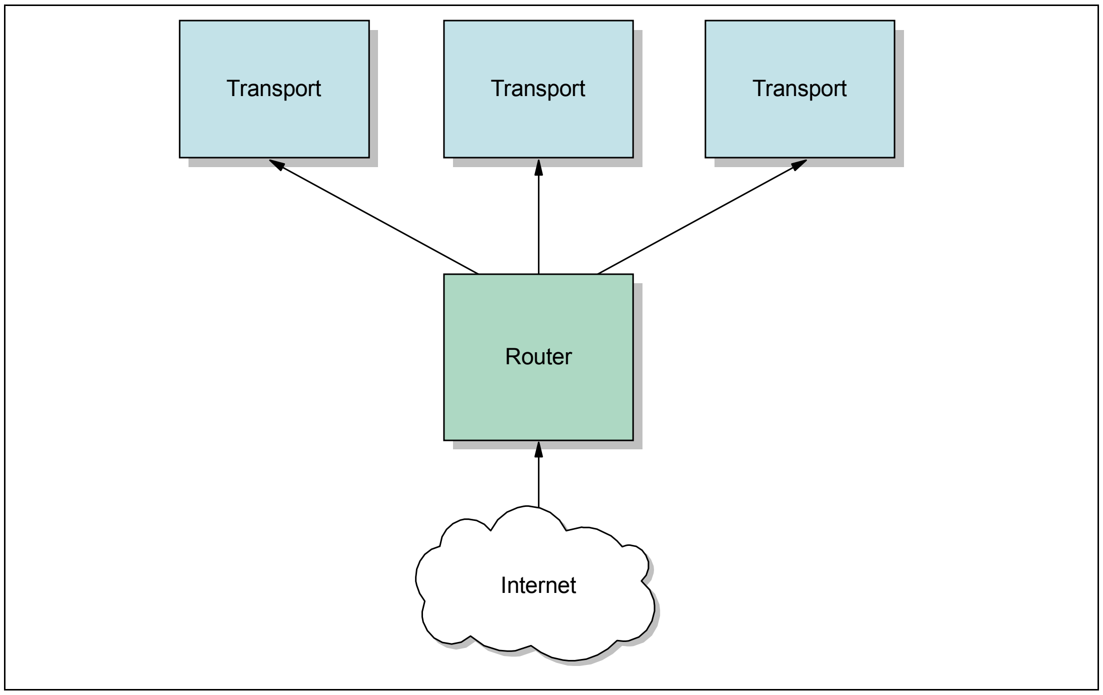


Figure 1-49 Inbound load sharing as provided by iSoft's P2PAgent

- The Admin role: this is the role that performs the outbound distribution, in addition to other services. Documents that have been prepared for sending to trading partners are distributed by the Admin role to the computers that provide the Transport role. In addition to this outbound load sharing, the Admin role also provides administration services to manage trading partner relationships, for example. Figure 1-50 on page 43 graphically shows how the different roles work together. The EDI Generator role, for example, could be provided by WebSphere Data Interchange.

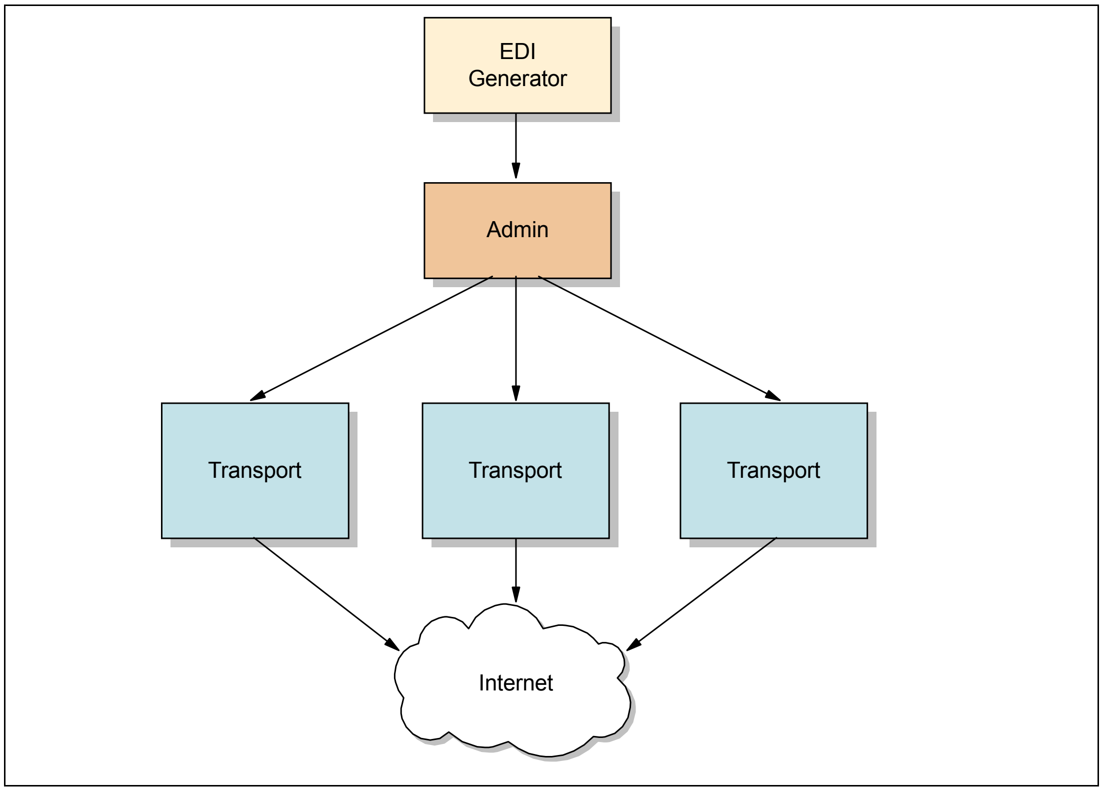


Figure 1-50 Outbound load sharing as provided by iSoft's P2PAgent

In this redbook, we will focus on an implementation of iSoft's P2PAgent where all roles are performed by a single computer. For more information about setting up a multi-machine implementation of the P2PAgent, please refer to the product documentation of iSoft's P2PAgent.

1.8 Introducing Trading Partner Interchange

The TPI Server enables the secure exchange of documents among trading partners over the Internet. The application packages documents in secure envelopes that are transmitted among trading partners according to user-configured schedules.

TPI organizes the information you need to exchange documents with your trading partners into company and partner profiles. This makes it easy to set up and maintain trading relationships. Company profiles define how you receive documents from your partners. Partner profiles define how you send documents to your partners.

To establish a trading relationship, you create and export your company profile to your trading partner, who imports it to TPI as your partner profile. Conversely, your partner creates and exports a company profile that you import as a partner profile on your system.

TPI supports HTTP/HTTPS, FTP, SMTP, and WebSphere MQ (MQSeries) for transporting documents among trading partners. The TPI Server uses a system of directories for delivery and retrieval of documents. The TPI Server maintains six document directories for each trading partner: three for inbound and three for outbound. Each of the directories holds incoming or outgoing documents of a specific format: XML, EDI, or binary.

In this section, the following topics are described:

- How the system works
- Company profile
- Partner profile
- The relationship between company and partner profiles
- Document sizes
- Transports

1.8.1 How the system works

Figure 1-51 on page 45 and Figure 1-52 on page 46 present high-level views of how TPI processes outbound and inbound documents. These figures show typical document flows, although your organization’s configuration might differ.

The flow starts on the left-hand side with the generation of a document by an internal application within the IT infrastructure. This application can be a translation engine, such as WebSphere Data Interchange, which translates documents in an internal representation to documents in a EDI standard. Or, it can be a back office application system, such as SAP. TPI makes a distinction between three possible structures of EDI documents. The document can be a binary document, an XML document or a traditional EDI document. Each document category has its own outbound directory. When the TPI Server detects the arrival of a new document in one of these three directories, it will store it in a back-up directory and parse it to extract routing information. This routing information is used to determine the partner profile that should be used to send the document to the correct partner. If the parsing fails or if the system cannot map the routing information in the document to a profile in the TPI database, the document is moved to a directory where rejected documents are stored.

Based on the configuration, the document might be compressed, signed and encrypted before it is queued for transmission to the partner. When the actual transmission takes place, the trading partner should send an MDN in return to acknowledge the receipt of the document. The sending partner will then store the sent document in its archive folder.

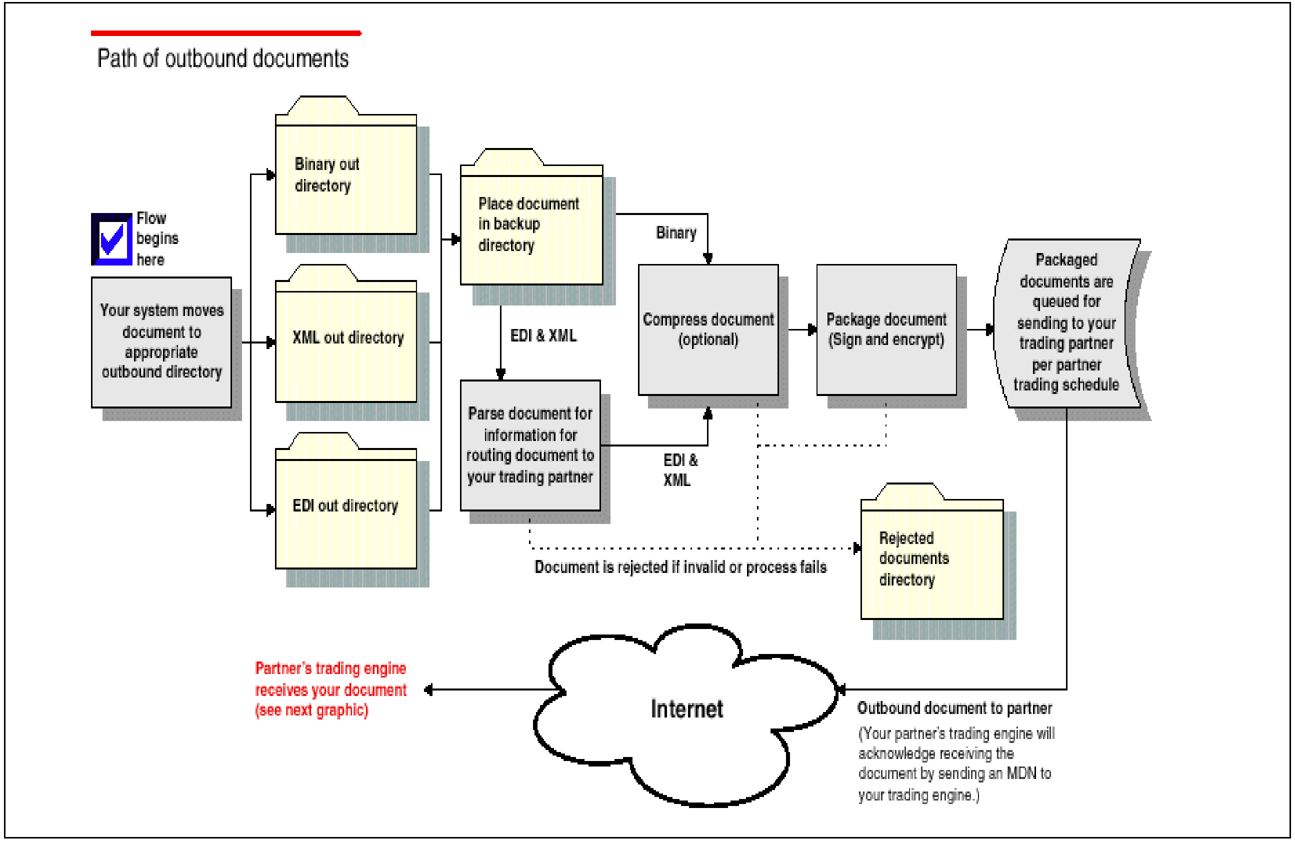

Path of outbound documents
Flow begins here
Your system moves document to appropriate outbound directory
Binary out directory
XML out directory
EDI out directory
Place document in backup directory
EDI & XML
Parse document for information for routing document to your trading partner
Binary
Compress document (optional)
EDI & XML
Package document (Sign and encrypt)
Packaged documents are queued for sending to your trading partner per partner trading schedule
Rejected documents directory
Document is rejected if invalid or process fails
Partner's trading engine receives your document (see next graphic)
Internet
Outbound document to partner
(Your partner's trading engine will acknowledge receiving the document by sending an MDN to your trading engine.)

Figure 1-51 Outbound document processing

When a document is being received by the TPI Server, it is first stored in a back-up directory. Next, the server will process header information so that it can determine whether the document is being sent by a valid and known trading partner. The header will also provide information to the server about the packaging, such as encryption, signature and compression. Based on that information, the server will decrypt the document, verify the signature and decompress it. Finally, the document is stored in one of the three folders corresponding to the document category. From there, the document can be picked up by a translation engine or another internal application system.

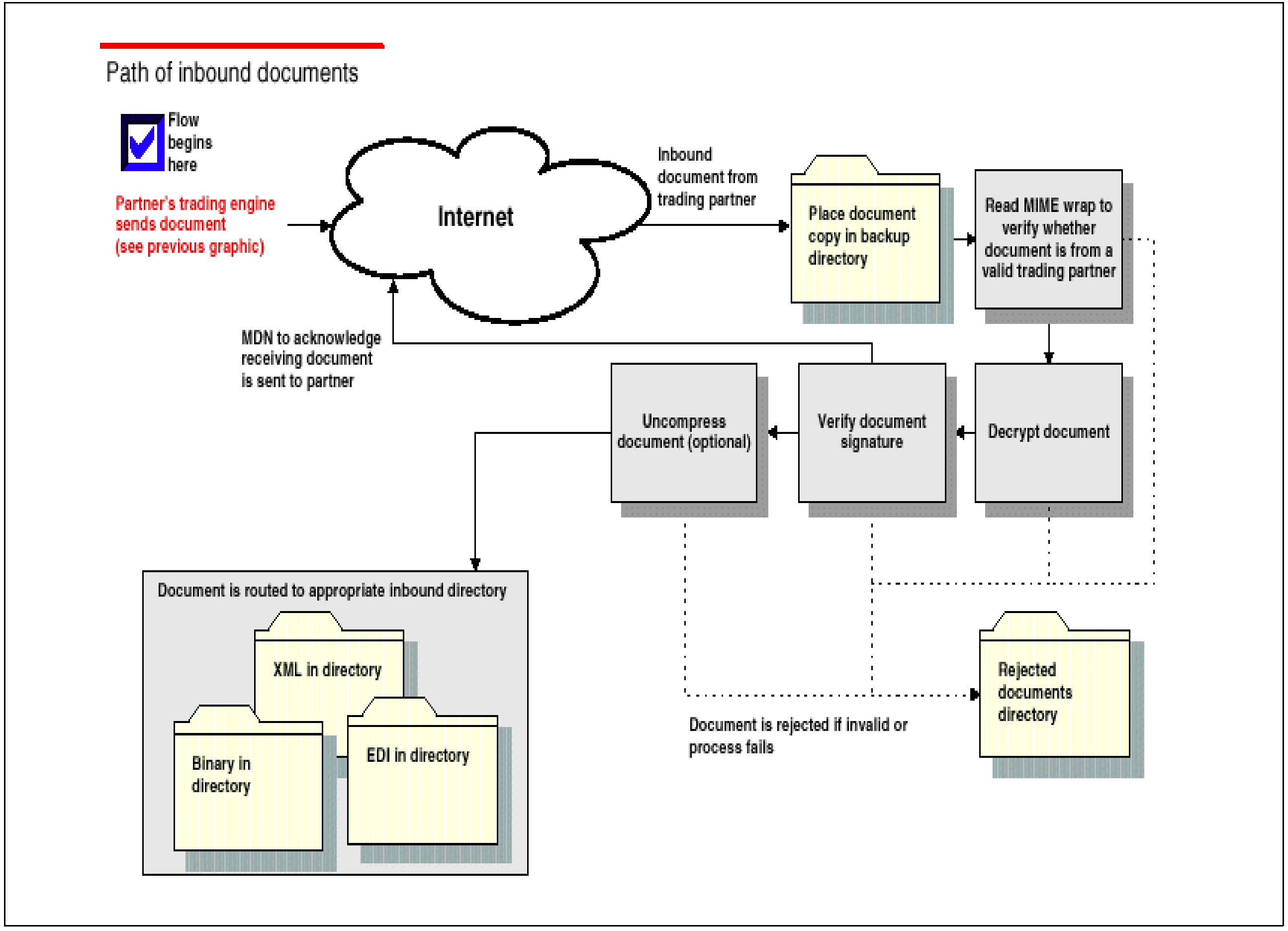

Path of inbound documents
Flow begins here
Partner's trading engine sends document (see previous graphic)
Internet
Inbound document from trading partner
Place document copy in backup directory
Read MIME wrap to verify whether document is from a valid trading partner
MDN to acknowledge receiving document is sent to partner
Uncompress document (optional)
Verify document signature
Decrypt document
Document is routed to appropriate inbound directory
XML in directory
Binary in directory
EDI in directory
Rejected documents directory
Document is rejected if invalid or process fails

Figure 1-52 Inbound document processing

1.8.2 Company profile

You can use the company profile information viewer to set up and maintain company profiles. With a company profile, you can trade with different trading partners using:

- Any transport; you may use different transports with different partners. You do not have to use the same transport method for all your trading partners.
- Any document type, including X12, EDIFACT, XML or binary documents such as those generated by legacy business applications, SAP, PeopleSoft or Oracle Financials.

You might find it necessary to set up more than one company profile. Each company profile you set up must have its own ID. Moreover, creating additional company profiles affects the performance of TPI by adding to its processing overhead. You should not create multiple company profiles unless you need them.

One reason for creating more than one company profile could be that you have more than one business unit, each using a different EDI ID.

1.8.3 Partner profile

One way to create a partner profile is to import a company profile file that was sent to you by a trading partner who also uses Trading Partner Interchange (TPI). When imported, the profile, which contains your partner's identity and transport information, becomes a partner profile on your system.

Importing a profile from a partner who uses TPI is the simplest and most direct method of adding a new partner profile to your system. You must manually create partner profiles for your partners who use a different communication engine.

Creating a partner profile lets TPI know how to send documents to that partner. The partner profile describes the transport the partner is using, as well as his name and address. This is just like giving your e-mail ID to someone, who loads it in his address book, so that he can e-mail you at any time.

1.8.4 The relationship between the company and partner profiles

When you or your partner sets up a company profile, you each decide what transport method or methods to make available to all of your trading partners. At a minimum, you must select one transport method by which your partners can send documents to you and complete all the fields for that method. If you choose to support additional transport methods, you fill in the appropriate information for them also.

When you import your partner's profile and open the Partner Profile window Inbound Transport tab, you see your partner's transport choices. You know that your partner is prepared to receive documents from you by way of any of the transport methods that your partner has indicated. Although your partner might have indicated two or more transports, you choose only the one you want to use to send documents to that partner. The one you select to send documents to your partner does not have to match the one your partner selects to send documents to you. Obviously, you would not choose a transport method that your partner has not made available to you.

Your choice of transports involves the security you want to apply to the documents you exchange with this partner, except for one important advisory. When you change settings on the Partner Profile window Security tab, you must coordinate the changes with your trading partner.

You and your partner must have the following settings selected in the Partner Profile Security tab to ensure that your document trading is EDI-INT-compliant:

- Sign documents
- Request acknowledgment of documents
- Request signed acknowledgment
- Encrypt documents.

You and your trading partners might want to change security settings based on what your systems can support. If you and your partner use TPI, you need to ensure that all your security settings are identical. If your trading partner uses other S/MIME- or EDI-INT-certified software, you need to coordinate and test the changes.

1.8.5 Document sizes

TPI has no limitations on maximum sizes of documents, whether inbound or outbound. Limitations on document sizes rest solely on you and your partners' hardware resources and software configurations for various transport methods. A document that one organization considers to be large might be small by another organization's standards. Your organization might have to conduct its own capacity tests to determine the optimal transport for your trading situation.

This depends entirely on the hardware resources and the transport that is used. If the transport is the Internet, for instance HTTP, SMTP or FTP, it depends entirely on the bandwidth and the speed of the Internet.

1.8.6 Transports

TPI Servers need transports to exchange documents between them. This is the medium in which the documents are transferred from one TPI Server to another. TPI supports all of the following transports:

- Bundled e-mail inbound transport
- Bundled HTTP inbound transport
- Bundled HTTPS inbound transport
- File system inbound transport
- FTP inbound transport
- HTTP inbound transport
- HTTPS inbound transport
- WebSphere MQ inbound transport
- JMS inbound transport
- Standard e-mail inbound transport

You need to use at least one transport to exchange documents between two TPI Servers. Choosing a transport depends entirely on the transaction rate and the size of the documents, and of course on the cost.

1.9 Internet references

More information about the different EDI products from IBM, the WebSphere platform in general, and about EDI standards can be found at the following Web sites:

- WebSphere Data Interchange

  `http://www.ibm.com/websphere/datainterchange`
- IBM Information Exchange

  `http://ieas.services.ibm.com/ie/index.shtml`
- Expedite family

  `http://edi.services.ibm.com/expedite`
- Business Exchange Internet transfer

  `http://ieas.services.ibm.com/ide/index.shtml`
- WebSphere software platform

  `http://www-3.ibm.com/software/info1/websphere`
- WebSphere Business Integration Family

  `http://www.ibm.com/websphere/integrationinfo/`
- EDI standards organizations

  `http://www.disa.org`
  `http://www.unece.org/trade/untdid/welcome.htm`

2

Implementing iSoft P2PAgent

In this chapter, we describe the implementation of iSoft P2PAgent in an environment with multiple trading partners. We look at issues such as routing and integration with EDI translation software, such as WebSphere Data Interchange. Also, the integration with WebSphere Business Integration Server is being discussed as well.

2.1 Business scenario

During this chapter, we will step through the implementation of iSoft at a company, named Supplier, that manufactures products for a number of retailers. These companies have decided to implement an AS2 solution whereby EDI documents are exchanged using Internet technologies. As AS2 client software, these companies are using iSoft P2PAgent (Express, Advanced or Enterprise Edition), which is also available from IBM.

The integration of iSoft with the back-end infrastructure of an enterprise can be file based or queue based. For outbound communication, this means that an EDI document can be handed over to iSoft either as file in a pre-configured directory, or as a message in a pre-configured queue of WebSphere MQ. Note that the integration of iSoft for one company has no effect on the other companies. This means that the company Supplier might use queue-based integration on their side while the company Retailer3 has a file-based integration.

For demonstration purposes, the setup for the company Supplier includes both file-based and queue-based integration. This setup will prove that you are not forced to make one or the other integration. The implementation of iSoft will be described from the viewpoint of the company Supplier. An initial implementation will be done for establishing communication between the company Supplier and a single retailer. This setup will then be expanded to exchange documents with more than one trading partner. Figure 2-1 shows an overall diagram for this first step of the implementation of iSoft.

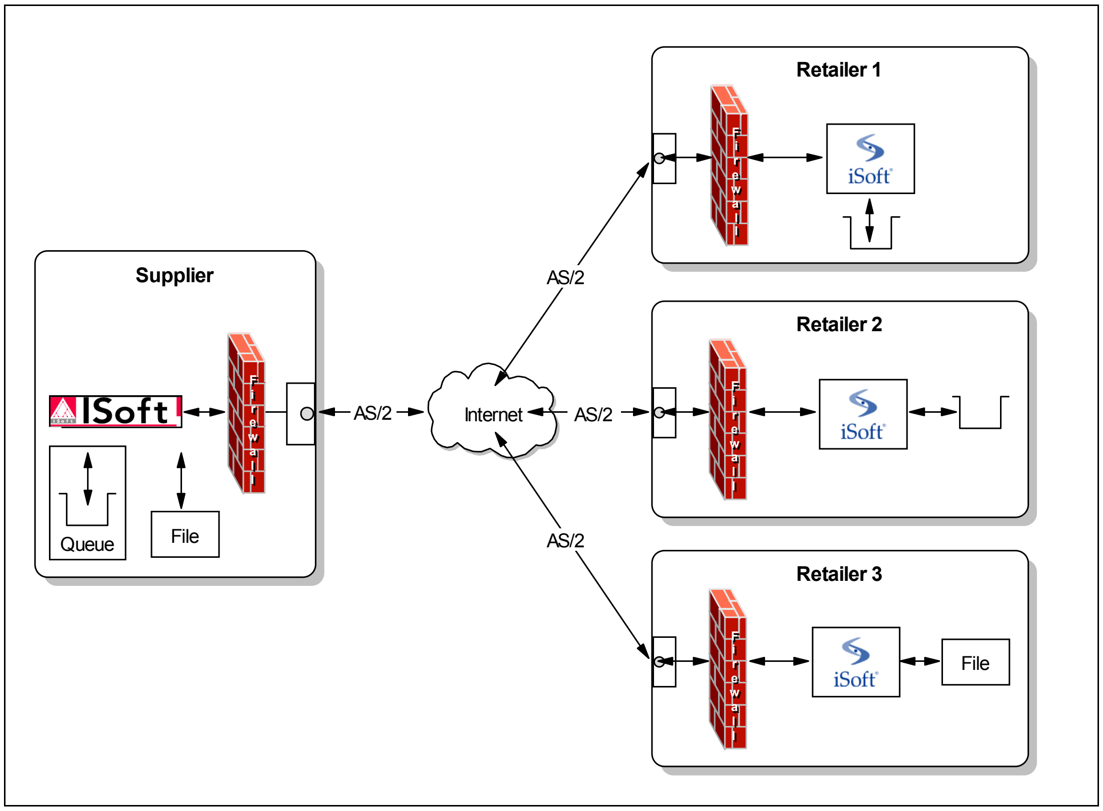


Figure 2-1 Implementation of iSoft for a company Supplier and a number of retailers

The next step during the implementation of iSoft at company Supplier is to focus on the integration of this B2B product with the overall IT infrastructure of the company. Two tasks are identified here:

- The translation of documents in an internal format into EDI documents, and vice-versa.
- Connecting the EDI data stream into the overall integration engine implemented by the InterChange Server.

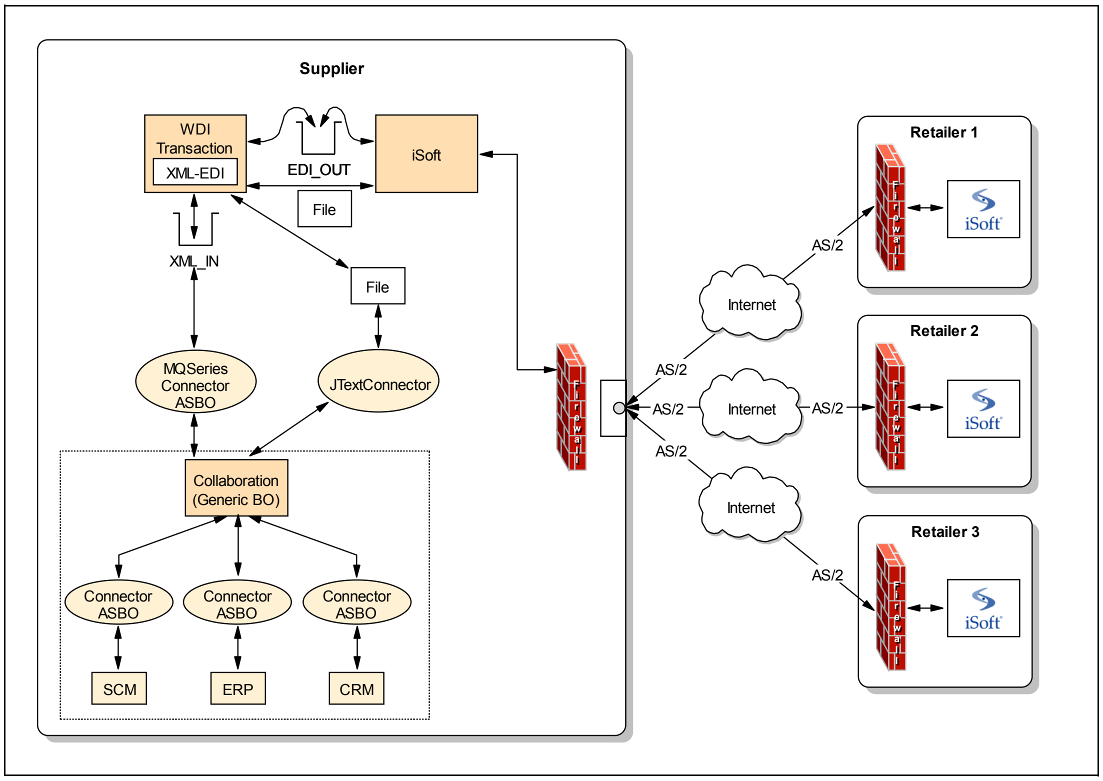


Figure 2-2 Integrating iSoft with WebSphere Data Interchange and Interchange Server

Figure 2-2 shows the overall data flow. Within the ICS, we assume an existing integration collaboration that connects to internal applications that manage supply chain, resource planning, and customer relationships. To extend such a collaboration with iSoft integration, we could use an MQSeries connector or a JText connector. The data that is being passed via these connectors flows to the EDI translation engine WebSphere Data Interchange. Based on the setup of WebSphere Data Interchange, the translation engine then generates an EDI document in a file or in a queue, from where it is being picked up by the P2PAgent program.

For inbound communication, the iSoft P2PAgent program drops the EDI document in a directory or in a queue, where the translation engine is retrieving it. The translation engine produces then a new document in an internal format and stores that either as a message in a queue or as a file in the directory. For message-based output, we can again use an MQSeries connector to retrieve that message and hand it over to an integration collaboration. For file-based output of the translation engine, we can again use the JText connector.

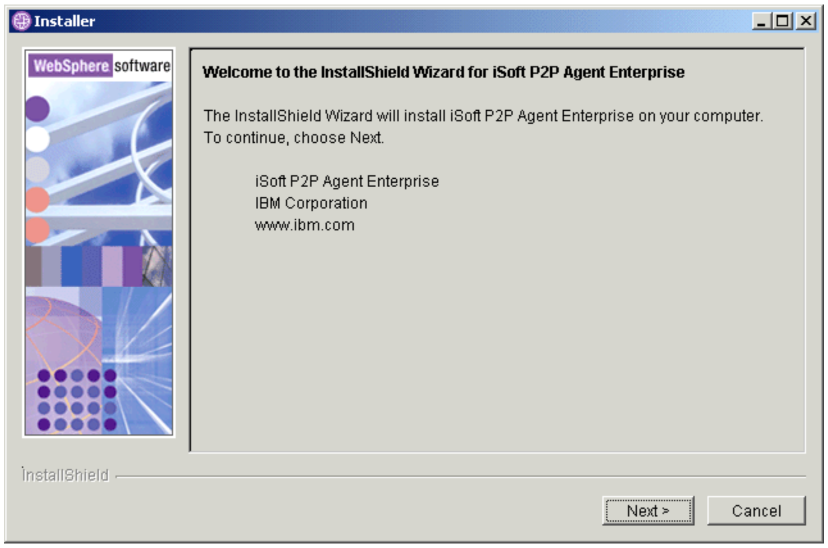
Examples of these integration scenarios are covered in 2.3, “Integration with WebSphere Data Interchange” on page 65 and 2.4, “Integration with the Interchange Server” on page 83.

2.2 Basic implementation of iSoft

Before we can discuss any integration scenarios, we need to create a basic implementation of iSoft, which is the subject of this section.

2.2.1 Installation and initial configuration

The AS2 client software is available in a few different packages from IBM or iSoft itself and with different licensing restrictions. In some packages you may or may not find sample files and documentation. However, in all cases, the installation of the software is straight forward. Some steps below may not be required since they were done during the installation itself.



Welcome to the InstallShield Wizard for iSoft P2P Agent Enterprise

The InstallShield Wizard will install iSoft P2P Agent Enterprise on your computer.
To continue, choose Next.

iSoft P2P Agent Enterprise
IBM Corporation
www.ibm.com

Figure 2-3 Installer for iSoft P2P Agent Enterprise edition

The P2PAgent program can use queues and/or files for several functions. To set up an initial environment, you should create a number of queues and/or directories to support those functions.

Example 2-1 lists a number of WebSphere MQ commands that you can use to create queues for use by the P2PAgent program. These queues can be used in the configuration file as described below.

Example 2-1 WebSphere MQ commands to create supporting queues

```
def ql('notices') defpsist(yes)
def ql('receipts’) defpsist(yes)
def ql('workorders') defpsist(yes)
def ql('outbox') defpsist(yes)
def ql('inbox') defpsist(yes)
def ql('log') defpsist(yes)
def ql('errors') defpsist(yes)
```

In case you want a file-based integration for some or all of the iSoft functions, you should create the following directories. Again, these directories are used in the configuration file described below.

- inbox
- outbox
- receipt
- workorders
- pki
- errors
- log
- notices

The P2PAgent's operations are controlled by a configuration file. The default location of this file is the install directory and the default name of this file is p2pagent.cfg. Note that it is possible to have a different name and location. You would then need to use start-up parameters to provide this information. Throughout this redbook, we'll use the default name and location. The configuration file has an XML format and is basically a sequence of commands that are interpreted by the P2PAgent program at start-up time. Note again that these commands are the same as the interactive commands or the work order commands.

For all settings related to file locations and directories, you can replace the name of a directory with an mq URI, in the format of `mq://queue_manager_name/queue_name`, unless it is said that only directory names are supported.

- Define the location to store error information

  To store error information in files in a directory named errors, use the following command:

  ```
  set -eperrors -ef
  ```

- Define the location to store logging information

  To store log information in files in a directory named log, use the following command:

  ```
  set -lplog -lf
  ```

  The name of the log file is P2PYYYYMMDD.log, where YYYYMMDD is the current day. This implies that the P2PAgent program will automatically create a new file for each day and append information to it. The contents of this log file is basically the same as the standard output of the agent program.

- Define the location to store notices

  Notices are files or messages that indicate the results of document transactions.

  ```
  set -npmq://cw_studenta.queue.manager/notices -nf
  ```

- Define the location of certificates and private keys.

  The PKI service component of the P2PAgent component will look in this directory to find any keys and certificates. The command below identifies this directory as pki.

  ```
  set -pppki
  ```

- Define the location to store receipts

  Receipts are the documents received by the sending agent when the receiving agent confirms the delivery of a given EDI document. In most case, such a receipt document will be signed using a digital signature algorithm. Receipts are delivered asynchronously.

  ```
  set -rpmq://cw_studenta.queue.manager/receipts
  ```

- Set time-out values

  The P2PAgent allows you to configure several types of time-outs. The command below sets the time-out for a first receive after accepting an inbound connection. For more information about other types of time-out values, refer to the iSoft product documentation.

  ```
  set -tr300s
  ```

- Set mailbox information

  Mailboxes can be file-based or queue-based or even stored in a database. The command below identifies a queue manager as the host of the mailboxes. Note that the location of a mailbox can still be set at the individual trading partner level.

  ```
  set -bhmq://cw_studenta.queue.manager
  ```

- Start the inbound service

  This command starts the inbound service, to listen on port 4080 for hostname studenta.

  ```
  start http://studenta:4080
  ```

There are more options and settings that can be controlled in the configuration file, but for an initial deployment, these values are sufficient. Besides a number of set commands, a configuration file typically contains a number of addpair commands and importkey commands. The addpair command defines a relationship between two symbolic trading partner names. The structure of the addpair command is as following:

```
addpair <from> <to> <to-URL> <rcpt-URL> <notify-name> <mailbox>
```

Applied to this first occurrence of the addpair command in Example 2-2, this means:

> When processing documents sent by partner SUP1 to partner RETAILER1, send this to the URL http://studentb:4080. Request that receipts are sent to the URL http://studenta:4080 and address those receipts to partner SUP1.

The second occurrence of the addpair command in Example 2-2 applies to the processing of incoming documents. Here, the rcpt-URL parameter is set to * to indicate that we do not override the settings that is requested by the sender of the document.

The importkey command assigns certificates and/or private keys to a trading partner relationship for a specific function, such as encryption and signing. While there are again many possibilities, a typical scenario is expressed in Example 2-2. The syntax of the importkey command is shown below:

```
importkey <from> <to> <usage> <options>
```

`From` and `To` identify the trading partner relationship. Usage is a one character code that identifies when the certificate and/or private key should be used. The `options` are used to identify the key and certificate.

The first command instructs the P2PAgent program that documents from SUP1 to RETAILER1 are to be signed and decrypted (option E) using the private key and certificate of the partner SUP1. The signing here relates to documents sent to RETAILER1. The decrypting relates to the decryption of MDNs received from RETAILER1 as a result of sending documents to RETAILER1.

The encryption for documents sent from SUP1 to RETAILER1 (option J) is done using the certificate of partner RETAILER1, as shown in the second importkey command. In practise this means that:

- SUP1 sends documents to RETAILER1 in a way that only RETAILER1 can read this, since we can assume that only RETAILER1 has access to its own private key.
- SUP1 sends documents to RETAILER1 in a way that SUP1 can never deny that it has sent that document. SUP1 will sign the document using its private key.

The reverse commands are needed too to control how documents should be received by SUP1 when sent by RETAILER1.

Example 2-2 P2P agent configuration file for one bi-directional communication link

```
<xml>
<command>set -eperrors -ef </command>
<command>set -lplog -lf </command>
<command>set -npmq://cw_studenta.queue.manager/notices    -nf </command>
<command>set -opmq://cw_studenta.queue.manager/workorders -of -oswo </command>
<command>set -pppki </command>
<command>set -rpmq://cw_studenta.queue.manager/receipts </command>
<command>set -tr300s </command>
<command>set -bhmq://cw_studenta.queue.manager </command>
```

```
<command>addpair   SUP1 RETAILER1 http://studentb:4080/      http://studenta:4080/ SUP1
mq://cw_studenta.queue.manager/outbox</command>
<command>addpair   RETAILER1 SUP1 http://studenta:4080/      *                    RETAILER1
mq://cw_studenta.queue.manager/inbox</command>
<command>importkey SUP1 RETAILER1 E -fCpki\SUP1-RETAILER1.cer -fKpki/SUP1-RETAILER1.prv</command>
<command>importkey SUP1 RETAILER1 J -fCpki\RETAILER1-SUP1.cer </command>
<command>importkey RETAILER1 SUP1 E -fCpki\SUP1-RETAILER1.cer -fKpki/SUP1-RETAILER1.prv</command>
<command>importkey RETAILER1 SUP1 J -fCpki\SUP1-RETAILER1.cer </command>

<command>start http://studenta:4080</command>
</xml>
```

This completes the first step of the configuration and you can now start-up the agent. Open a command window and switch to the directory that hold the configuration file (shown in Example 2-2 on page 54) and the program executable. Start the program and you should see output similar to the output shown in Example 2-3.

Example 2-3 Standard output of first start-up of the P2PAgent program

```
C:\iSoft_Enterprise>p2pagent_odbc_ibm_enterprise.exe
iSoft(R) Peer-to-Peer Agent(TM) for MQSeries(R)
(C) Copyright 2001-2002 iSoft Corp.
Build: 3.1.2002.10.30.1 [Nov 27 2002 15:08:00]
IBM Enterprise Edition
Authorized License
2003.01.27 13:25:43.299      POPT OK  Error path set to [error]
2003.01.27 13:25:43.299      POPT OK  Inbound errant will be stored
2003.01.27 13:25:43.309      POPT OK  Log path set to [log]
2003.01.27 13:25:43.309      POPT OK  Trace set to WRITE_FILE
2003.01.27 13:25:43.329      POPT OK  Notice path set to [mq://cw_studenta.queue.manager/notices]
2003.01.27 13:25:43.329      POPT OK  Notices will be written to file
2003.01.27 13:25:43.349      POPT OK  Work-order path set to [mq://cw_studenta.queue.manager/workorders]
2003.01.27 13:25:43.349      POPT OK  Work-order searching enabled
2003.01.27 13:25:43.349      POPT OK  Work-order file-spec set to [wo]
2003.01.27 13:25:43.359      POPT OK  PKI path set to [pki]
2003.01.27 13:25:43.379      POPT OK  Async. receipt path set to [mq://cw_studenta.queue.manager/receipts]
2003.01.27 13:25:43.389      POPT OK  First-receive interval set to [300000ms]
2003.01.27 13:25:43.399      POPT OK  Mailbox host set to [mq://cw_studenta.queue.manager]
2003.01.27 13:25:43.399      POPT OK  Mailbox address set to [0.0.0.0]
2003.01.27 13:25:43.409      POPT OK  Mailbox port set to [0]
2003.01.27 13:25:43.479      HPIM OK  HTTP inbound service started
2003.01.27 13:25:48.506      PIKC ERR Unable to import keys
2003.01.27 13:25:48.536      PIKC ERR Unable to import keys
2003.01.27 13:25:48.556      PIKC ERR Unable to import keys
2003.01.27 13:25:48.576      PIKC ERR Unable to import keys
```

Since we have not yet generated any keys and since we have not received any certificates from our trading partner RETAILER1, it is not surprising that the four importkey commands are failing. To generate these keys, we need to run an addkey command. However, since the addkey command only needs to be run once, it is not included in the configuration file. To generate keys, create a work order file (for example addkeys.wo) that contains a XML document, such as the one shown in Example 2-4 on page 56.

The structure of the addkey command is as following:

```
addkey <from> <to> <function> <key length> <issuer> <subject>
```

Applied to the first occurrence of the addkey command, we request to generate a key that is used for communication from SUP1 to RETAILER1 for the functions sign, encrypt, decrypt and signature verification (option O) with a key length of 1024 bits for RSA. The certificate is self-signed and has the provided subject. Since we need to set up communication for two more retailers, we will need to create keys for that relationship too. Note that we use a different identity (SUP1, SUP2, SUP3) for communicating with each retailer. This is not mandatory but it can increase the level of security.

Example 2-4 Addkeys.wo work order file

```
<xml>

# Create Keys

<command>addkey SUP1 RETAILER1 O 1024 self C=US;S=TX;L=Dallas;O=iSoft;CN=RETAILER1</command>
<command>addkey SUP2 RETAILER2 O 1024 self C=US;S=TX;L=Dallas;O=iSoft;CN=RETAILER2</command>
<command>addkey SUP3 RETAILER3 O 1024 self C=US;S=TX;L=Dallas;O=iSoft;CN=RETAILER3</command>

</xml>
```

To execute the commands of the work order, you need to execute the batch command in an interactive session of the P2PAgent program. Type `batch addkeys.wo` followed by the enter key in the command window (see Example 2-5).

Example 2-5 Output of addkeys.wo work order

```
C:\iSoft_Enterprise>p2pagent_odbc_ibm_enterprise.exe
iSoft(R) Peer-to-Peer Agent(TM) for MQSeries(R)
(C) Copyright 2001-2002 iSoft Corp.
Build: 3.1.2002.10.30.1 [Nov 27 2002 15:08:00]
IBM Enterprise Edition
Authorized License
2003.01.27 13:26:52.448       POPT OK  Error path set to [error]
2003.01.27 13:26:52.448       POPT OK  Inbound errant will be stored
2003.01.27 13:26:52.458       POPT OK  Log path set to [log]
2003.01.27 13:26:52.458       POPT OK  Trace set to WRITE_FILE
2003.01.27 13:26:52.478       POPT OK  Notice path set to [mq://cw_studenta.queue.manager/notices]
2003.01.27 13:26:52.478       POPT OK  Notices will be written to file
2003.01.27 13:26:52.498       POPT OK  Work-order path set to [mq://cw_studenta.queue.manager/workorders]
2003.01.27 13:26:52.498       POPT OK  Work-order searching enabled
2003.01.27 13:26:52.508       POPT OK  Work-order file-spec set to [wo]
2003.01.27 13:26:52.518       POPT OK  PKI path set to [pki]
2003.01.27 13:26:52.528       POPT OK  Async. receipt path set to [mq://cw_studenta.queue.manager/receipts]
2003.01.27 13:26:52.538       POPT OK  First-receive interval set to [300000ms]
2003.01.27 13:26:52.558       POPT OK  Mailbox host set to [mq://cw_studenta.queue.manager]
2003.01.27 13:26:52.558       POPT OK  Mailbox address set to [0.0.0.0]
2003.01.27 13:26:52.569       POPT OK  Mailbox port set to [0]
2003.01.27 13:26:52.649       PIKC ERR Unable to import keys
2003.01.27 13:26:52.649       HPIM OK  HTTP inbound service started
2003.01.27 13:26:52.669       PIKC ERR Unable to import keys
2003.01.27 13:26:52.689       PIKC ERR Unable to import keys
2003.01.27 13:26:52.719       PIKC ERR Unable to import keys
batch addkeys.wo
ok
2003.01.27 13:27:18.125       PAKC OK  Keypair generated
```

Since we need to share the certificate with our trading partner RETAILER1, we need to export the key in a file. Here, we use again the concept of a work order to perform these actions. Example 2-6 on page 57 shows the Exportkeys.wo work order file for all three trading partners.

Example 2-6 Exportkeys.wo work order file

```
<xml>

# Export Keys

<command>exportkey SUP1 RETAILER1 O SUP1-RETAILER1.cer SUP1-RETAILER1.prv</command>
<command>exportkey SUP2 RETAILER2 O SUP2-RETAILER2.cer SUP2-RETAILER2.prv</command>
<command>exportkey SUP3 RETAILER3 O SUP3-RETAILER3.cer SUP3-RETAILER3.prv</command>

</xml>
```

To run these commands in an interactive session, we use again the batch command, as shown below in Example 2-7.

Example 2-7 Output of the exportkeys.wo work order

```
C:\iSoft_Enterprise>p2pagent_odbc_ibm_enterprise.exe
iSoft(R) Peer-to-Peer Agent(TM) for MQSeries(R)
(C) Copyright 2001-2002 iSoft Corp.
Build: 3.1.2002.10.30.1 [Nov 27 2002 15:08:00]
IBM Enterprise Edition
Authorized License
2003.01.27 13:26:52.448       POPT OK  Error path set to [error]
2003.01.27 13:26:52.448       POPT OK  Inbound errant will be stored
2003.01.27 13:26:52.458       POPT OK  Log path set to [log]
2003.01.27 13:26:52.458       POPT OK  Trace set to WRITE_FILE
2003.01.27 13:26:52.478       POPT OK  Notice path set to [mq://cw_studenta.queue.manager/notices]
2003.01.27 13:26:52.478       POPT OK  Notices will be written to file
2003.01.27 13:26:52.498       POPT OK  Work-order path set to [mq://cw_studenta.queue.manager/workorders]
2003.01.27 13:26:52.498       POPT OK  Work-order searching enabled
2003.01.27 13:26:52.508       POPT OK  Work-order file-spec set to [wo]
2003.01.27 13:26:52.518       POPT OK  PKI path set to [pki]
2003.01.27 13:26:52.528       POPT OK  Async. receipt path set to [mq://cw_studenta.queue.manager/receipts]
2003.01.27 13:26:52.538       POPT OK  First-receive interval set to [300000ms]
2003.01.27 13:26:52.558       POPT OK  Mailbox host set to [mq://cw_studenta.queue.manager]
2003.01.27 13:26:52.558       POPT OK  Mailbox address set to [0.0.0.0]
2003.01.27 13:26:52.569       POPT OK  Mailbox port set to [0]
2003.01.27 13:26:52.649       PIKC ERR Unable to import keys
2003.01.27 13:26:52.649       HPIM OK  HTTP inbound service started
2003.01.27 13:26:52.669       PIKC ERR Unable to import keys
2003.01.27 13:26:52.689       PIKC ERR Unable to import keys
2003.01.27 13:26:52.719       PIKC ERR Unable to import keys
batch addkeys.wo
ok
2003.01.27 13:27:18.125       PAKC OK  Keypair generated
batch exportkeys.wo
ok
2003.01.27 13:27:45.645       POKC OK  Key-pair exported
```

At this time, stop the P2PAgent program using the shutdown command. Assuming that a similar setup was done at the side of RETAILER1, you can now exchange certificates and store that certificate in the pki directory. Make sure that the filename of the certificate matches with what you have set in the configuration file. The configuration file in Example 2-2 on page 54 assumes a name of RETAILER1-SUP1.cer for the certificate received from RETAILER1 and to be used by SUP1.

You can now restart the P2PAgent at both sides. This time, the output should not any more contain any errors, as shown below in Example 2-8 on page 58. The setup is now complete and we can now validate the setup by sending some documents from SUP1 to RETAILER1 and vice versa.

Example 2-8 Standard output of restarted P2PAgent program

```
C:\iSoft_Enterprise>p2pagent_odbc_ibm_enterprise.exe
iSoft(R) Peer-to-Peer Agent(TM) for MQSeries(R)
(C) Copyright 2001-2002 iSoft Corp.
Build: 3.1.2002.10.30.1 [Nov 27 2002 15:08:00]
IBM Enterprise Edition
Authorized License
2003.01.27 13:32:54.209      POPT OK  Error path set to [error]
2003.01.27 13:32:54.209      POPT OK  Inbound errant will be stored
2003.01.27 13:32:54.219      POPT OK  Log path set to [log]
2003.01.27 13:32:54.229      POPT OK  Trace set to WRITE_FILE
2003.01.27 13:32:54.239      POPT OK  Notice path set to [mq://cw_studenta.queue.manager/notices]
2003.01.27 13:32:54.239      POPT OK  Notices will be written to file
2003.01.27 13:32:54.249      POPT OK  Work-order path set to [mq://cw_studenta.queue.manager/workorders]
2003.01.27 13:32:54.259      POPT OK  Work-order searching enabled
2003.01.27 13:32:54.259      POPT OK  Work-order file-spec set to [wo]
2003.01.27 13:32:54.269      POPT OK  PKI path set to [pki]
2003.01.27 13:32:54.289      POPT OK  Async. receipt path set to [mq://cw_studenta.queue.manager/receipts]
2003.01.27 13:32:54.299      POPT OK  First-receive interval set to [300000ms]
2003.01.27 13:32:54.309      POPT OK  Mailbox host set to [mq://cw_studenta.queue.manager]
2003.01.27 13:32:54.309      POPT OK  Mailbox address set to [0.0.0.0]
2003.01.27 13:32:54.319      POPT OK  Mailbox port set to [0]
2003.01.27 13:32:54.399      HPIM OK  HTTP inbound service started
```

Note: The steps above did not expand in detail how certificates are exchanged. It should be mentioned that the P2PAgent program contains a sendcert command to send certificates to trading partners. This might be a sufficient solution for your security requirements. However, it might very well be that you require a more formal exchange procedure where both partners acknowledge that certificates have been received. Procedures can vary from a simple e-mail attachment to a delivery by a courier. For the purposes of this redbook, it is sufficient to assume that an exchange procedure can be set up and that certificates become available for the P2PAgent program.

2.2.2 Validating the configuration

The configuration that we have described in the previous section does not yet involve the start-up of outbound communication tasks. At this time, dropping a message in the outbox queue will not result in an actual transmission. To start-up this outbound transmission task, we'll need to include a send command in the configuration file. But before we do that, we would like to perform a manual send to allow us to step through the whole flow and validate that the setup so far is correct.

The send command can have many parameters to perform different functions. Here, in Example 2-9 on page 59, the command initiates a send from partner SUP1 to partner RETAILER1. The data to be sent is stored in the current directory in a file called test.txt (-fN parameter). We ask to try to send only once (-n parameter) and we request an unsigned MDN (message disposition notification) receipt. This command can be entered in the interactive session, as shown below in Example 2-9 on page 59. The output also shows that the P2PAgent program has twice stored data in MQ. The first entry maps to the creation of a notice message in the queue notices, as configured in the configuration file. The second entry maps to the arrival of an MDN receipt by SUP1. This receipt is stored in the queue receipts as it was set in the configuration file.

Example 2-9 Perform an initial send

```
send http SUP1 RETAILER1 -fNtest.txt -n1 -r
ok
2003.01.27 14:34:40.372 73957 HPOS OK  Outbound session started - mbox=[0] batch=[0] attempt=[1 of 1]
2003.01.27 14:34:41.313 35014 STMQ OK  Data stored
2003.01.27 14:34:41.333 11530 STMQ OK  Data stored
2003.01.27 14:34:41.333 73957 HPOS OK  Outbound session stopping - batch=[0]
```

The output of the P2PAgent program at the receiving side is shown below in Example 2-10. The arrival of an inbound connection request is logged, together with the IP address that originated the connection request. This is followed by a two log entries indicating that information was stored in MQ. The first entry maps to the storage of the incoming data as an MQ message in the queue inbox. The second entry is for the notice message. The setup for RETAILER1 is similar to the setup of SUP1. Queue names, outbox and inbox configuration is exactly the same.

Note: Time stamps in Example 2-9 and Example 2-10 do not match completely, since the clock of both computers was not synchronized. However, the logging in Example 2-9 does match the logging in Example 2-10.

Example 2-10 Receiving a message at RETAILER1

```
2003.01.27 14:34:05.789 44452 HPIS OK  HTTP inbound session started
2003.01.27 14:34:05.789 44452 HPIS OK  HTTP client: 9.24.104.158:1087
2003.01.27 14:34:05.829 44452 STMQ OK  Data stored
2003.01.27 14:34:05.859 58670 STMQ OK  Data stored
2003.01.27 14:34:05.869 44452 HPIS OK  HTTP inbound session stopping
```

The previous send command resulted in a number of MQ operations, at both the sending and receiving side. But it did not yet validate access to the mailbox, which is also hosted by MQ. Therefore, we use a utility such as RFHUTIL to load the same file text.txt into the queue outbox. The send command in Example 2-11 will pick up this message and send it out to REATILER1.

The parameter -ds indicates to the P2PAgent where to find the data. This is set to be a mailbox, with identifier outbox. Note that in this case, we have not asked to request a MDN receipt of RETAILER1. As a result, there is only one MQ message stored as a result of the transactions (the notice).

Example 2-11 Validating the mailbox implementation

```
send http SUP1 RETAILER1 -dsMAILBOXID=outbox -n1
ok
2003.01.27 16:41:54.118 22768 HPOS OK  Outbound session started - mbox=[outbox] batch=[0] attempt=[1 of 1]
2003.01.27 16:41:54.148 22768 EXMQ OK  File extracted - [1533] bytes
2003.01.27 16:41:54.369 30639 STMQ OK  Data stored
2003.01.27 16:41:54.379 22768 HPOS OK  Outbound session stopping - batch=[0]
```

The above tests validated the whole setup with respect to MQ and Internet communication. We should also do a number of tests to validate encryption, digital signatures and possible compression. The following commands can be executed for these purposes.

- Sign outbound EDI file

```
send http SUP1 RETAILER1 -fNtest.txt -n1 -s -r1
```

The -s parameter signs the document using the default SHA-1 algorithm. If you would need an MD5 algorithm, use the option -s5. When you would like to use Base64 encoding for the signature, add B to the parameter (-sB and -s5B).

Previously, we had used -r to request a receipt. Here, we used -r1, to indicate that we request a signed receipt, using the SHA-1 algorithm.

- Encrypt and sign the outbound EDI file

  ```
  send http SUP1 RETAILER1 -fNtest.txt -n1 -e -s -r1
  ```

  The option -e is used to indicate that the outbound EDI document needs to be encrypted using the Triple DES algorithm. You can also use the RC2 algorithm by specifying the option -e2.

- Encrypt, sign and compress the outbound EDI file

  ```
  send http SUP1 RETAILER1 -fNtest.txt -n1 -e -s -oZ -r1
  ```

  The option -oZ indicates that the message payload needs to be compressed using the ZLIB compression algorithm. This compression is then followed by encryption and signing of the payload.

- More options to control the receipt.

  So far, we've seen the options -r and -r1 or requesting an unsigned or signed receipt. You can also use the option -r5, to request a MD5 signed receipt, instead of the default SHA-1. For each of these three cases, you can add the option A to the parameter to indicate that the receipt may be sent asynchronously (thus, -rA, -rA1, -rA5).

2.2.3 Automating the send process

So far, we've initiated the transmission of EDI documents by entering manually a send command in the interactive session of the P2PAgent program. This works fine for testing purposes, but in a real implementation, you would like that the agent sends the data without any manual intervention.

Using work orders

As part of the configuration file in Example 2-2 on page 54, we had the configuration for a source of work orders. Our sample configuration used a queue called workorders. To validate that this configuration is actually working, issue the command **`set -m6`** in the interactive session and you will see that the agent opens this queue every ten seconds to see if any work orders are outstanding. Note that the above command is merely used for debugging purposes. To switch back to normal logging, issue the command **`set -m3`**. A work order can be any type of command that the agent understands, including a send command. Thus, one option to automate the transmission of outbound documents, is to put a message on the queue workorders containing a send command, such as the ones that we have seen in 2.2.2, "Validating the configuration" on page 58. The generation of this message could be done at the same time that the EDI document is being generated. If for example the generation is being handled by WebSphere MQ Integrator or WebSphere Data Interchange, you can augment the message flow to generate a message with a send command in the correct queue.

Persistent send

The alternative to use work orders, is to include a so-called persistent send in the configuration file itself. Such a send command will contain an option that tells the agent how long the send command should persist and what the polling interval should be.

For file-based input to the P2PAgent program:

```
send http SUP1 RETAILER1 -fPoutbox -fSout -fE.pend -tE20031231000000 -s -e -r1 -n2
```

This command will:

1. Send files from SUP1 to RETAILER1
2. Use http as a protocol
3. Load the files from the directory outbox
4. Send only those files with extension out
5. Append the sent files with the string .pend
6. Keep on sending until 12/31/2003 at mid-night
7. Sign the document
8. Encrypt the document
9. Request a signed MDN
10. Try to send the document twice if the first time failed (-n2)

Instead of renaming the file (option -fE) after the transmission, you can ask to delete the file after transmission. In that case, replace the `-fE.pend` string in the command with `-x`.

You should include such a send command for all partner relationships that you have defined.

For queue-based input, the send command could be as following:

```
send http SUP1 RETAILER1 -dsMAILBOXID=outbox -tE20031231000000 -s -e -r1 -n2
```

This command behaves the same as the previous command, except for the following

- Input is now the queue outbox
- The sent message is not being saved

Choosing persistent send or work orders depends on your environment. Persisting send is probably a better solution when the transaction volume is high. For a low volume and many trading partners, the persistent send option implies many active threads that are polling the system for any new files. In that environment, the work order mechanism might be more appropriate.

2.2.4 Connecting to partners RETAILER2 and RETAILER3

So far, the setup of iSoft's P2PAgent program has been to enable document exchange between two partners. In most situations, there will be a requirement to connect tens or hundreds, sometimes even thousands of partners. In this section we will look at a number of configuration options how we can handle this kind of requirement. For the purposes of this redbook, we will not look into multi-machine setups of the P2PAgent program, which can be required to achieve high levels of throughput and/or high availability.

Assuming that the P2PAgent maintains the information that translates a trading partner ID (SUP1, RETAILER1) into a network address (IP address or qualified hostname), then it is the responsibility of the generator of the document to indicate which partner is the final recipient of that document. The generator of a document can be an EDI translation engine, such as WebSphere Data Interchange, or a message broker such as WebSphere MQ Integrator or back-end application software, such as SAP R/3. There are three levels of information that can be used to derive the trading partner name.

1. The location of the document.

   Given the example configuration file (see Example 2-2 on page 54), it is clear how we can handle this. One can easily create a separate directory for each partner. Each partner to which we would like to send documents should have its own outbox directory or outbox queue. For inbound communication, we can use a similar solution. Each sending partner could have its own inbox queue or directory on the receiving machine.

   The concept of individual inboxes and outboxes is attractive because of its simplicity, but it becomes an administrative issue when you have to communicate with thousands of trading partners. Creating and managing thousands of queues or directories is not a trivial task.

2. The meta-data information that is wrapped around the actual data.

   One obvious example of meta-data is the use of the well-known MQRFH2 header in WebSphere MQ. When iSoft's P2PAgent is requested to store an incoming document in a queue, it will automatically add an MQRFH2 header. iSoft has defined a custom usr folder with the following elements:

   i. to: holds the sending trading partner ID as it is known in iSoft
   ii. from: holds the receiving trading partner ID as it is known in iSoft
   iii. subject: a customizable subject (-u option on send command)
   iv. msgid: a generated ID that is used for correlating asynchronous MDNs

   Applications generating documents for transmission by iSoft can also generate such an MQRFH2 header and as such provide the required information for iSoft to locate network parameters based on the contents of the usr folder. However, in the current release of iSoft, this header is only validated if present but not used.

   Example 2-12 shows a portion of a message dump that includes an MQRFH2 header and the start of the ISA segment of an EDI document.

Example 2-12 Usage of the MQRF2 header by iSoft

```
00000000:  5246 4820 0200 0000 C000 0000 2202 0000 'RFH ....+...."...'
00000010:  B501 0000 2020 2020 2020 2020 0000 0000 '¦...        ....'
00000020:  B804 0000 9800 0000 3C6D 6364 3E3C 4D73 '+...ÿ...<mcd><Ms'
00000030:  643E 6A6D 735F 7465 7874 3C2F 4D73 643E 'd>jms_text</Msd>'
00000040:  3C2F 6D63 643E 3C75 7372 3E3C 6672 6F6D '</mcd><usr><from'
00000050:  3E53 5550 323C 2F66 726F 6D3E 3C74 6F3E '>SUP2</from><to>'
00000060:  5245 5441 494C 4552 323C 2F74 6F3E 3C73 'RETAILER2</to><s'
00000070:  7562 6A65 6374 3E45 4449 494E 5444 4154 'ubject>EDIINTDAT'
00000080:  413C 2F73 7562 6A65 6374 3E3C 6D73 6769 'A</subject><msgi'
00000090:  643E 3C32 3030 3330 3132 3731 3834 3230 'd><2003012718420'
000000A0:  3738 3941 4641 3244 3740 5355 5032 3E3C '789AFA2D7@SUP2><'
000000B0:  2F6D 7367 6964 3E3C 2F75 7372 3E20 2020 '/msgid></usr>   '
000000C0:  4953 412A 3030 2A20 2020 2020 2020 2020 'ISA*00*         '
000000D0:  202A 3030 2A20 2020 2020 2020 2020 202A ' *00*          *'
000000E0:  5245 2A54 4149 4C45 5232 2020 2020 2020 'RE*TAILER2      '
000000F0:  202A 5355 2A50 3220 2020 2020 2020 2020 ' *SU*P2         '
00000100:  2020 2020 2A39 3631 3030 372A 3230 3133 '    *961007*2013'
```

   Meta-data for file-based integration with iSoft could be embedded in the name of the file. As you have seen in 2.2.2, "Validating the configuration" on page 58, the send command has options to specify file extensions (-fE) or file specifications (-fS). For example, all files with destination RETAILER2 should start with RET2 and have an extension out. The applicable options would be -fEout -fSRET2*.

3. The contents of the document itself

   Many types of EDI documents include already a field or segment that addresses document routing. For example the ISA segment in an EDI X12 document contains exactly that information. At present, there is no option in iSoft's P2PAgent program to enforce that it looks up the trading partner information in the document itself.

For inbound communication, iSoft P2PAgent can quite happily drop every received document in the same location. But then it is up to the document processing application to match the document with a trading partner. For EDI documents that are processed by WebSphere Data Interchange, there is little problem if all received documents are stored in the same inbox.

The three options for handling routing information are mainly discussing the outbound aspect. Given a document to be sent, how does iSoft know to whom to send it? For inbound communication, any requirements on the location of the document or the name of the document depends on what application is going to process the incoming document. If for example the incoming document is first being picked up by WebSphere Data Interchange, then there is no requirement at all. WebSphere Data Interchange can handle the MQRFH2 that is being generated by iSoft or it can look directly at the ISA segment. For XML documents, it is possible in WebSphere Data Interchange to tell it what element contains partner information and to use that information in any downstream processing of the document. If the processing application is the InterChange Server and the iSoft Connector, then the information in the MQRFH2 is available to make any decisions about further processing.

Now that we have a broader understanding of document routing and its implications towards the implementation of iSoft, let us go back to our business scenario as outlined in 2.1, "Business scenario" on page 50. The setup for iSoft at RETAILER2 is similar to what we have described for RETAILER1. The configuration file needs a number of small changes, such as queue manager name and hostname, besides the obvious change from RETAILER1 to RETAILER2. As discussed before, we need to generate keys, export them and exchange them. For RETAILER3, we have chosen a complete file-based integration of iSoft. Example 2-13 lists the configuration file as it is used at the iSoft machine for RETAILER3.

Note: Some packages of the iSoft P2PAgent contain a version of the product that will always try to load WebSphere MQ modules, even if your configuration file does not refer to any WebSphere MQ objects. This is not a problem if you have WebSphere MQ installed. However, if you do not have WebSphere MQ, the start of the P2PAgent program will fail. If you happen to have such a version, please contact IBM Support to obtain an updated version of the software.

Example 2-13   File-based setup of the agent for Retailer3

```
<xml>
<command>set -eperror                     -ef       </command>
<command>set -lplog                       -lf       </command>
<command>set -npnotices    -nf       </command>
<command>set -opworkorders -of -oswo </command>

<command>set -pppki                                 </command>
<command>set -rpreceipts             </command>
<command>set -tr300s                                </command>

<command>addpair   RETAILER3 SUP3 http://studenta:4080/ http://ispddemo:4080/ RETAILER3 outbox</command>
<command>addpair   SUP3 RETAILER3 http://ispddemo:4080/      *                    SUP3 inbox</command>
<command>importkey RETAILER3 SUP3 E -fCpki\RETAILER3-SUP3.cer -fKpki/RETAILER3-SUP3.prv</command>
<command>importkey RETAILER3 SUP3 J -fCpki\SUP3-RETAILER3.cer </command>
<command>importkey SUP3 RETAILER3 E -fCpki\RETAILER3-SUP3.cer -fKpki/RETAILER3-SUP3.prv</command>
<command>importkey SUP3 RETAILER3 J -fCpki\RETAILER3-SUP3.cer </command>
```

```
<command>start http://ispddemo:4080</command>
</xml>
```

Most changes are of course to be made at the side of Supplier. Besides the routing discussion we had before, there is another decision to be made. Do we use the same identity to interact with each retailer or do we use a different one? Example 2-4 on page 56 gave already an indication that we will chose to option to have different identities for each retailer. This is again one of those issues that can lead to endless discussions and for which there is no single answer. All options have their strengths and weaknesses. In the scenario described in this redbook, we chose to have one matching ID for each trading partner and each ID would have its own private key and certificate. Indeed, this means a lot more work. More keys need to be generated and managed. Likely more information about partner IDs needs to be propagated to upstream applications, such as WebSphere Data Interchange and the ICS. The advantage is that there is higher level of security. Assuming that you and your partners exchange encrypted and signed documents, that means that a network intruder needs

- The private key of the receiver to decrypt and read an intercepted message
- The certificate of the receiver to create an encrypted message himself
- The private key of the sender to sign the message that he created himself

Thus, an intruder needs to get access to several pieces of information to break the security. But if he succeeds and if a single certificate and key is used with all trading partners, then suddenly the intruder can play around with all your partners and cause damage at a much wider scale. In real networks, trading partner networks can span thousands of companies and you can not assume that nothing will ever happen. Having certificates for each partner limits the risks to a single pair of trading partners. Having a shared certificate makes the whole network at risk.

Given the decision to create unique identities SUP1, SUP2 and SUP3, we need to create a new set of keys and export those keys. Sample commands for these tasks were given before in Example 2-4 on page 56 and Example 2-6 on page 57. Then, new partner pairs need to be added to the configuration file. Example 2-14 lists this updated configuration file. The section for the relationship SUP1 and RETAITER1 is basically the same. The name of the outbox queue has been changed to outbox.retailer1. The relationship record for SUP2 and RETAILER2 is similar. This time, the name of the outbox queue is outbox.retailer2. For the SUP3 and RETAILER3 combination, we chose a file-based integration, as a proof that you can combine both types of integration within the same instance of the P2PAgent program.

Finally, you should not forget to define the additional queues and directories:

- Queues outbox.retailer2 and outbox.retailer3
- Directories inbox\retailer3 and outbox\retailer3

Note also that the inbox for retailer1 and retailer2 is shared (the queue inbox), while the naming of the inbox directory suggests that it will be used for receiving documents from retailer3 only.

Example 2-14 Extended configuration file for the company Supplier

```
<xml>
<command>set -eperror -ef       </command>
<command>set -lplog -lf       </command>
<command>set -npmq://cw_studenta.queue.manager/notices    -nf       </command>
<command>set -opmq://cw_studenta.queue.manager/workorders -of -oswo </command>

<command>set -pppki </command>
<command>set -rpmq://cw_studenta.queue.manager/receipts </command>
<command>set -tr300s </command>
```

```
<command>set -bhmq://cw_studenta.queue.manager </command>

<command>addpair   SUP1 RETAILER1 http://studentb:4080/ http://studenta:4080/ SUP1
mq://cw_studenta.queue.manager/outbox.retailer1</command>
<command>addpair   RETAILER1 SUP1 http://studenta:4080/ * RETAILER1
mq://cw_studenta.queue.manager/inbox</command>
<command>importkey SUP1 RETAILER1 E -fCpki\SUP1-RETAILER1.cer -fKpki/SUP1-RETAILER1.prv</command>
<command>importkey SUP1 RETAILER1 J -fCpki\RETAILER1-SUP1.cer </command>
<command>importkey RETAILER1 SUP1 E -fCpki\SUP1-RETAILER1.cer -fKpki/SUP1-RETAILER1.prv</command>
<command>importkey RETAILER1 SUP1 J -fCpki\SUP1-RETAILER1.cer </command>

<command>addpair   SUP2 RETAILER2 http://vdputteg:4080/ http://studenta:4080/ SUP2
mq://cw_studenta.queue.manager/outbox.retailer2</command>
<command>addpair   RETAILER2 SUP2 http://studenta:4080/ * RETAILER2
mq://cw_studenta.queue.manager/inbox</command>
<command>importkey SUP2 RETAILER2 E -fCpki\SUP2-RETAILER2.cer -fKpki/SUP2-RETAILER2.prv</command>
<command>importkey SUP2 RETAILER2 J -fCpki\RETAILER2-SUP2.cer </command>
<command>importkey RETAILER2 SUP2 E -fCpki\SUP2-RETAILER2.cer -fKpki/SUP2-RETAILER2.prv</command>
<command>importkey RETAILER2 SUP2 J -fCpki\SUP2-RETAILER2.cer </command>

<command>addpair   SUP3 RETAILER3 http://ispddemo:4080/ http://studenta:4080/ SUP3
outbox\retailer3</command>
<command>addpair   RETAILER3 SUP3 http://studenta:4080/ * RETAILER3 inbox\retailer3</command>
<command>importkey SUP3 RETAILER3 E -fCpki\SUP3-RETAILER3.cer -fKpki/SUP3-RETAILER3.prv</command>
<command>importkey SUP3 RETAILER3 J -fCpki\RETAILER3-SUP3.cer </command>
<command>importkey RETAILER3 SUP3 E -fCpki\SUP3-RETAILER3.cer -fKpki/SUP3-RETAILER3.prv</command>
<command>importkey RETAILER3 SUP3 J -fCpki\SUP3-RETAILER3.cer </command>

<command>start http://studenta:4080</command>
</xml>
```

Once these changes have been implemented at Supplier and the iSoft implementations have been completed for Retailer2 and Retailer3, you can again use the send command to validate the setup if all possible directions. Use the commands discussed in 2.2.2, "Validating the configuration" on page 58 to test the communication, encryption and signatures.

2.3 Integration with WebSphere Data Interchange

Business documents such as purchase orders and invoices are in general created, managed, and stored in back-end systems that are very much enterprise specific. As a result, the structure and data format of those documents can be very different from what industry organizations have agreed upon as a standard. An EDI 850 document, for purchase orders, has a specific layout that internal applications can not always generate. This mismatch between the internal document and the standard document to be used in an B2B transactions has resulted in the development of EDI specific document translators. An example of such a product is WebSphere Data Interchange.

Since the use of EDI translators is very common, a typical implementation of iSoft's P2PAgent involves the integration with products such as WebSphere Data Integration.

Throughout this section, we are assuming that you have a working WebSphere Data Interchange environment with at least one client and one server. Also, we are assuming that the server is running on the same machine as the P2PAgent program.

For more information about configuring a WebSphere Data Interchange environment, refer to the Redpaper *WebSphere Data Interchange installation and configuration*, REDP3600.

2.3.1 Translating received EDI documents

In this section we are going to describe the flow of inbound documents, where documents are received by the P2PAgent program and stored in queues or files. WebSphere Data Interchange picks up these documents and performs the required translation into an internal format, which happen to be an XML format.

Figure 2-4 shows the overall data flow. Data that is received from retailer1 or retailer2 arrives in a queue, while data received from retailer3 arrives in a file in a given directory. Note that the configuration of the P2PAgent in Example 2-14 on page 64 had the same queue for incoming documents for both retailer1 and retailer2. Figure 2-4 assumes that each retailer has its own queue. From a WebSphere Data Interchange point of view, there is little difference between both types of setup. WebSphere Data Interchange can find the appropriate trading partner information in the ISA segment of the incoming EDI document and use that information to apply the correct translation rules.

The configuration of WebSphere Data Interchange involves the following steps:

- Definition of trading partner profiles for all retailers and the supplier itself
- Document definition
  - Import of the EDI 850 document definition
  - Import of a DTD, matching the internal representation of a purchase order
- Definition of the translation map
- Definition of mailboxes, network profiles, queue profiles and service profiles
- Definition of the rules associated with the map
- Creating a command file to process incoming EDI files
- Definitions of queue and process objects in WebSphere MQ to support the automatic translation of incoming EDI documents in queues.

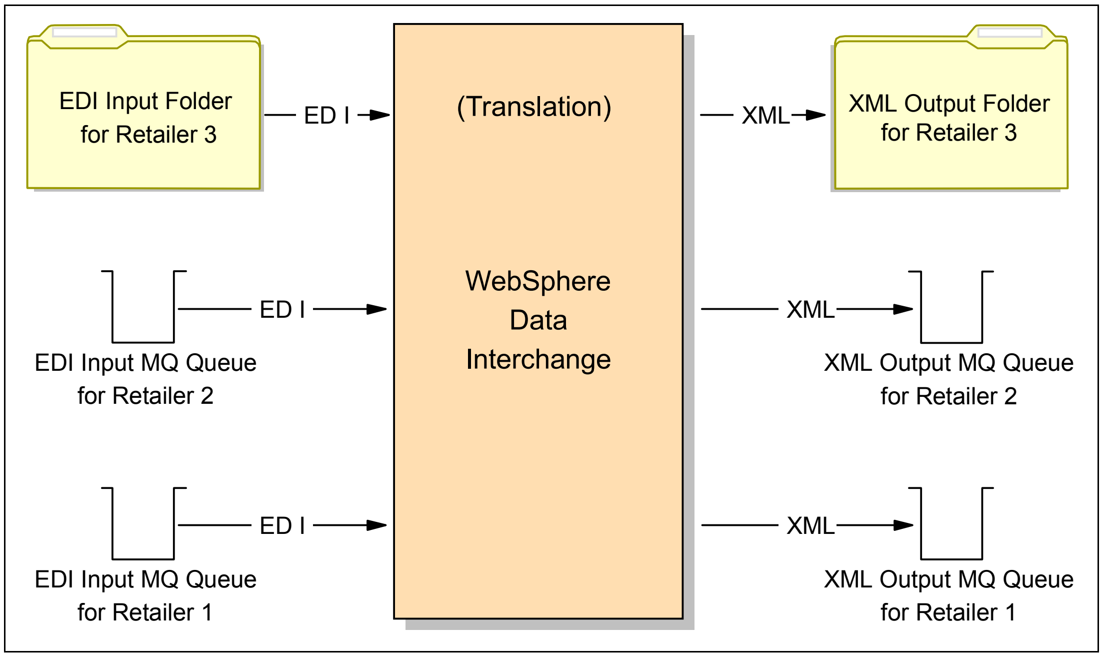

EDI Input Folder for Retailer 3
ED I
(Translation)
XML
XML Output Folder for Retailer 3
ED I
WebSphere Data Interchange
XML
EDI Input MQ Queue for Retailer 2
XML Output MQ Queue for Retailer 2
ED I
XML
EDI Input MQ Queue for Retailer 1
XML Output MQ Queue for Retailer 1

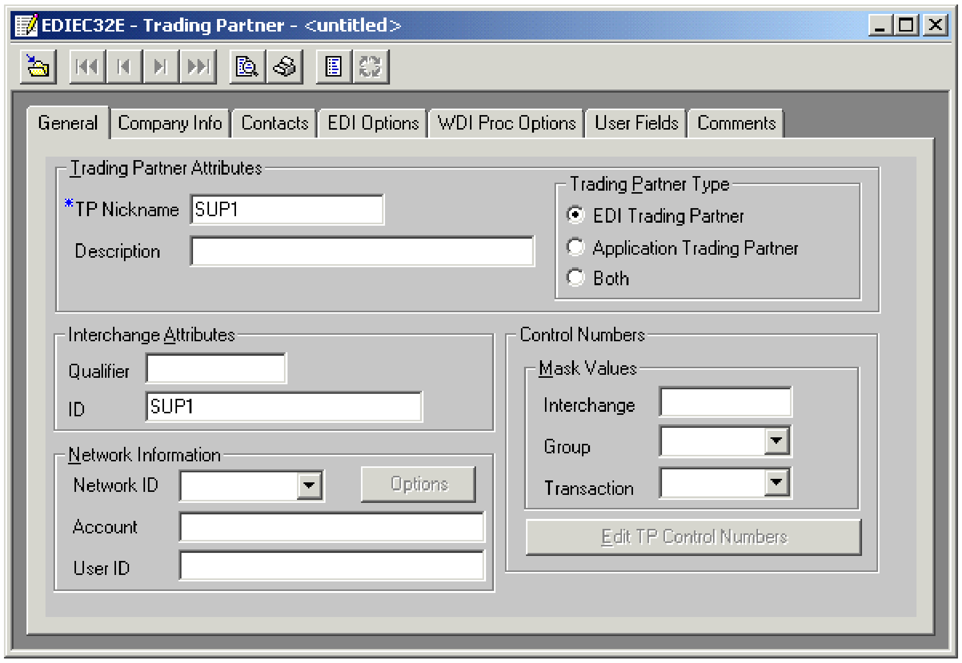
Figure 2-4 Inbound data flow

Trading partner setup

Start the WebSphere Data Interchange client program and click the trading partner setup icon in the tool bar. A new window will appear with a list of currently defined trading partners. Click the new document button in the tool bar. On the tab General, provide a nickname for the new trading partner (SUP1) and fill in the Interchange Qualifier and ID (see Figure 2-5), which are found in the EDI document. These fields in the ISA segment will be used by WebSphere Data Interchange as a key to locate the trading partner document (nickname) and that nickname will then be used in further processing within WebSphere Data Interchange. The value for the nickname is only relevant within the scope of WebSphere Data Interchange.

Repeat this process for the trading partners Retailer1, Retailer2 and Retailer3. Create profiles also for the secondary IDs SUP2 and SUP3 that are used by the Supplier (refer to the configuration in Example 2-14 on page 64.

Note: The creation of trading partner profiles will be required because we have different routing requirements depending on the actual destination/origin of the document. If a document is destined for Retailer3, WebSphere Data Interchange needs to write it in a file instead of a queue. A more common reason for trading partner specific rules is the fact that you may have different translation requirements for the same EDI document, depending on the target or origin trading partner.


EDIEC32E - Trading Partner - <untitled>

General | Company Info | Contacts | EDI Options | WDI Proc Options | User Fields | Comments

Trading Partner Attributes
*TP Nickname: SUP1
Description:

Trading Partner Type
(●) EDI Trading Partner
( ) Application Trading Partner
( ) Both

Interchange Attributes
Qualifier:
ID: SUP1

Network Information
Network ID:
Account:
User ID:

Control Numbers
Mask Values
Interchange:
Group:
Transaction:

Figure 2-5 Trading partner definition

EDI document definition

For each EDI document standard, there are a number of versions and releases. You can download import files for ANSI X12 and other standards from the following Web site:

http://www-3.ibm.com/software/integration/appconn/wdi/downloads/

For our purposes we have used the ANSI X12 Standard Version 4 Release 3. This standard can be downloaded as an export/import file (eif) for WebSphere Data Interchange. Select **File -> Open Import File** to load the definitions in your database and point the file browser to the downloaded file X12V4R3.eif. A window of document definitions will appear that lists all the definitions that are included in this file. Select the documents that you would need, for example 850 and 855. Use the Control key to select multiple definitions. Press the enter key and select the correct system (database) to import the document definitions. Note that both the 850 and 855 documents are quite big and contain many segments and fields. As a result, the import might take a while to complete.

XML document definition

WebSphere Data Interchange supports the import of DTDs to define XML documents to the translator. In general the structure of XML documents that are used by a company's internal applications, is very specific. Since this redbook is not just about EDI translation and mapping, we're going to use a simple DTD that helps us focusing on the integration issue instead of the mapping issue. Example 2-15 lists the DTD, while Example 2-16 provides a sample message that complies with this DTD.

Example 2-15 DTD for XML document representing a purchase order

```
<?xml encoding="US-ASCII"?>

<!ELEMENT PO (Header, Detail)>

<!ELEMENT Header (FROM, TO, PONO,PODate)>
<!ELEMENT FROM (#PCDATA)>
<!ELEMENT TO (#PCDATA)>
<!ELEMENT PONO (#PCDATA)>
<!ELEMENT PODate (#PCDATA)>

<!ELEMENT Detail (QTY, ITEMNO, DESC)>
<!ELEMENT QTY (#PCDATA)>
<!ELEMENT ITEMNO (#PCDATA)>
<!ELEMENT DESC (#PCDATA)>
```

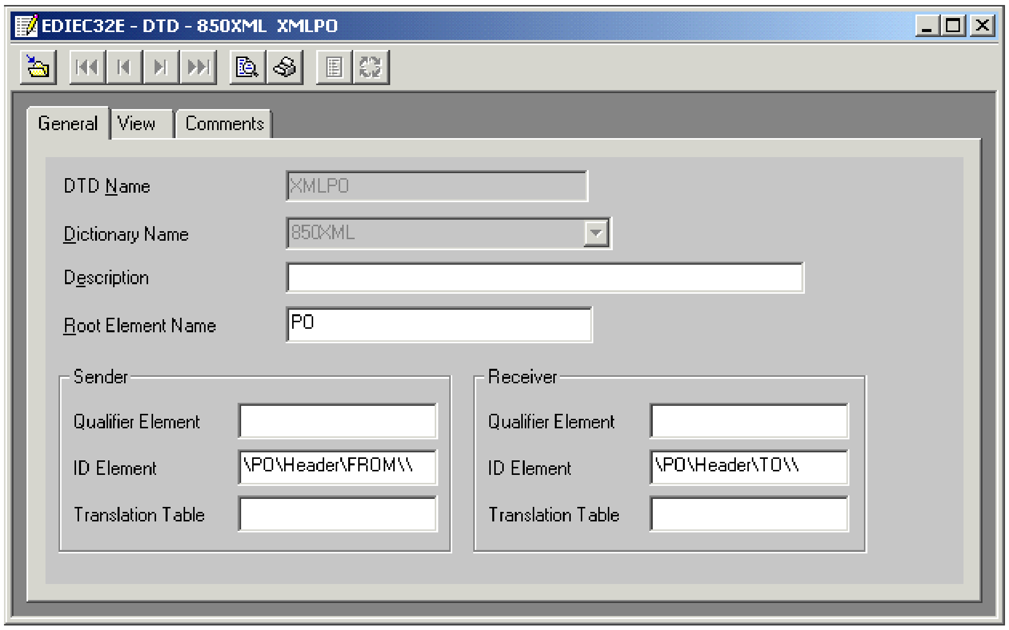
Example 2-16 Sample XML purchase order

```
<?xml version="1.0"?>
<!DOCTYPE Order SYSTEM "XMLPO.dtd">

<PO>
   <Header>
   <FROM>RETAILER1</FROM>
   <TO>SUP1</TO>
   <PONO> 12345669 </PONO>
   <PODate> 20021018 </PODate>
   </Header>

   <Detail>
     <QTY> 1000 </QTY>
     <ITEMNO> ABC2 </ITEMNO>
     <DESC> Some product </DESC>
   </Detail>
</PO>
```

To import this DTD in WebSphere Data Interchange, you need to have a dictionary to hold the DTD. If you do not have already a dictionary, or if you want to create a new one to store this DTD, open the XML window by clicking the XML button in the main tool bar of WebSphere Data Interchange. A new window will appear that lists XML dictionaries and XML DTDs. Select the tab XML Dictionary and select **File -> New** to create a new dictionary. Name the new dictionary 850XML, for example. Save the new dictionary by selecting **File -> Save**. Now you can import the DTD itself. Select **File -> Open Import File** to start this import process. In the file browser window, change the file type property from Export/Import Files (*.eif) to XML DTD File (*.dtd). During the import you will be prompted to provide the following information:

- DTD file name
- DTD object name in WebSphere Data Interchange, for example POXML
- Name of the dictionary, to be selected from a drop-down box
- Name of the target database, to be selected from a list
- Name of the root element: set this to PO for the DTD listed in Example 2-15 on page 68.

When the import has completed, open the DTD document within WebSphere Data Interchange (see Figure 2-6). Within this document, you can name the XML elements that identify sender and receiver. If the names in the XML document do not match directly with the EDI names, you can provide the name of a translation table that WebSphere Data Interchange can use at runtime to look-up the correct partner information. Note that this information is used when building an EDI



DTD Name	XMLPO
Dictionary Name	850XML
Description	
Root Element Name	PO

Sender

Qualifier Element	
ID Element	\PO\Header\FROM\\
Translation Table	

Receiver

Qualifier Element	
ID Element	\PO\Header\TO\\
Translation Table	

Figure 2-6 Finding partner information in the XML document

The DTD that was provided in Example 2-15 on page 68 has only one field to provide information about the sender and another field to provide information about the receiver. To make the link, set:

- Field ID Element for Sender: `\PO\Header\FROM\\`
- Field ID Element for Receiver: `\PO\Header\TO\\`

Note the double back-slash symbol at the end.

The result is now that we can build custom rules for routing and mapping based on who is the target or source partner for a given document. For an incoming EDI 850 document, the information in the ISA segment will be copied into the XML elements that were set in Figure 2-6 and we will not require specific mapping statements in the translation map.

Definition of the translation map

Since the Supplier company expects to receive EDI 850 documents (purchase orders) from its customers, the retailers, we need a map that translates the incoming EDI document into an XML document that can be handled by the Supplier's internal systems. To create this map, start the map editor by clicking the mapping button in the main tool bar of WebSphere Data Interchange. Within the map editor, click either the new document button or select **File -> New**. A map definition wizard will appear. Provide the following values for these parameters:

- Map name: 850TOXML
- Target or source based: Target
- Source document definition: EDI Standard
- Source dictionary: X12V4R3
- Source EDI transaction: 850
- Target syntax type: XML
- Target dictionary: 850XML
- Target DTD: POXML

Given the simple structure of the XML document, the mapping itself is quite easy too. Note that the target XML document is likely too simple for most practical purposes. We do not intend to cover all options for mapping EDI documents in this redbook. Figure 2-7 shows the mapping statements between the EDI segments and fields and the target XML document. These statements are obtained by dragging the EDI field onto the XML element.

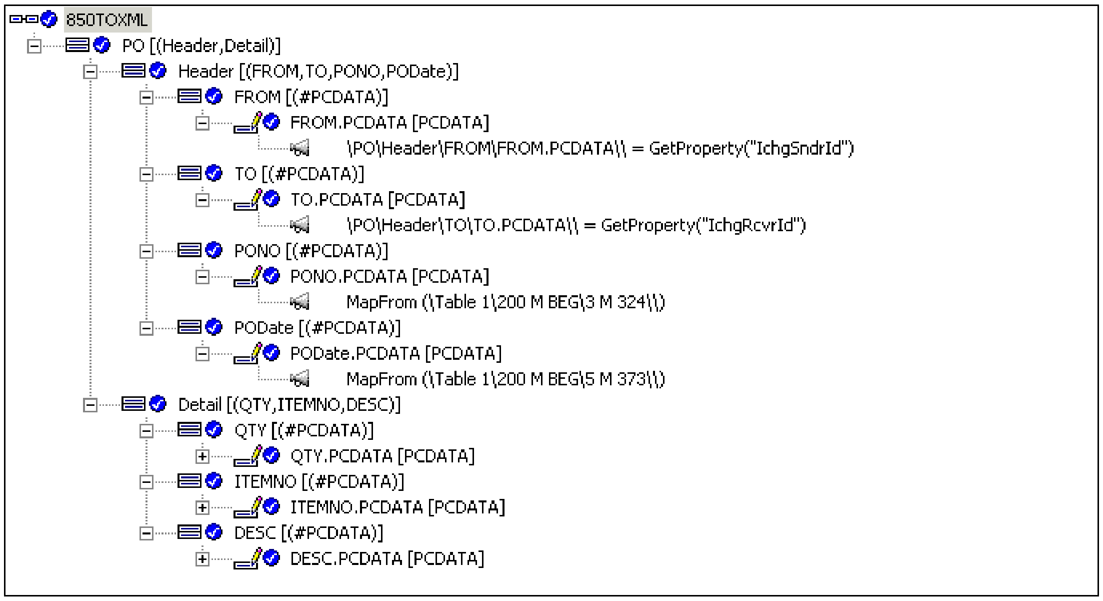

```
850TOXML
  PO [(Header,Detail)]
    Header [(FROM,TO,PONO,PODate)]
      FROM [(#PCDATA)]
        FROM.PCDATA [PCDATA]
            \PO\Header\FROM\FROM.PCDATA\\ = GetProperty("IchgSndrId")
      TO [(#PCDATA)]
        TO.PCDATA [PCDATA]
            \PO\Header\TO\TO.PCDATA\\ = GetProperty("IchgRcvrId")
      PONO [(#PCDATA)]
        PONO.PCDATA [PCDATA]
            MapFrom (\Table 1\200 M BEG\3 M 324\\)
      PODate [(#PCDATA)]
        PODate.PCDATA [PCDATA]
            MapFrom (\Table 1\200 M BEG\5 M 373\\)
    Detail [(QTY,ITEMNO,DESC)]
      QTY [(#PCDATA)]
        QTY.PCDATA [PCDATA]
      ITEMNO [(#PCDATA)]
        ITEMNO.PCDATA [PCDATA]
      DESC [(#PCDATA)]
        DESC.PCDATA [PCDATA]
```

Figure 2-7 Mapping the EDI 850 document into an XML document.

The two statements in Figure 2-7 that were not created by drag-and-drop, are the statements to fill in the elements FROM and TO in the Header element. To pass along the information of the EDI ISA header into the XML document, you can add an assignment command and call the function GetProperty to obtain the value of a field in the ISA header. Adding an assignment is performed by right-clicking the target field and selecting **Insert After -> Command -> Assignment** from the context menu. This will open a command editor to assist you in building the command.

Setup within WebSphere Data Interchange for SUP1 and RETAILER1

Now that we have a map, we need to tie it to a couple of other objects in WebSphere Data Interchange, by creating a rule for the map. However, before we can do that, we need to create those other objects in WebSphere Data Interchange.

The queue inbox will receive EDI documents from the retailers 1 and 2. The documents will be written in this queue by iSoft's P2PAgent. To make this queue known to WebSphere Data Interchange, we need to define a queue profile. Open the setup window in WebSphere Data

Interchange and select the tab MQSeries Queues. Select File -> New to create a new document. You need to name this document, for example INBOX. Set the name of the queue to inbox and specify the name of the queue manager, cw_studenta.queue.manager in our example. If your documents are large, you should consider setting an appropriate value for the field Maximum Message Length. The default value is 32KB, which might not be sufficient for your environment. Select File -> Save to store this new document in the database.

When WebSphere Data Interchange has translated the incoming EDI 850 document, it will need to send it to the internal system for processing. That destination might be a queue or a file in a given directory. We will describe both examples below. The target destination, either queue of directory, can be dependent of the document and/or dependent of the trading partner. For most environments, the back-end system will not require specific locations for each trading partner. But it is quite common that purchase orders are to be stored in a different location than for example requests for quotation.

In our example we're going to assume that all purchase orders are to be sent to the same queue, called purchase.orders, except for purchase orders coming from retailer3 for which we want to store the documents in a directory. As described before, create a queue profile PO_IN in WebSphere Data Interchange and set the queue name to purchase.orders and set the name of the queue manager, cw_studenta.queue.manager in our example. Save the document.

The next object we should define, is a network profile. While it is possible to use a single network profile to describe the access to the queues inbox and purchase.orders, it might be easier to separate those two. The queue inbox might contain messages with an RFH2 header, while the messages for the queue purchase.orders should not have this header. To create network profiles, open the setup window in WebSphere Data Interchange and select the tab Network Profiles. Select File -> New to create a new document.

Create a network profile INBOX, with the following values:

- Network ID: INBOX
- Communication Routine: VANIMQ
- Network Program: EDIRFH2 (iSoft's P2PAgent adds an RFH2 header)
- Network Parameters: RECEIVEMQ=INBOX (the name of the queue profile)

Save the document by selecting File -> Save and create a second network profile, with the following values:

- Network ID: `PO_IN`
- Communication Routine: `VANIMQ`
- Network Program: `EDIMQSR`
- Network Parameters: `SENDMQ=PO_IN` (the name of the queue profile)

The next step is to create mailboxes. A mailbox in WebSphere Data Interchange is a logical start point or end point for a translation. It can map onto a mailbox in a VAN solution or to any other resource, such as a queue or a file.

- Create mailbox INBOX with network ID INBOX.
- Create mailbox PO_IN with network ID PO_IN.

Finally, the setup within WebSphere Data Interchange is complete by creating services profiles. A service profile is named after a mailbox and contains the WebSphere Data Interchange commands that need to be executed when a document arrives in a mailbox.

Create service profile INBOX with the following settings:

- Perform command:
```
PERFORM TRANSFORM WHERE INFILE(INBOX) SYNTAX(E) OUTFILE(PO_IN)
```

- Output files: PO_IN ..\xml\po_in.txt

Create a service profile PO_IN with the following settings:

- Perform command:
  `PERFORM SEND WHERE REQID(PO_IN) OUTFILE(PO_IN) OUTTYPE(MQ) CLEARFILE(Y)`
- Input files: PO_IN ..\xml\po_in.txt

Creating a rule for the translation map

So far, we have created a transformation map and we have created objects within WebSphere Data Interchange that control the flow of data. The last object to create is a rule (or usage) associated with a map. The rule will tell WebSphere Data Interchange when a map should be used.

Open the mapping window and select the map 850TOXML that we have created before. Select Actions -> Usages. Since a rule is associated with a map, some conditions are already set up front. It's clear that the map 850TOXML will only be used to transform EDI 850 documents that adhere to the standard X12V4R3. But typically, you will need more granularity. Often a map should only be used when the document comes from one partner or when the document is going to be sent to one partner, while another map should be used for yet another partner. That is the kind of information that you can encode in a data transformation map. Select File -> New to create a new rule for the map 850TOXML. Set the following attributes:

- Usage Indicator: `Production`
- Trading Partners:
  - Sending: `ANY`
  - Receiving: `SUP1`
- Make the map active by selecting the check box **Active**.

This results in a map and rule where documents received from any partner and targeted for the internal trading partner SUP1 are all translated according to the same rules.

Updating WebSphere MQ resources

The translation process that we have described in the previous section assumes that messages are going to arrive in a queue called inbox and that they can be written after translation in another queue, called purchase.orders. The queue inbox was already created before when we had configured iSoft's P2PAgent. Define now the queue purchase.orders using WebSphere MQ Explorer.

To automate the translation process, we would like to use the triggering features of WebSphere MQ. Therefore, update the definition of the queue inbox and set the following attributes using WebSphere MQ Explorer:

On the tab labeled Triggering:

- Trigger Control: `On`
- Trigger Type: `First`
- Initiation Queue: `WDI.INIT.Q`
- Process Name: `WDI.PROC`

Usually, the setup of a WebSphere Data Interchange server includes the creation of a number of WebSphere MQ objects. The commands to create these objects, are available in the file wdimqcommands.txt in the samples directory of a WebSphere Data Interchange server installation. If you have executed these commands, then the queue WDI.INIT.Q already exists and also the process WDI.PROC already exists. If the process WDI.PROC and the queue

WDI.INIT.Q do not exist for your queue manager, consult the file wdimqcommands.txt to know the requirements for these objects.

Completing the setup of WebSphere Data Interchange

The above configuration of WebSphere Data Interchange works fine for input received from Retailer1. The EDI 850 document from Retailer1 is written in the queue inbox by iSoft's P2PAgent and WebSphere Data Interchange will pick it up and translate it based on the rule that we created before. What needs to be done when the Retailer1 sends other types of EDI documents? For sure, we need to make sure that the EDI definition for this document is imported and that an equivalent DTD is imported. Also, we will need to define a map between the EDI definition and the XML document. This new EDI document will arrive in the same queue called inbox. WebSphere Data Interchange is able to detect that this is a different document than the EDI 850 and hence can pick up the correct map. But, will this new document after translation be passed onto the same queue purchase.orders? Likely not. How is WebSphere Data Interchange going to perform this routing?

At this point, any message in the queue inbox is going to be handed over to the queue purchase.orders. To tie the routing to the document type, and not the origin, perform the following changes to the configuration that you have created so far:

- Update the rule for 850TOXML. Set output file to PO_IN and type to MQ on the tab General of the data transformation rule.
- Update the service profile INBOX and remove the references to the output file on the perform command for the service profile INBOX. The updated perform command looks now as follows:

  ```
  PERFORM TRANSFORM WHERE INFILE(INBOX) SYNTAX(E)
  ```

At this time, the target destination (PO_IN) is set in the rule and can be tied to either trading partner combination and/or document combination.

For the rule associated with the map for the new EDI document, set the output file to a different file. Add a queue profile, a service profile, a mailbox profile, and a network profile to the WebSphere Data Interchange configuration.

When adding more trading partners (Retailer2, Retailer3 and so on), we need to make sure that these trading partners are known in WebSphere Data Interchange. The rule itself was not linked to a source trading partner. However, in the setup of iSoft's P2PAgent, we had created a configuration where there was a one-to-one relationship between a trading partner and an internal ID (SUP1, SUP2, and SUP3). If you are sure that the rule for the translation of 850 documents is partner independent, you can update the rule to make it an any-to-any rule. Alternatively, you need to create an additional rule, for example to set a specific output file. Or, you may need to create an additional map and rule, to handle specific translation and/or destination requirements.

Processing file-based input for WebSphere Data Interchange

When iSoft's P2PAgent receives an EDI document and it is configured to store the document as a file in a given directory, it will generate a unique file name for each incoming document. The default generated name contains sender and receiver identification, followed by the date and time and a message id. An example of such a name is shown below:

```
RETAILER3.SUP3.20030127192149C8A1302A.file.in
```

The file name can be controlled by using the iSoft command **`set -fs`**. Then, the file name will not include partner IDs.

Besides variable file names, there is also the issue of how to start the EDI translation engine. When using WebSphere MQ, you can rely on MQ triggering to start the EDI translation engine. The most common solution is to use a scheduler tool and run a command file at a regular interval to process incoming EDI documents.

Both issues, variable file names and kicking off the translation process, can be solved in a variety of ways using command files and scripts. We describe here one solution that will mainly focus on the interaction between WebSphere Data Interchange and iSoft's P2PAgent and not on the scripting aspect.

Assuming that we have a configuration of iSoft as described in Example 2-14 on page 64, the EDI documents will be written by the P2PAgent program in a directory inbox\retailer3. Assume that the installation directory of WebSphere Data Interchange Server is C:\WDIServer32. In the directory C:\WDIServer32\runtime\dicmd, we created a command file, called wdi.bat with the following commands

Example 2-17 Command file wdi.bat

```
echo off
For %%f in (c:\isoft_enterprise\inbox\retailer3\*.file.in) do @translate.bat %%f
```

This command file will result in running another command file, translate.bat, as many times as there are EDI document files in the directory inbox\retailer3. We need to make sure that we don't process an EDI document twice or never. The set of file names will be built at the beginning of the execution of the command file wdi.bat. The name of each file will be passed as a parameter to translate.bat. It is clear that the command file translate.bat will have to move or rename the file when the processing is complete, otherwise it will be processed again the next time that the wdi.bat command file is executed. If a new document arrives during the period of time that wdi.bat is already running, it will not be found before the next time that wdi.bat is scheduled to run.

The contents of translate.bat is shown in Example 2-18. Since the command file knows the variable name of the file for which it is started, we can copy it to a fixed name, in this case edi.in. Then we make sure that the bin directory of WebSphere Data Interchange is part of the PATH environment variable. Next, we call the EDISERVR program which is the WebSphere Data Interchange engine. That program is given some WebSphere Data Interchange commands via indirection. Finally, we copy the original source file to add the extension .processed to its name and delete the original file.

Since all files will be copied at some point to the file edi.in, we can not run multiple instances of translate.bat at the same time. As a result we can not run multiple instances of wdi.bat at the same time. If this would cause a problem for your environment, you would need to write smarter scripts to handle that.

Example 2-18 Command file translate.bat

```
echo %1
copy %1 c:\isoft_enterprise\inbox\retailer3\edi.in
set WDIRESTOREPATH=%PATH%
set PATH=C:\WDISERVER32\bin;%PATH%
ediservr < wdicmds.txt
set PATH=%WDIRESTOREPATH%
copy %1 %1.processed
del %1
```

Finally, let us look at the contents of the file wdicmds.txt, the content of which is given next.

Example 2-19 Contents of the file wdicmds.txt

```
set plan(WDIC);
init;
set file(PRTFILE,prtfile.txt);
set file(TRKFILE,trkfile.txt);
set file(EXPFILE, expfile.txt);
set file(EDI_IN,c:\isoft_enterprise\inbox\retailer3\edi.in);
set file(PO_IN,c:\isoft_enterprise\inbox\retailer3\po.in);
PERFORM TRANSFORM WHERE INFILE(EDI_IN) SYNTAX(E);
term;
```

You can easily see the correspondence between these commands and what we have configured before using the client interface of WebSphere Data Interchange. The set file commands correspond to the service profile settings where we had given values for similar parameters. Setting the plan to the name of the database was something we did before in the file wdi.properties. The PERFORM command in Example 2-19is the same command as we had in the service profile INBOX.

It should be noted that the translated XML document is always stored in a file called po.in. By default, WebSphere Data Interchange will append to this file, if it already exists. A single file with multiple XML documents might cause problems for other applications that are going to process this incoming order. If that is the case, an easy solution might be to add a copy command to translate.bat. For example:

```
copy c:\isoft_enterprise\inbox\retailer3\po.in %1.translated
```

2.3.2 Preparing EDI documents

The process of translating XML documents to EDI documents is conceptually not much different from the reverse process that we explained in the previous section. Figure 2-8 on page 76 shows this flow in a graphical way. It shows that WebSphere Data Interchange can handle file-based and queue-based XML input at the same time. Figure 2-8 on page 76 shows also that documents for Retailer2 and Retailer1 are in a different queue. But they might as well be stored in the same queue, called POACKQ. The translated documents should be written in unique queues per trading partners, since the P2PAgent program can not yet handle routing information in the MQRFH2 at this time. This is likely to change in future versions of the P2PAgent program. Given the configuration of the P2PAgent program that was shown in Example 2-14 on page 64, the names of these queues are outbox.retailer1 and outbox.retailer2.

In this section, we will use EDI 855 (purchase order acknowledgement) as the document that is being sent by the supplier to the retailer that was sent previously a purchase order.

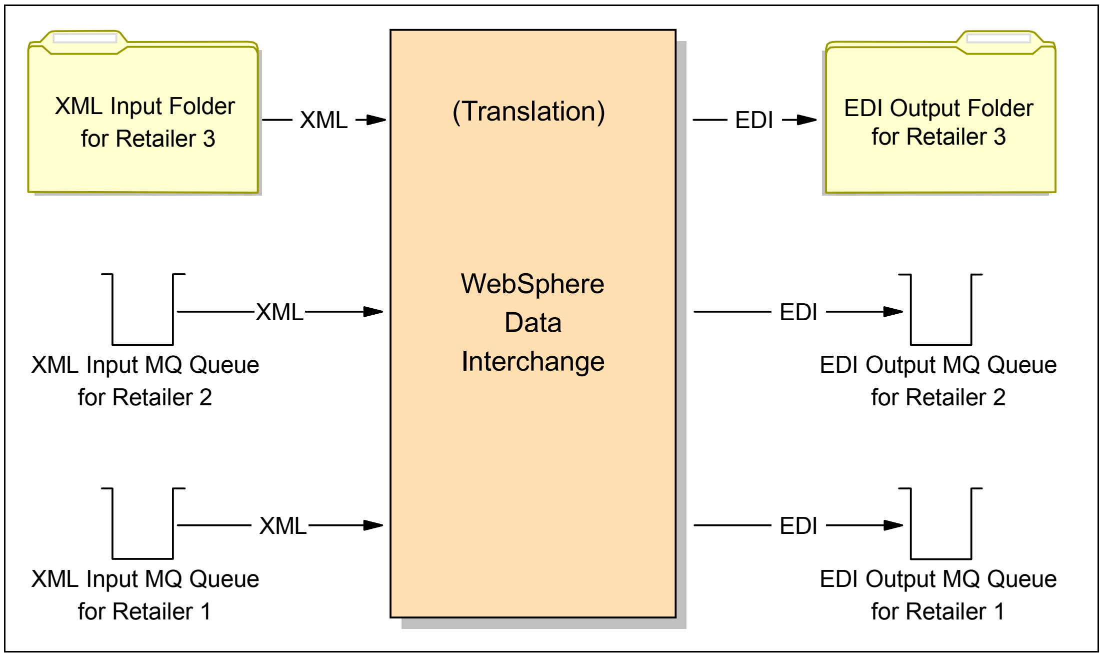

XML Input Folder for Retailer 3
XML
(Translation)
EDI
EDI Output Folder for Retailer 3
XML
WebSphere Data Interchange
EDI
XML Input MQ Queue for Retailer 2
EDI Output MQ Queue for Retailer 2
XML
EDI
XML Input MQ Queue for Retailer 1
EDI Output MQ Queue for Retailer 1

Figure 2-8 Outbound data flow

The process of configuring WebSphere Data Interchange for this task consists of the following steps:

- Definition of trading partner profiles for all retailers and the supplier itself. This step had been completed already when we described the setup for the inbound flow.
- Document definition
  - Import of the EDI 855 document definition. During the import of the 850 document, we had also selected the 855 document. Thus, we can skip this step.
  - Import of a DTD, matching the internal representation of a purchase order.
- Definition of the translation map
- Definition of mailboxes, network profiles, queue profiles and service profiles
- Definition of the rules associated with the map
- Creating a command file to process incoming XML files
- Definitions of queue and process objects in WebSphere MQ to support the automatic translation of incoming XML documents in queues.

XML document definition

In 2.3.1, “Translating received EDI documents” on page 66, we described how the company Supplier was processing incoming EDI 850 documents. In general, when a company receives such a document, it will respond with a purchase acknowledgement, which is an 855 document in EDI terminology. The internal systems of the company Supplier will likely not generate an 855 document directly. The format will be company and application specific. Example 2-20 on page 77 shows a simple DTD representing an XML document that contains information that is typically found in a PO Ack. Again, the sample DTD is simple to not loose focus in this redbook.

Example 2-20 DTD representing a PO acknowledgement

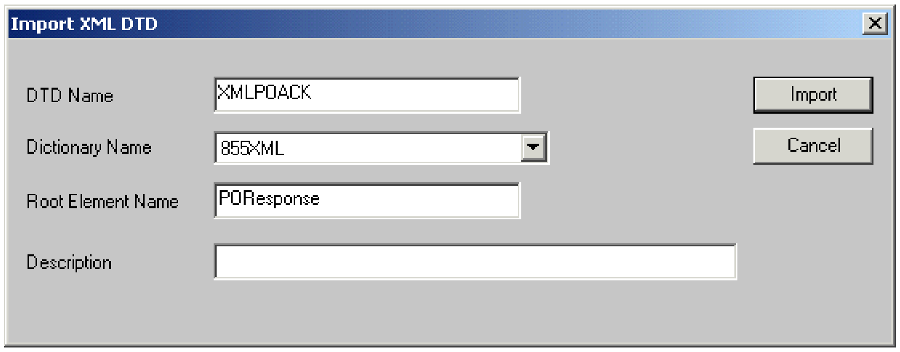
```
<?xml encoding="US-ASCII"?>
<!ELEMENT POResponse (Header,Detail)>

<!ELEMENT Header (PONumber,TargetPartnerID,Response)>

<!ELEMENT PONumber (#PCDATA)>
<!ELEMENT TargetPartnerID (#PCDATA)>
<!ELEMENT Response (#PCDATA)>

<!ELEMENT Detail (ItemNumber,Quantity,Description)>

<!ELEMENT ItemNumber (#PCDATA)>
<!ELEMENT Quantity (#PCDATA)>
<!ELEMENT Description (#PCDATA)>
```

Example 2-21 shows a sample message that complies with the DTD of Example 2-20. It contains an ID representing the target partner, a PO ID and a response field. Further down, we also see a detailed view of the actual order.

Example 2-21 Sample XML document representing a PO acknowledgement

```
<?xml version="1.0"?>
<POResponse>
   <Header>
      <PONumber>P12347</PONumber>
      <TargetPartnerID>RETAILER2</TradingPartnerID>
      <Response>AT</Response>
   </Header>
   <Detail>
      <ItemNumber>00123</ItemNumber>
      <Quantity>10</Quantity>
      <Description>Parts</Description>
   </Detail>
</POResponse>
```

The DTD is imported in a dictionary called 855XML and the root element name is set to POResponse (see Figure 2-9).



DTD Name: XMLPOACK
Dictionary Name: 855XML
Root Element Name: POResponse
Description:

Figure 2-9 Import XML DTD in WebSphere Data Interchange

After importing the DTD, we need to tell WebSphere Data Interchange about the role of the field TargetPartnerID. Open the DTD again in WebSphere Data Interchange and set the field ID Element for Receiver to \POResponse\Header\TargetTradingPartnerID\\. Refer to Figure 2-6 on page 69 to understand where we did this for the PO DTD.

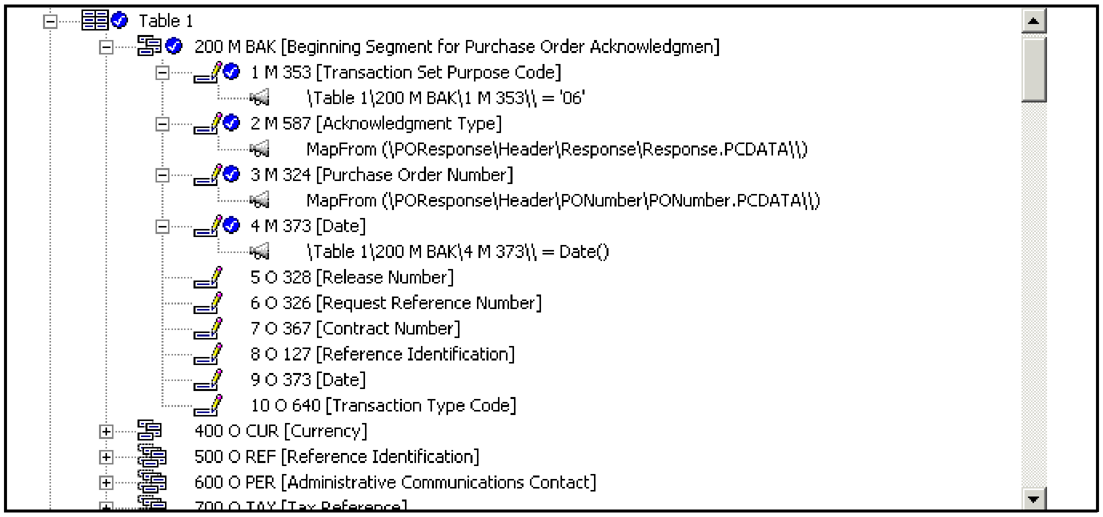
Definition of the translation map

The next step is to create the translation map to translate the XML document in the corresponding EDI 855 document. Open the map editor and select **File -> New**. Use the following values in the map definition wizard:

- Map name: `XMLTO855`
- Target or source based: `Target`
- Source document definition: `XML`
- Source dictionary name: `855XML`
- Source DTD: `XMLPOACK`
- Target document definition: `EDI`
- Target dictionary name: `X12V4R3`
- Target EDI standard transaction: `855`

After reviewing your selections in the summary window of the wizard, select **Finish** to start the actual mapping between the XML document and the EDI transaction.

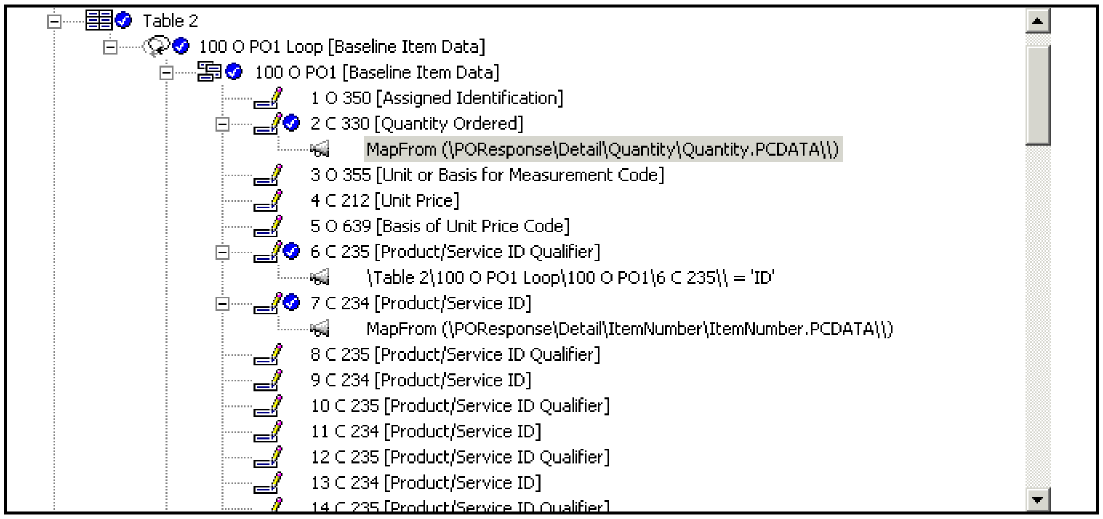
Figure 2-10 shows the first portion of the map. It shows the mappings for Table 1. The data segments of the BAK segment are filled in by dragging the corresponding elements from the XML document. The field Date is filled in via an assignment command in which we invoke the built-in function Date(). The field Transaction Set Purpose Code is filled in via an assignment in which the field is set to the constant value ‘06’.



```
Table 1
  200 M BAK [Beginning Segment for Purchase Order Acknowledgmen]
    1 M 353 [Transaction Set Purpose Code]
        \Table 1\200 M BAK\1 M 353\\ = '06'
    2 M 587 [Acknowledgment Type]
        MapFrom (\POResponse\Header\Response\Response.PCDATA\\)
    3 M 324 [Purchase Order Number]
        MapFrom (\POResponse\Header\PONumber\PONumber.PCDATA\\)
    4 M 373 [Date]
        \Table 1\200 M BAK\4 M 373\\ = Date()
    5 O 328 [Release Number]
    6 O 326 [Request Reference Number]
    7 O 367 [Contract Number]
    8 O 127 [Reference Identification]
    9 O 373 [Date]
    10 O 640 [Transaction Type Code]
  400 O CUR [Currency]
  500 O REF [Reference Identification]
  600 O PER [Administrative Communications Contact]
  700 O TAX [Tax Reference]
```

Figure 2-10 Building an EDI 855 document - step 1

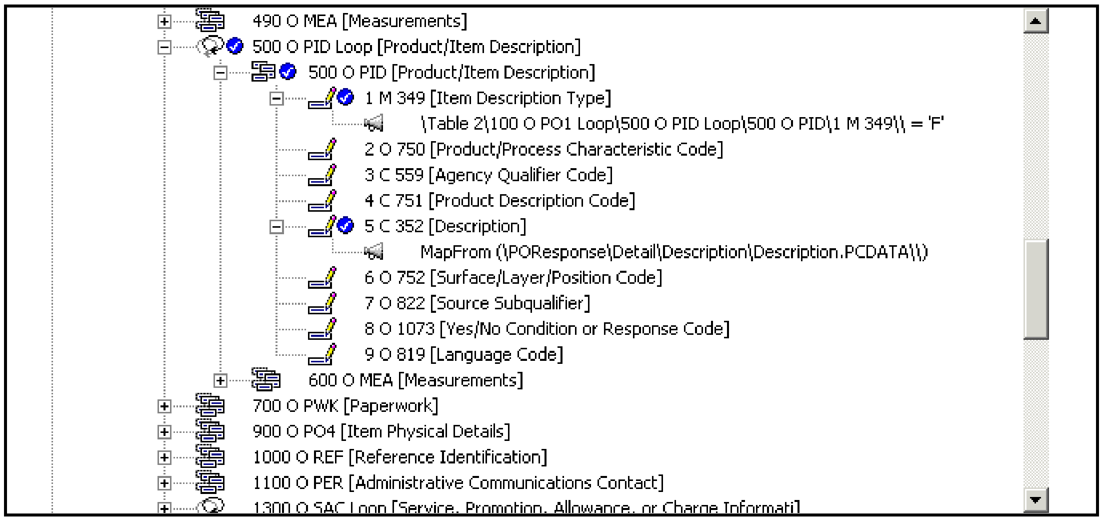
Figure 2-11 on page 79 shows the first part of the mapping for Table 2. Two data elements are mapped directly from the source XML document, while the element Product/Service ID Qualifier is filled in via an assignment.



```
Table 2
  100 O PO1 Loop [Baseline Item Data]
    100 O PO1 [Baseline Item Data]
      1 O 350 [Assigned Identification]
      2 C 330 [Quantity Ordered]
        MapFrom (\POResponse\Detail\Quantity\Quantity.PCDATA\\)
      3 O 355 [Unit or Basis for Measurement Code]
      4 C 212 [Unit Price]
      5 O 639 [Basis of Unit Price Code]
      6 C 235 [Product/Service ID Qualifier]
        \Table 2\100 O PO1 Loop\100 O PO1\6 C 235\\ = 'ID'
      7 C 234 [Product/Service ID]
        MapFrom (\POResponse\Detail\ItemNumber\ItemNumber.PCDATA\\)
      8 C 235 [Product/Service ID Qualifier]
      9 C 234 [Product/Service ID]
      10 C 235 [Product/Service ID Qualifier]
      11 C 234 [Product/Service ID]
      12 C 235 [Product/Service ID Qualifier]
      13 C 234 [Product/Service ID]
```

Figure 2-11 Building an EDI 855 document - step 2

Figure 2-12 shows the second part of the mapping for Table 2. One element is filled via a direct mapping from the corresponding element in the XML document, while the other element is filled in via assignment.



```
    490 O MEA [Measurements]
    500 O PID Loop [Product/Item Description]
      500 O PID [Product/Item Description]
        1 M 349 [Item Description Type]
          \Table 2\100 O PO1 Loop\500 O PID Loop\500 O PID\1 M 349\\ = 'F'
        2 O 750 [Product/Process Characteristic Code]
        3 C 559 [Agency Qualifier Code]
        4 C 751 [Product Description Code]
        5 C 352 [Description]
          MapFrom (\POResponse\Detail\Description\Description.PCDATA\\)
        6 O 752 [Surface/Layer/Position Code]
        7 O 822 [Source Subqualifier]
        8 O 1073 [Yes/No Condition or Response Code]
        9 O 819 [Language Code]
      600 O MEA [Measurements]
    700 O PWK [Paperwork]
    900 O PO4 [Item Physical Details]
    1000 O REF [Reference Identification]
    1100 O PER [Administrative Communications Contact]
```

Figure 2-12 Building an EDI 855 document - step 3

Setup within WebSphere Data Interchange

Similar to what we have done for the inbound flow, we need to set up a number of objects in WebSphere Data Interchange to support this map.

Open the setup window in WebSphere Data Interchange and define an MQSeries queue profile POACKQ for the queue POACKQ. Set also the name of the queue manager in this profile. Define a second MQSeries queue profile called OUT_RET1 for the queue outbox.retailer1. Define also an MQSeries queue profile OUT_RET2 for the queue outbox.retailer2.

Notice the implied naming convention for the queues. The queue POACKQ, and also the queue PO_IN that was used before, have a name that imply a function, since the contents of the queue will be application dependent. The queues inbox, outbox.retailer1 and outbox.retailer2 are not application dependent but destination dependent. The queue inbox was used for EDI 850 documents but can be used for any type of documents for which the destination is the internal system of the Supplier. The queues outbox.retailer1 and outbox.retailer2 are to be used for any type of document targeted for the implied trading partner.

The next step is the definition of three network profiles, which are defined in the setup window of WebSphere Data Interchange.

- Create network profile OUT_RET1 with values:
  - Network ID: `OUT_RET1`
  - Communication Routine: `VANIMQ`
  - Network Program: `EDIMQSR`
  - Network Parameters: `SENDMQ=OUT_RET1`
- Create network profile OUT_RET2 with values:
  - Network ID: `OUT_RET2`
  - Communication Routine: `VANIMQ`
  - Network Program: `EDIMQSR`
  - Network Parameters: `SENDMQ=OUT_RET2`
- Create network profile POACKQ with values:
  - Network ID: `POACKQ`
  - Communication Routine: `VANIMQ`
  - Network Program: `EDIMQSR`
  - Network Parameters: `RECEIVEMQ=POACKQ`

Next, we need some mailboxes to represent the new destination and source queues. Create the following mailboxes:

- OUT_RET1: set network profile ID to `OUT_RET1`
- OUT_RET2: set network profile ID to `OUT_RET2`
- POACKQ: set network profile ID to `POACKQ`

Finally, we need to create service profiles that describe the actions that WebSphere Data Interchange should perform when documents are posted in a mailbox.

- Service profile OUT_RET1
  - Perform command:
    `PERFORM SEND WHERE REQID(OUT_RET1) CLEARFILE(Y)`
  - Input files: OUT_RET1 - ..\xml\out_ret1.txt
- Service profile OUT_RET2
  - Perform command:
    `PERFORM SEND WHERE REQID(OUT_RET1) CLEARFILE(Y)`
  - Input files: OUT_RET2 - ..\xml\out_ret2.txt
- Service profile POACKQ
  - Perform command:
    `PERFORM TRANSFORM WHERE INFILE(POACKQ) SYNTAX(X)`
  - Input files: OUT_RET1 - ..\xml\out_ret1.txt
  - Input files: OUT_RET2 - ..\xml\out_ret2.txt

Tip: Since these profiles are all similar, you can use the copy function of WebSphere Data Interchange by selecting the menu option **Action -> Copy**.

Creating an envelope profile

Before we can tie everything together, we need one more object in WebSphere Data Interchange, which is an envelope profile. EDI documents can be grouped in a single file and surrounded by an envelope. An envelope can contain documents of different transactions. In our setup, we always process (send and receive) the documents as soon as they become available. You may have situations where you want to batch the documents before enveloping them and then passing them to the trading partner. However, even when you will send and receive the documents one at a time, you still need to have an envelope.

Envelope profiles are used to set values for specific segments in an EDI document, such as the ISA segment. The ISA segment contains, beside others, the fields sender and receiver ID and the fields sender and receiver qualifier. In the DTD definition, we have made the link between the XML element TargetPartnerID and receiver ID. However, no information was available in the XML document to fill in the field sender ID. We could have set a value in the map, using the setProperty built-in function. However, that would result in a value (say SUP1) that is independent of the target partner. Remember that the iSoft configuration had a separate ID for the supplier to interact with a retailer. An elegant solution to set the correct value (SUP1, SUP2 or SUP3) in the ISA segment is to use separate X envelope profiles.

Envelope profiles are managed in the setup window. Different types exist corresponding with different EDI standards. Since we have used X12 documents, we will need to create an X profile.

Select the tab **X Envelope Profiles** and select **File -> New**. Name the profile 855SUP1. Select the tab **Interchange Header (ISA)** and set the field ISA06 to `SUP1`. Save the document. Create similar profiles 855SUP2 and 855SUP3 where the ISA06 field is set to `SUP2` and `SUP3`, respectively.

Creating a rule for the map

Given that the destination of the translated document is different based on the target partner ID, we will need to define three new rules for the map XMLTO855. Open the map editor and select the map XMLTO855 that we created before. Select Action -> Usages to open the rules editor. Select **File -> New** to create a rule and set these fields to the following values:

- Usage Indicator: `Production`
- Trading Partners:
  - Sending: `ANY`
  - Receiving: `RETAILER1`
- Output file and type: `OUT_RET1` and `MQ`
- Make the map active by selecting the check box **Active**.
- Select the tab **Envelope Attributes**
  - Envelope Type: `X`
  - Envelope Profile Name: `855SUP1`

Create a second and third rule for Retailer2 and Retailer3 with matching values.

Updating WebSphere MQ resources

The queues outbox.retailer1 and outbox.retailer2 were defined previously to support the configuration of the P2PAgent program. In order to validate the WebSphere Data Interchange configuration discussed above, we need to define a queue called POACKQ. This queue could be modeled after the queue inbox that was defined to support the inbound data flow. The queue POACKQ should have the following values for these attributes:

On the tab labeled Triggering:

- Trigger Control: `On`
- Trigger Type: `First`
- Initiation Queue: `WDI.INIT.Q`
- Process Name: `WDI.PROC`

These values can be set by using WebSphere MQ Explorer. Note that it is assumed that the objects WDI.INIT.Q and WDI.PROC have been created as part of the standard configuration of WebSphere Data Interchange.

When you write an MQ message (such as the one shown in Example 2-21 on page 77) on the queue POACKQ, the WDIAdapter program should be launched by the WebSphere MQ Trigger Monitor program and the queue outbox.retailer2 should contain a message as shown in Example 2-22.

Example 2-22 Sample translated EDI document

```
00000000 ISA*  *          *  *          *  *SUP2           *  *RETAILER2      *060303*1506* *
*000000009* *P*:!
00000106 GS*PR*  *  *060303*1506*000000009* *004030!
00000149 ST*855*000000009!
00000166 BAK*06*AT*P12347*20030306!
00000192 PO1**10****ID*00123!
00000212 PID*F****Parts!
00000227 SE*5*000000009!
00000242 GE*1*000000009!
00000257 IEA*1*000000009!
00000273 .
```

Processing file-based input for WebSphere Data Interchange

As shown in Figure 2-8 on page 76, the outbound flow for Retailer3 is file based. WebSphere Data Interchange needs to read files containing XML documents from a given directory and write them in the directory that is used by the P2PAgent program. Similar to the inbound flow, we need to make sure that script files are written in such a way that each file is processed exactly once. The P2PAgent program will have a persistent send command to monitor a given directory. An example of such a command is shown below:

```
send http SUP3 RETAILER3 -fPoutbox -fSout -fE.pend -tE2003231000000 -s -e -r1 -n2
```

This command will instruct the P2PAgent program to monitor the directory outbox and send all files with the extension out to partner RETAILER3. Sent files are renamed with an extension .pend.

On the input side, we need to assume again that the application that generates the XML files, uses filenames that are unique so that a file is not overwritten by this application before WebSphere Data Interchange has translated the XML document into an EDI document.

And finally, there is again the issue of automating the translation process, since we can not rely on WebSphere MQ triggering to start the translation engine.

To handle all these issues, we will present some script files. The first script file is called by a scheduler program at regular intervals and looks for files in the directory C:\iSoft_Enterprise\outbox\retailer3 with an extension of .file.txt. For each found file, the command file translate_out.bat is being called, passing it the name of the XML file. such a command file is shown in Example 2-23.

Example 2-23 Command file wdi_out.bat

```
echo off
For %%f in (c:\isoft_enterprise\outbox\retailer3\*.txt) do @translate_out.bat %%f
```

The second command file, translate_out.bat, prepares the WebSphere Data Interchange environment by setting the PATH environment variable correctly. It also copies the current file (passed to the command file as the first argument) to the intermediate file xml.in and then calls the actual translation engine. When the engine returns, the current file is renamed to have an extension .out, which is what the P2PAgent program is looking for.

Example 2-24 Command file translate_out.bat

```
echo %1
copy %1 c:\isoft_enterprise\outbox\retailer3\xml_in
set WDIRESTOREPATH=%PATH%
set PATH=C:\WDISERVER32\bin;%PATH%
ediservr < wdi_out_cmds.txt
copy C:\iSoft_Enterprise\outbox\retailer3\edi_out %1.out
copy %1 %1.processed
del %1
set PATH=%WDIRESTOREPATH%
```

The actual WebSphere Data Interchange commands are stored in the file wdi_out.cmds, shown in Example 2-25. Similar to the inbound flow, the commands consist of a series of environment setup commands followed by the familiar PERFORM command.

Example 2-25 WebSphere Data Interchange commands

```
set plan(WDIC);
init;
set file(PRTFILE,prtfile.txt);
set file(TRKFILE,trkfile.txt);
set file(EXPFILE, expfile.txt);
set file(XML_IN,c:\isoft_enterprise\outbox\retailer3\xml_in);
set file(POACK,c:\isoft_enterprise\outbox\retailer3\edi_out);
PERFORM TRANSFORM WHERE INFILE(XML_IN) SYNTAX(X) OUTFILE(POACK);
term;
```

The generated files, with the extension .out, are now ready for transmission by the P2PAgent program and will be picked up by it at the next polling interval.

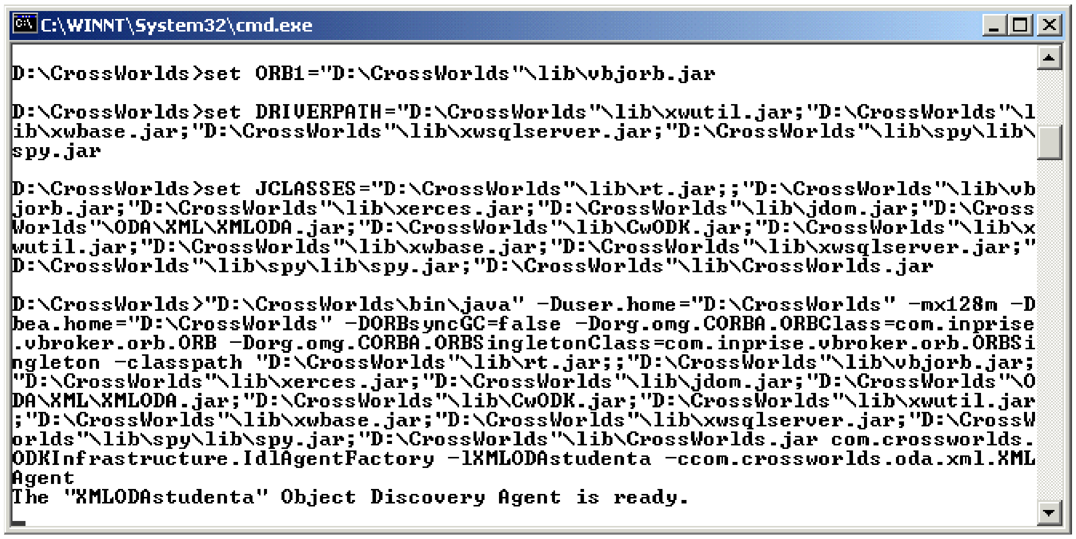
2.4 Integration with the Interchange Server

The InterChange Server (ICS) is often used as a platform for integrating applications within an enterprise. While we can not cover all aspects of using this technology in a single redbook, this section will describe some typical operations that would allow the ICS to interact with WebSphere Data Interchange. We will cover the use of the MQSeries Connector to send and receive data to and from products such as WebSphere Data Interchange. The use of other connectors, such as JTextConnector, is very similar.

2.4.1 Creating business objects

The first step would be the creation of a business object matching the DTD that we used before in WebSphere Data Interchange. You can use the Business Object Designer and define the fields manually. However, for a more realistic DTD representing a purchase order, there will be a lot more fields than what we used here. Defining the business object manually would then become an error-prone operation.

The Interchange Server provides some tools to make the definition of a business object easier. An optional installation component of the ICS is the XMLODA, XML Object Discovery Agent. Launch the agent, from the ODA\XML directory. When it is started, you should see a command windows as shown in Figure 2-13.



```
D:\CrossWorlds>set ORB1="D:\CrossWorlds"\lib\vbjorb.jar

D:\CrossWorlds>set DRIVERPATH="D:\CrossWorlds"\lib\xwutil.jar;"D:\CrossWorlds"\l
ib\xwbase.jar;"D:\CrossWorlds"\lib\xwsqlserver.jar;"D:\CrossWorlds"\lib\spy\lib\
spy.jar

D:\CrossWorlds>set JCLASSES="D:\CrossWorlds"\lib\rt.jar;;"D:\CrossWorlds"\lib\vb
jorb.jar;"D:\CrossWorlds"\lib\xerces.jar;"D:\CrossWorlds"\lib\jdom.jar;"D:\Cross
Worlds"\ODA\XML\XMLODA.jar;"D:\CrossWorlds"\lib\CwODK.jar;"D:\CrossWorlds"\lib\x
wutil.jar;"D:\CrossWorlds"\lib\xwbase.jar;"D:\CrossWorlds"\lib\xwsqlserver.jar;"
D:\CrossWorlds"\lib\spy\lib\spy.jar;"D:\CrossWorlds"\lib\CrossWorlds.jar

D:\CrossWorlds>"D:\CrossWorlds\bin\java" -Duser.home="D:\CrossWorlds" -mx128m -D
bea.home="D:\CrossWorlds" -DORBsyncGC=false -Dorg.omg.CORBA.ORBClass=com.inprise
.vbroker.orb.ORB -Dorg.omg.CORBA.ORBSingletonClass=com.inprise.vbroker.orb.ORBSi
ngleton -classpath "D:\CrossWorlds"\lib\rt.jar;;"D:\CrossWorlds"\lib\vbjorb.jar;
"D:\CrossWorlds"\lib\xerces.jar;"D:\CrossWorlds"\lib\jdom.jar;"D:\CrossWorlds"\O
DA\XML\XMLODA.jar;"D:\CrossWorlds"\lib\CwODK.jar;"D:\CrossWorlds"\lib\xwutil.jar
;"D:\CrossWorlds"\lib\xwbase.jar;"D:\CrossWorlds"\lib\xwsqlserver.jar;"D:\CrossW
orlds"\lib\spy\lib\spy.jar;"D:\CrossWorlds"\lib\CrossWorlds.jar com.crossworlds.
ODKInfrastructure.IdlAgentFactory -lXMLODAstudenta -ccom.crossworlds.oda.xml.XML
Agent
The "XMLODAstudenta" Object Discovery Agent is ready.
_
```

Figure 2-13 XML Object Discovery Agent is running

Launch now the Business Object Designer and select **File -> New Using ODA** from the menu, as shown in Figure 2-14 on page 85.

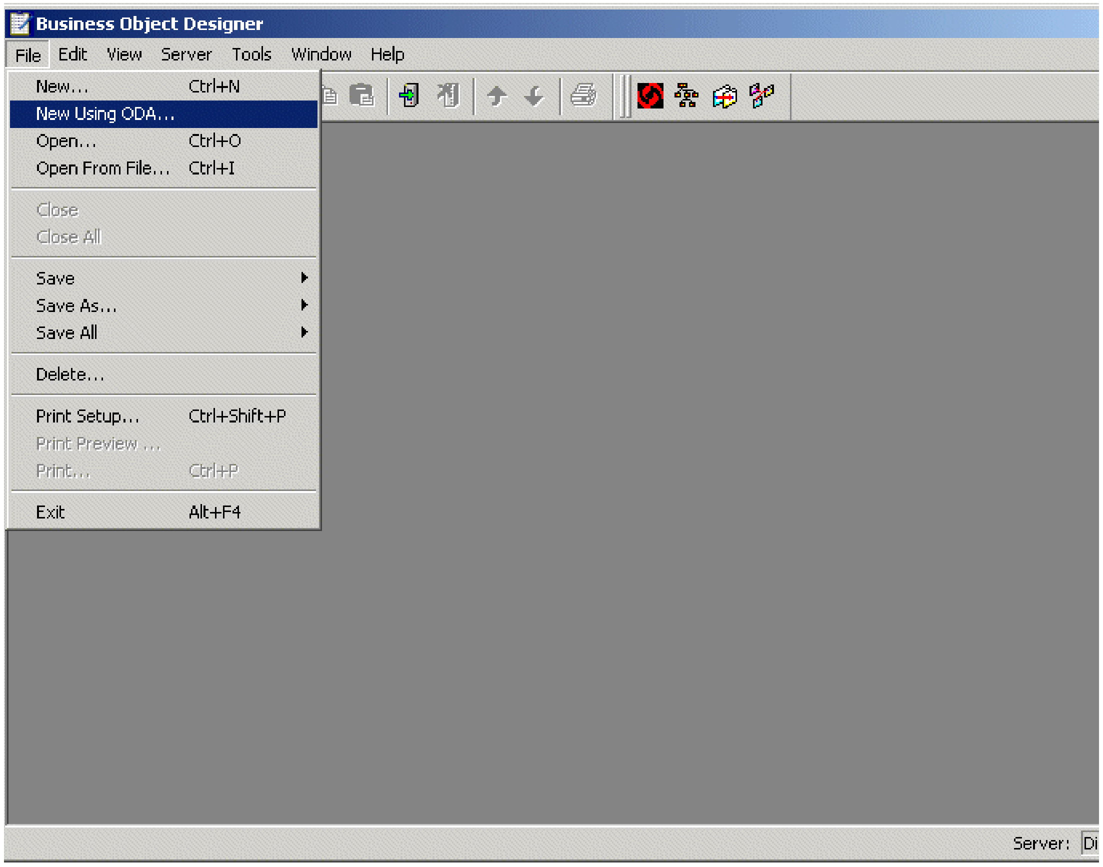
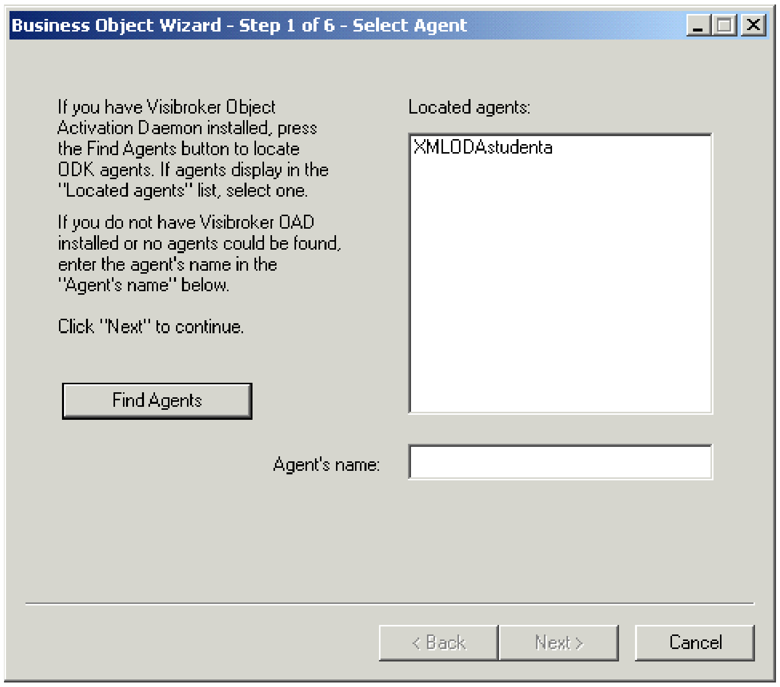


Figure 2-14 Using the Business Object Designer

A new window will appear that will guide you through the definition process. Click the button **Find Agents** to populate the right pane with available agents and select the XML ODA agent out of the list. Select **Next** to continue (Figure 2-15 on page 86).

Note: The Visibroker component should be running to get this list of available agents.


Business Object Wizard - Step 1 of 6 - Select Agent

If you have Visibroker Object Activation Daemon installed, press the Find Agents button to locate ODK agents. If agents display in the "Located agents" list, select one.

If you do not have Visibroker OAD installed or no agents could be found, enter the agent's name in the "Agent's name" below.

Click "Next" to continue.

Located agents: XMLODAstudenta

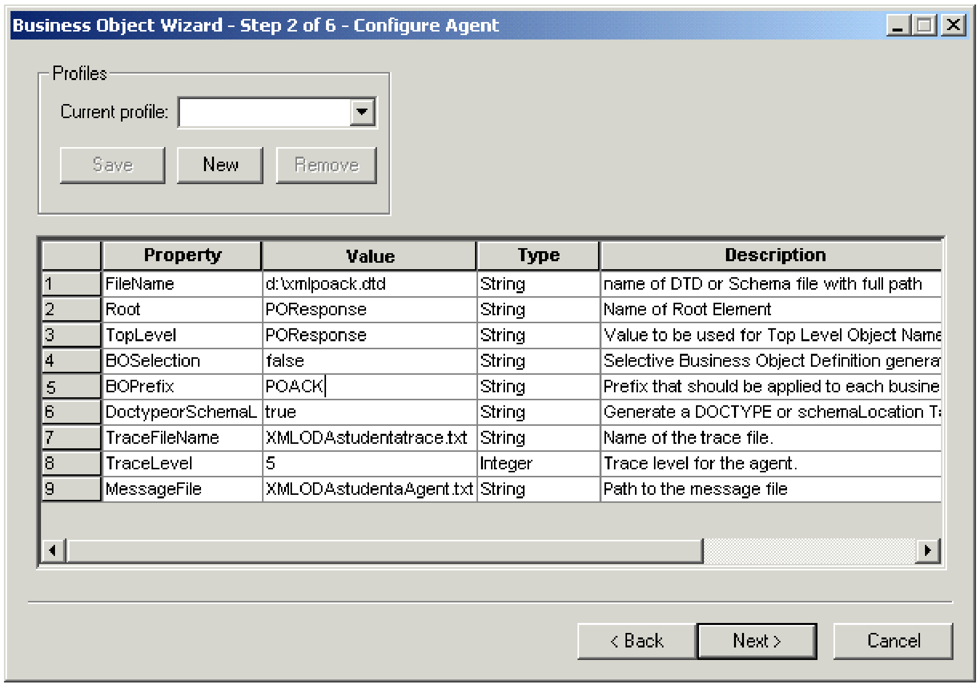
Agent's name:

Figure 2-15 Business Object wizard - Step 1

Most of the fields in step 2 are populated by default. Provide the following information:

- Name of the file that contains the DTD
- Root element
- Top Level element
- BOPrefix

and select **Next** to continue (Figure 2-16).


Business Object Wizard - Step 2 of 6 - Configure Agent

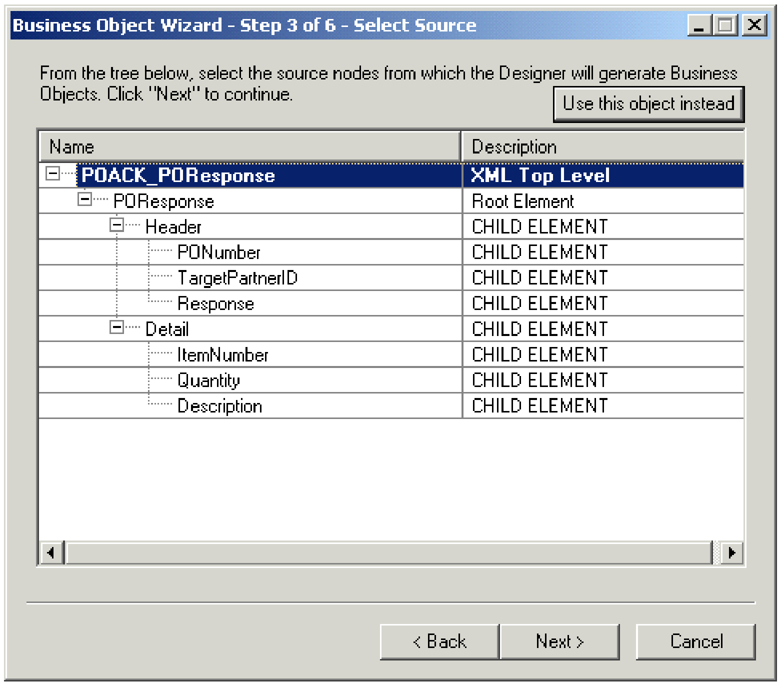
Profiles

Current profile:

	Property	Value	Type	Description
1	FileName	d:\xmlpoack.dtd	String	name of DTD or Schema file with full path
2	Root	POResponse	String	Name of Root Element
3	TopLevel	POResponse	String	Value to be used for Top Level Object Name
4	BOSelection	false	String	Selective Business Object Definition genera
5	BOPrefix	POACK	String	Prefix that should be applied to each busine
6	DoctypeorSchemaL	true	String	Generate a DOCTYPE or schemaLocation T
7	TraceFileName	XMLODAstudentatrace.txt	String	Name of the trace file.
8	TraceLevel	5	Integer	Trace level for the agent.
9	MessageFile	XMLODAstudentaAgent.txt	String	Path to the message file

Figure 2-16 Business Object wizard - Step 2

The next step allows you to select other levels (or nodes) in the XML document for which you would like to create a business object definition. You might for example require an object to represent a single Detail element. For our purposes, this is not required. So, we select the top one and click **Next** to continue (Figure 2-17).

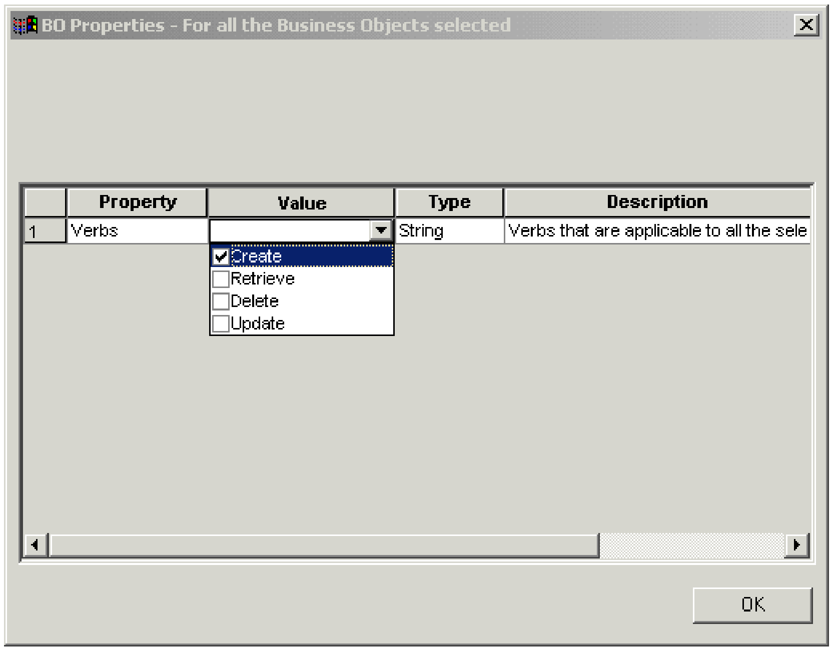

Business Object Wizard - Step 3 of 6 - Select Source

From the tree below, select the source nodes from which the Designer will generate Business Objects. Click "Next" to continue.

Name	Description
POACK_POResponse	XML Top Level
POResponse	Root Element
Header	CHILD ELEMENT
PONumber	CHILD ELEMENT
TargetPartnerID	CHILD ELEMENT
Response	CHILD ELEMENT
Detail	CHILD ELEMENT
ItemNumber	CHILD ELEMENT
Quantity	CHILD ELEMENT
Description	CHILD ELEMENT

Figure 2-17 Business Object wizard - Step 3

Step 4 summarizes your selections so far. At step 5, you need to select a verb with the business objects. Figure 2-18 shows the selection of the Create verb. Select **OK** to continue.

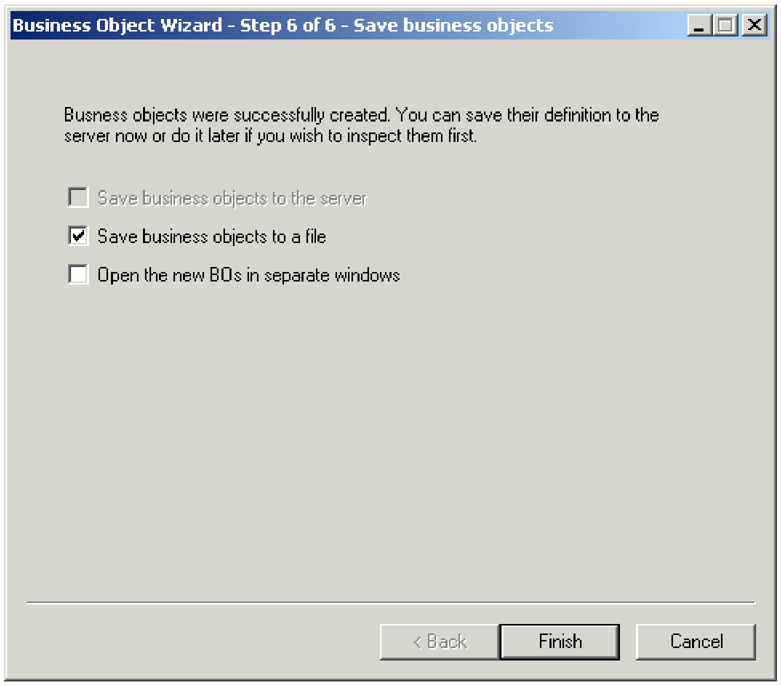

BO Properties - For all the Business Objects selected

	Property	Value	Type	Description
1	Verbs		String	Verbs that are applicable to all the sele

[✓] Create
[ ] Retrieve
[ ] Delete
[ ] Update

Figure 2-18 Business Object wizard - Step 5

Finally, in step 6, you can select where to save the business object. If the ICS is running and you are connected to it, the first option, save to server, should be available. Alternatively, save the business object to an import file (selected option in Figure 2-19) and open the file later in the System Manager via **File -> Open from File**.


Business Object Wizard - Step 6 of 6 - Save business objects

Busness objects were successfully created. You can save their definition to the server now or do it later if you wish to inspect them first.

- [ ] Save business objects to the server
- [✓] Save business objects to a file
- [ ] Open the new BOs in separate windows

Figure 2-19 Business Object wizard - Step 6

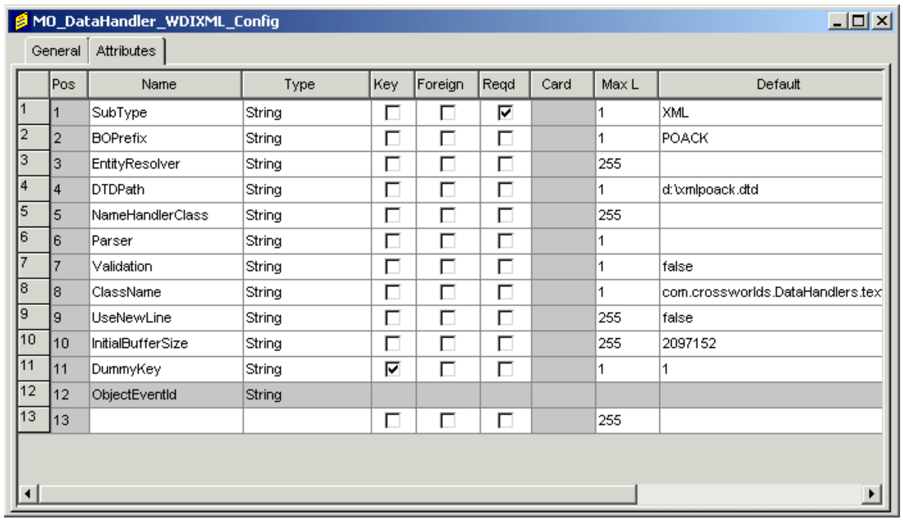
2.4.2 Configuring the MQSeries connector

The configuration of the MQSeries connector consists of the definition of some meta-data business objects and the configuration of the connector itself.

Updating the XML meta-object

Open the business object MO_DataHandler_DefaultXMLConfig and save it as MO_DataHandler_WDIXML_Config. Make the following changes to this business object:

- Set the attribute DTDPath to the directory that holds the DTD for the POResponse XML document.
- Set the BOPrefix to POACK, which is the prefix used during the creation of the business object (see Figure 2-16 on page 86).
- Save this business object to the server.



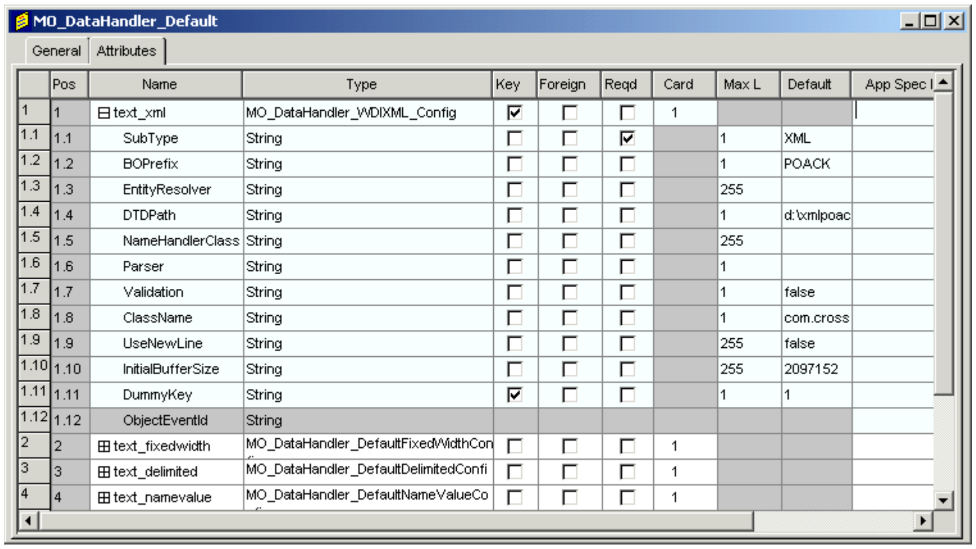
MO_DataHandler_WDIXML_Config

General | Attributes

	Pos	Name	Type	Key	Foreign	Reqd	Card	Max L	Default
1	1	SubType	String	☐	☐	☑		1	XML
2	2	BOPrefix	String	☐	☐	☐		1	POACK
3	3	EntityResolver	String	☐	☐	☐		255	
4	4	DTDPath	String	☐	☐	☐		1	d:\xmlpoack.dtd
5	5	NameHandlerClass	String	☐	☐	☐		255	
6	6	Parser	String	☐	☐	☐		1	
7	7	Validation	String	☐	☐	☐		1	false
8	8	ClassName	String	☐	☐	☐		1	com.crossworlds.DataHandlers.tex
9	9	UseNewLine	String	☐	☐	☐		255	false
10	10	InitialBufferSize	String	☐	☐	☐		255	2097152
11	11	DummyKey	String	☑	☐	☐		1	1
12	12	ObjectEventId	String						
13	13			☐	☐	☐		255	

Figure 2-20 Data Handler business object

Open now the business object MO_DataHandler_Default. Update the type field for element text_xml and set it to the XML Data Handler object MO_DataHandler_WDIXML_Config that we created before. Save this business object.



MO_DataHandler_Default

General | Attributes

	Pos	Name	Type	Key	Foreign	Reqd	Card	Max L	Default	App Spec I
1	1	⊟ text_xml	MO_DataHandler_WDIXML_Config	☑	☐	☐	1			
1.1	1.1	SubType	String	☐	☐	☑		1	XML	
1.2	1.2	BOPrefix	String	☐	☐	☐		1	POACK	
1.3	1.3	EntityResolver	String	☐	☐	☐		255		
1.4	1.4	DTDPath	String	☐	☐	☐		1	d:\xmlpoac	
1.5	1.5	NameHandlerClass	String	☐	☐	☐		255		
1.6	1.6	Parser	String	☐	☐	☐		1		
1.7	1.7	Validation	String	☐	☐	☐		1	false	
1.8	1.8	ClassName	String	☐	☐	☐		1	com.cross	
1.9	1.9	UseNewLine	String	☐	☐	☐		255	false	
1.10	1.10	InitialBufferSize	String	☐	☐	☐		255	2097152	
1.11	1.11	DummyKey	String	☑	☐	☐		1	1	
1.12	1.12	ObjectEventId	String							
2	2	⊞ text_fixedwidth	MO_DataHandler_DefaultFixedWidthCon	☐	☐	☐	1			
3	3	⊞ text_delimited	MO_DataHandler_DefaultDelimitedConfi	☐	☐	☐	1			
4	4	⊞ text_namevalue	MO_DataHandler_DefaultNameValueCo	☐	☐	☐	1			

Figure 2-21 Default Data Handler business object

Define meta-object MO_WDIXML_config

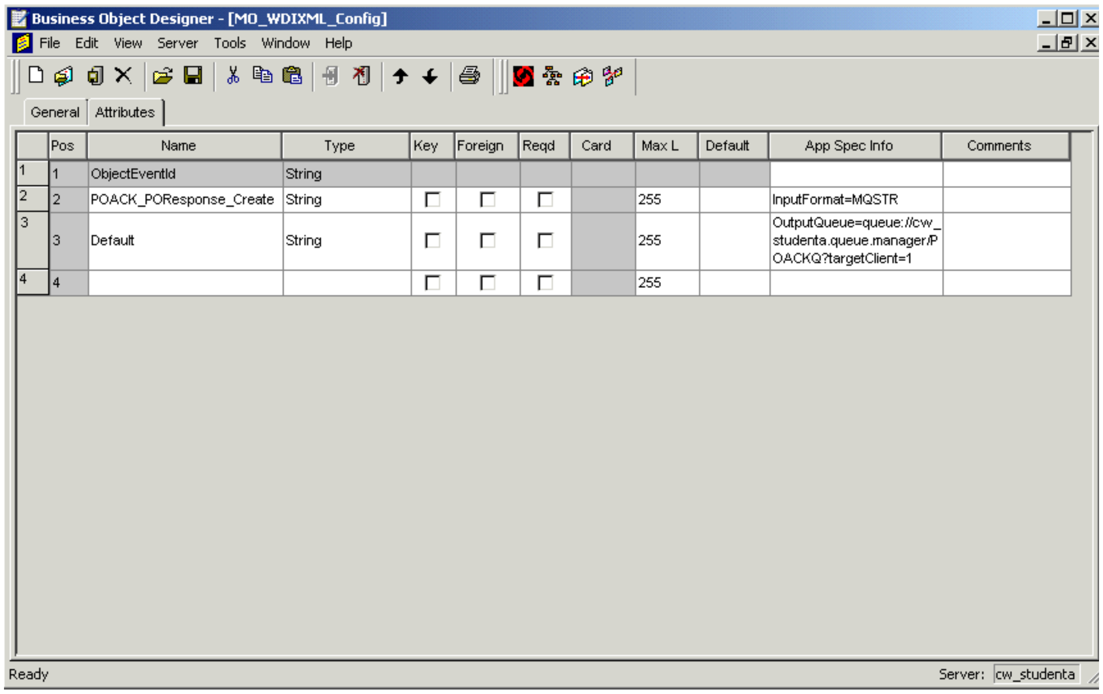
The connector requires a meta-object that describes how to convert the business object to an XML message in a queue. Open the Business Object Designer and create a new business object, named MO_WDIXML_Config. When the object designer window appears, select the tab Attributes and make the following changes:

- In the name field, add `POACK_POResponse_Create`
- In the field App Spec Info, type `InputFormat=MQSTR`
- Add another attribute. In the name field, type `Default`

- Select the check box **Key** for this attribute
- In the field App Spec Info, type:

```
OutputQueue=queue://cw_studenta.queue.manager/POACKQ?targetClient=1
```

POACKQ is the name of the triggered queue for which WebSphere MQ will launch the WDIAdapter program, as configured in 2.3.2, “Preparing EDI documents” on page 75. Replace cw_studenta.queue.manager with the name of your queue manager. The option targetClient=1 instructs the ICS to generate a standard WebSphere MQ message, instead of a JMS message.


Business Object Designer - [MO_WDIXML_Config]
File Edit View Server Tools Window Help
General Attributes
Ready
Server: cw_studenta

	Pos	Name	Type	Key	Foreign	Reqd	Card	Max L	Default	App Spec Info	Comments
1	1	ObjectEventId	String								
2	2	POACK_POResponse_Create	String	☐	☐	☐		255		InputFormat=MQSTR	
3	3	Default	String	☐	☐	☐		255		OutputQueue=queue://cw_studenta.queue.manager/POACKQ?targetClient=1	
4	4			☐	☐	☐		255			

Figure 2-22 Business object MO_WDIXML_Config

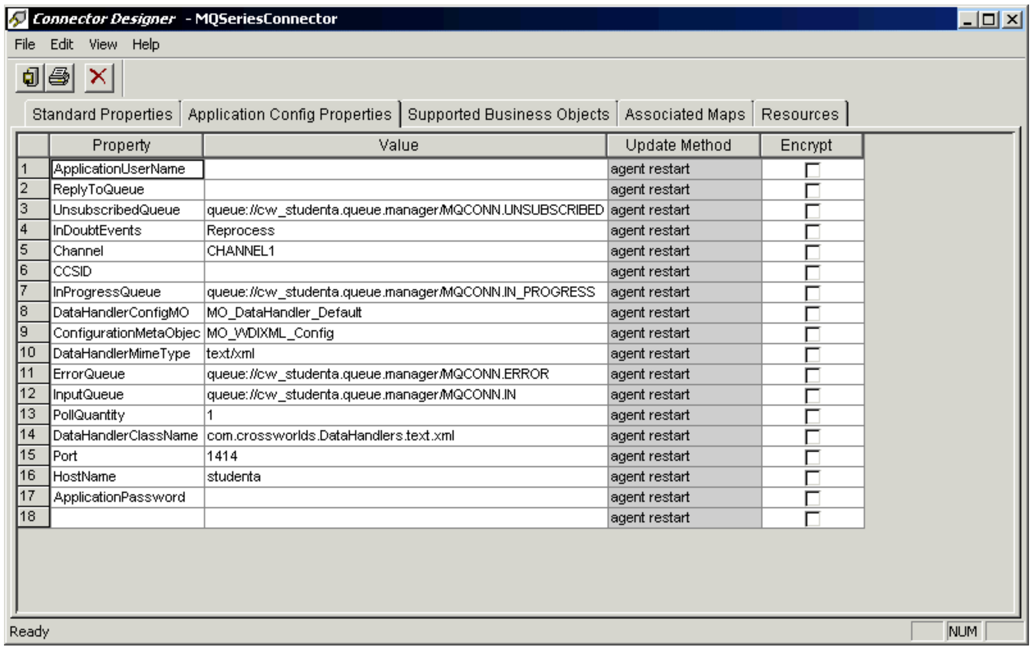
Configuring the MQSeries connector

Expand the folder Connectors in the System Manager and double-click the object **MQSeries Connector**. Click the tab **Connector Agent** to specify the connector specific properties, as detailed in Table 2-1 on page 91.

Table 2-1   Connector properties

Property	Value
InDoubtEvents	Reprocess
Channel	CHANNEL1
InProgressQueue	queue://cw_studenta.queue.manager/MQCONN .IN_PROGRESS
DataHandlerConfigMO	MO_DataHandler_Default
ConfigurationMetaObject	MO_WDIXML_Config
DataHandlerMimeType	text/xml
Port	1414
Hostname	studenta

Figure 2-23 shows the Connector Designer window where you need to specify the values listed in Table 2-1.


Connector Designer - MQSeriesConnector
File Edit View Help
Standard Properties
Application Config Properties
Supported Business Objects
Associated Maps
Resources
Ready
NUM

	Property	Value	Update Method	Encrypt
1	ApplicationUserName		agent restart	☐
2	ReplyToQueue		agent restart	☐
3	UnsubscribedQueue	queue://cw_studenta.queue.manager/MQCONN.UNSUBSCRIBED	agent restart	☐
4	InDoubtEvents	Reprocess	agent restart	☐
5	Channel	CHANNEL1	agent restart	☐
6	CCSID		agent restart	☐
7	InProgressQueue	queue://cw_studenta.queue.manager/MQCONN.IN_PROGRESS	agent restart	☐
8	DataHandlerConfigMO	MO_DataHandler_Default	agent restart	☐
9	ConfigurationMetaObjec	MO_WDIXML_Config	agent restart	☐
10	DataHandlerMimeType	text/xml	agent restart	☐
11	ErrorQueue	queue://cw_studenta.queue.manager/MQCONN.ERROR	agent restart	☐
12	InputQueue	queue://cw_studenta.queue.manager/MQCONN.IN	agent restart	☐
13	PollQuantity	1	agent restart	☐
14	DataHandlerClassName	com.crossworlds.DataHandlers.text.xml	agent restart	☐
15	Port	1414	agent restart	☐
16	HostName	studenta	agent restart	☐
17	ApplicationPassword		agent restart	☐
18			agent restart	☐

Figure 2-23   Configuring the MQSeries connector

Select then the tab Supported Business Objects. Click the blank cell under the heading Business Object Name. A drop-down box will appear. Select **POACK_POResponse** from the list and select the check box **Agent Support**. Add the meta-object MO_DataHandler_default to this table and select the check box **Agent Support**.

When finished, select **File -> Save to Server**. During the save, you may receive warnings about the need to restart the connector. You can accept those warnings. When the save process is finished, switch to the System Manager and right-click the MQSeries connector in the folder Connectors, then stop and restart the connector.

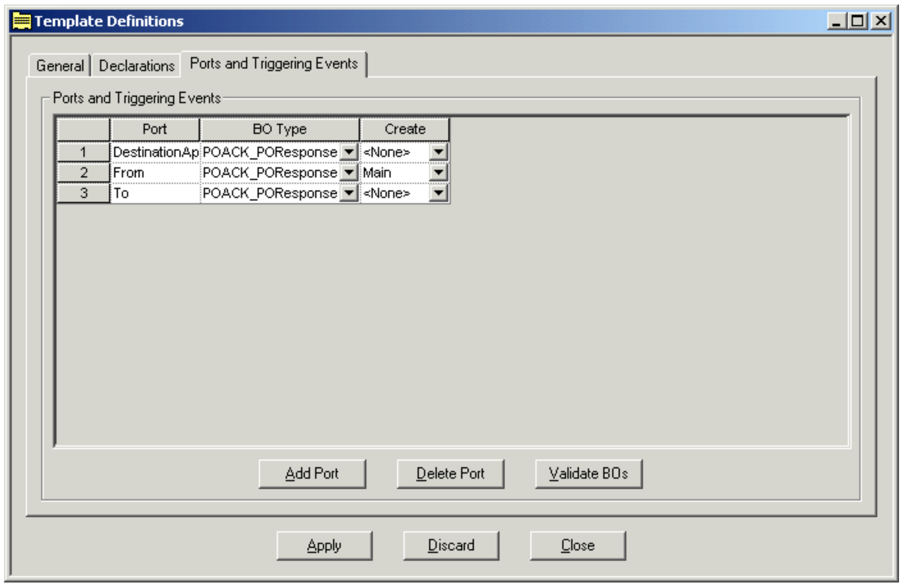
2.4.3 Developing a test collaboration

The next step is the development of a collaboration that will generate the POResponse document for processing by WebSphere Data Interchange. Open the System Manager and expand the folder Collaboration Templates. Locate the template CollaborationFoundation and copy/paste it in the folder Collaboration Templates.

Note: If your installation of the Interchange Server does not have this template, you can find an import file for this template as part of the additional material for this redbook. Refer to Appendix B, “Additional material” on page 217.

Name the copied template WDI_Outbound_Template. Open the new template in the Process Designer, by double-clicking it. Select **Template -> Open Template Definitions** to update the template. Select the tab Ports and Triggering Events. Update the BO Name for each port and set it to POACK_POResponse. Change the field Create for the row From to Main (see Figure 2-24).

Apply the changes and compile the updated template.You can delete the port DestinationAppRetrieve, but that is not required.



	Port	BO Type	Create
1	DestinationAp	POACK_POResponse	<None>
2	From	POACK_POResponse	Main
3	To	POACK_POResponse	<None>

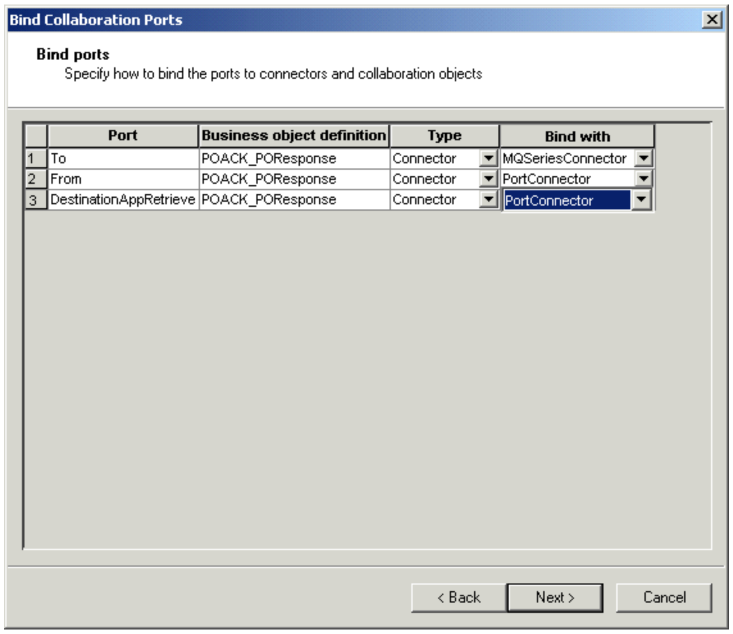
Figure 2-24 Update the template definitions

Now that we have a template that fits our needs, we can create a collaboration object. Right-click the folder Collaborations and select **New collaboration object** from the context menu. Select the template WDI_Outbound_Template and name the new collaboration WDI_Outbound. Select **Next**. Bind now the ports to connectors, as shown in Figure 2-25 on page 93.

Set the From port to the PortConnector and the To port to the MQSeriesConnector. Set also the DestinationAppRetriever port to the PortConnector, if you have not deleted this port in the template.

Note: If you do not see the PortConnector as an available choice, you need to update the list of supported business objects for the PortConnector and include the business object POACK_POResponse.

Click **Next** twice and click **Finish** to complete the definition of this collaboration.



Bind Collaboration Ports

Bind ports

Specify how to bind the ports to connectors and collaboration objects

	Port	Business object definition	Type	Bind with
1	To	POACK_POResponse	Connector	MQSeriesConnector
2	From	POACK_POResponse	Connector	PortConnector
3	DestinationAppRetrieve	POACK_POResponse	Connector	PortConnector

Figure 2-25 Bind the ports of the collaboration

If all steps went well, the System Manager will now show a graphical representation of the collaboration, as shown in Figure 2-26.

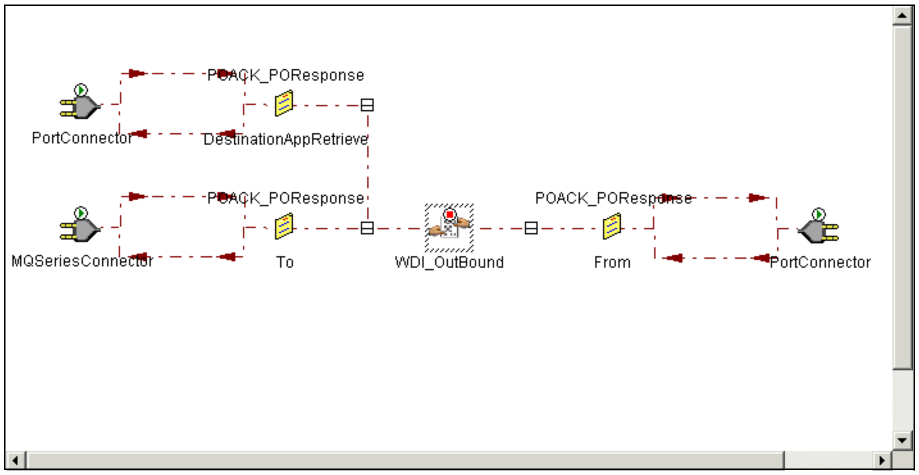


Figure 2-26 WDI_Outbound collaboration

Finally, start the collaboration by selecting **Component -> Start WDI_Outbound**. Or select the start command from the context menu of the collaboration.

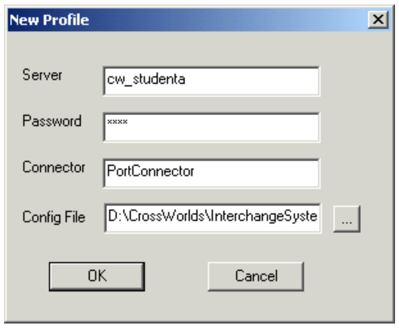
2.4.4 Using the Test Connector

The final step is to use the Test Connector tool to drive the collaboration. Before you start this tool, you should perform a number of validation tasks.

- Expand the folder Connectors in the System Manager and verify that the MQSeriesConnector is started.
- Verify that the PortConnector is started. Sometimes this connector is in a paused state, which is usually caused by a missing queue in WebSphere MQ. If you can not get the PortConnector to start, verify that the queue AP/PORTCONNECTOR/CW_STUDENTA exist. Replace CW_STUDENTA with the name of your ICS. If this queue does not exist, create it with default attributes.
- Start the MQSeries Connector that we configured before by selecting **Start -> Programs -> IBM CrossWorlds -> Connectors -> MQSeries Connector**. Verify in the output that the connector starts correctly. You can also start the System Monitor (from the Tools menu in the System Manager) and verify that the agent state of the MQSeries Connector is Active.

Start now the Test Connector by selecting **Start -> Programs -> IBM CrossWorlds -> Connectors -> Test Connector**. When the tool is started, select **File -> New Profile**, which will bring up a profile selection window. Select **Add** to create a new profile.

Provide the name of the server, the password of the admin user ID (usually the word null) and the name of the connector that we're going to simulate: PortConnector. You can leave the field Config File blank. The test connector should be able to locate the configuration file by itself. Alternatively, you should point to the ICS configuration file manually (usually D:\CrossWorlds\InterchangeSystem.cfg). Select **OK** to close this window.

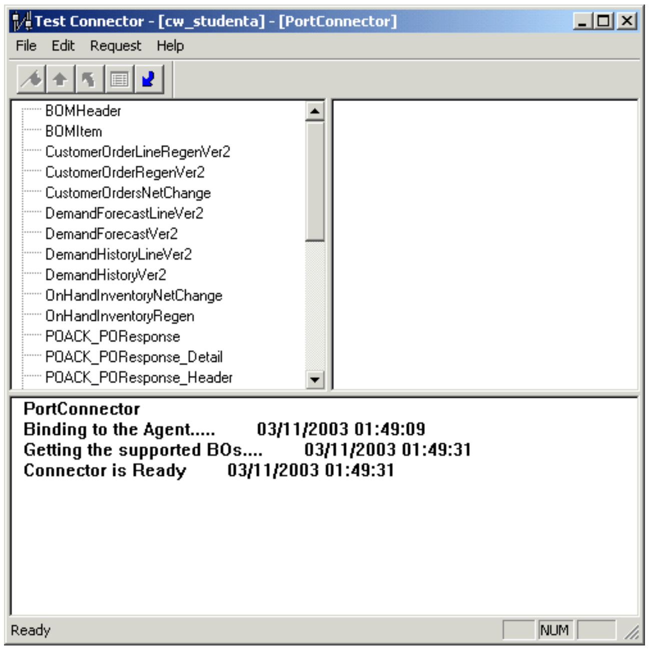

New Profile

Server: cw_studenta
Password: xxxx
Connector: PortConnector
Config File: D:\CrossWorlds\InterchangeSyste

Figure 2-27 Create a new profile for the test connector

Select now the new profile in the profile list window and click **OK**. The test connector is now loaded with the correct profile. Select **File -> Connect Agent** to connect to the ICS.

When the connection succeeds, the test connector will list all business objects that are supported by the port connector, as shown in Figure 2-28 on page 95.

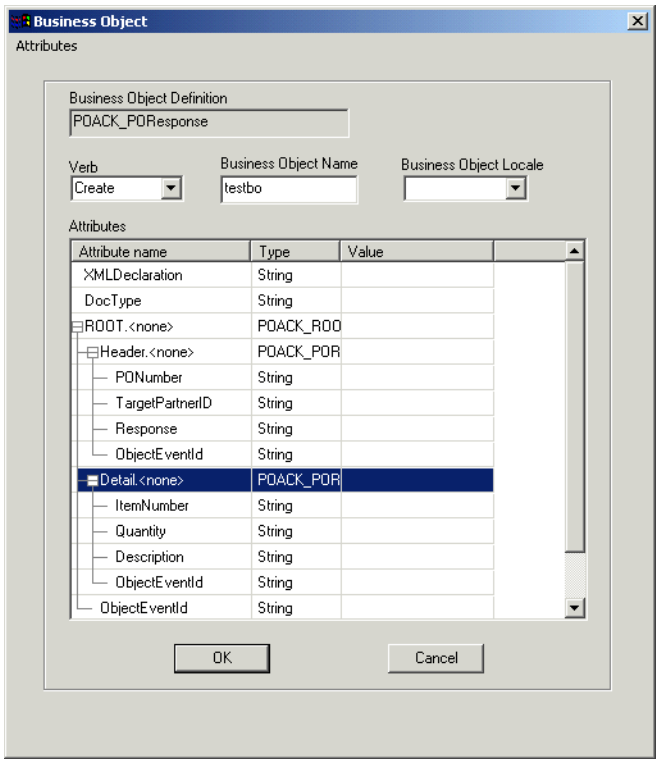


BOMHeader
BOMItem
CustomerOrderLineRegenVer2
CustomerOrderRegenVer2
CustomerOrdersNetChange
DemandForecastLineVer2
DemandForecastVer2
DemandHistoryLineVer2
DemandHistoryVer2
OnHandInventoryNetChange
OnHandInventoryRegen
POACK_POResponse
POACK_POResponse_Detail
POACK_POResponse_Header

```
PortConnector
Binding to the Agent.....        03/11/2003 01:49:09
Getting the supported BOs....        03/11/2003 01:49:31
Connector is Ready        03/11/2003 01:49:31
```

Figure 2-28 Running the test connector

Double-click the business object **POACK_POResponse** to bring up the next window. Select the verb **Create** and name the instance of the business object (for example `testbo`). Right-click the element **ROOT** and select **Add Instance**. Now expand the ROOT element which will list two child elements, Header and Detail. Right-click these elements too and select **Add Instance** each time. The business object window should now look as shown in Figure 2-29.



Business Object Definition: POACK_POResponse

Verb: Create

Business Object Name: testbo

Business Object Locale:

Attributes

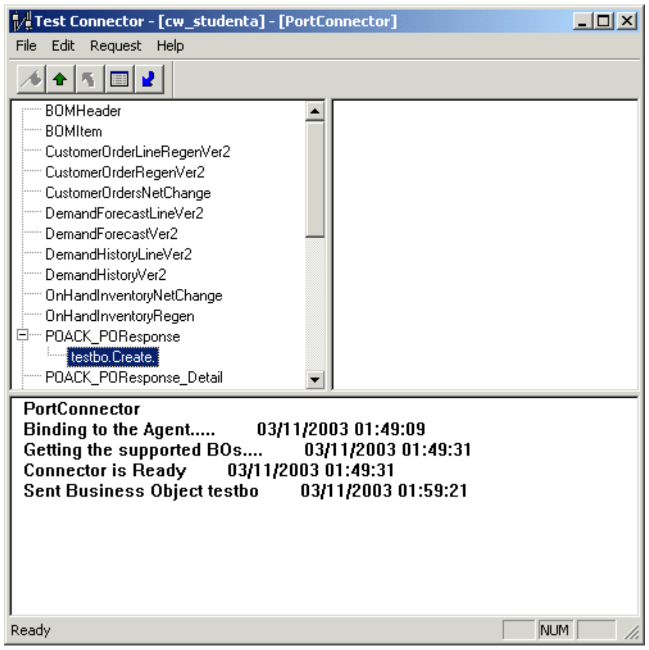
Attribute name	Type	Value
XMLDeclaration	String	
DocType	String	
ROOT.<none>	POACK_ROO	
Header.<none>	POACK_POR	
PONumber	String	
TargetPartnerID	String	
Response	String	
ObjectEventId	String	
Detail.<none>	POACK_POR	
ItemNumber	String	
Quantity	String	
Description	String	
ObjectEventId	String	
ObjectEventId	String	

Figure 2-29 Setting values for the business object

Provide values for the six data elements of this business object and click **OK**.

Important: Since this business object will result in a message that is going to be processed by WebSphere Data Interchange, you should provide data that makes sense for WebSphere Data Interchange. Setting a random value for TargetPartnerID will likely result in an unprocessed document within WebSphere Data Interchange.

This will bring you back to the main window of the test connector. Select the new business object in the tree structure and select **Request -> Send**.



```
PortConnector
Binding to the Agent.....        03/11/2003 01:49:09
Getting the supported BOs....        03/11/2003 01:49:31
Connector is Ready       03/11/2003 01:49:31
Sent Business Object testbo        03/11/2003 01:59:21
```

Figure 2-30 Business object is being sent

Open now WebSphere MQ Explorer and browse the queue POACKQ which should contain an XML message representing this business object. However, if the WebSphere MQ Trigger Monitor is still running, your message might be consumed already and you may need to inspect the output queues of WebSphere Data Interchange. And, if iSoft's P2PAgent is still running, your message might be gone to a trading partner.

This completes the basic integration of the Interchange Server in a solution with WebSphere Data Interchange. The collaboration can now be extended to include ports to real back-office applications and at that time you will probably need to develop some maps in the System Manager.

2.4.5 Inbound flow

The previous sections described in detail the integration process for the outbound flow. The steps of integrating the InterChange Server in the inbound flow are quite similar.

Business object

First, we need again a business object to represent the incoming purchase order. The DTD, listed in Example 2-15 on page 68, can be imported in the InterChange Server using the XML ODA as described in 2.4.1, "Creating business objects" on page 84.

Specify the following values for the PO DTD:

- Root element: `PO`
- Top Level element: `PO`
- BOPrefix: `PO`

The MQSeriesInbound connector

The next step is to create an additional MQSeries connector. Perform the following steps:

- Copy/paste the existing MQSeriesConnector object in the folder Connectors in the System Manager. Name it MQSeriesInboundConnector.
- Open a file browser and find the directory MQSeries in the connectors directory of the ICS installation. Copy the whole directory and name it MQSeriesInbound.
- Open the folder Connectors in the Start menu and copy/paste the existing short-cut MQSeries Connector as MQSeriesInbound Connector.
- Open the properties of this new short-cut and update the field Target:

  ```
  D:\CrossWorlds\connectors\MQSeriesInbound\start_MQSeries.bat MQSeriesInbound cw_studenta
  -cD:\CrossWorlds\connectors\MQSeriesInbound\MQSeriesAgentConfig.cfg
  ```

  MQSeries has been replaced three times with MQSeriesInbound. Update also the field Start in to the name of the new directory:

  ```
  D:\CrossWorlds\connectors\MQSeriesInbound\
  ```

- Define a new queue AP/MQSERIESINBOUNDCONNECTOR/CW_STUDENTA on the queue manager used by the ICS. Replace CW_STUDENTA with the name of your ICS.
- Restart the ICS. After the restart, verify that the connector is running via the System Manager.
- Start the connector agent via the short-cut in the Programs folder and verify that the Agent State in the System Monitor is active.

Open the PortConnector object and update the supported business objects. Include business object PO_PO in the list and make sure to check the field Agent Support.

Create meta-objects

Once you have verified that the new connector can be started, proceed with the definition of meta-objects. Open the meta-object MO_DataHandler_WDIXML_Config and safe it as MO_DataHandler_CWXML_Config. Make the following changes:

- Provide the path to the location of the DTD and the file name
- Set the BOPrefix to `PO`

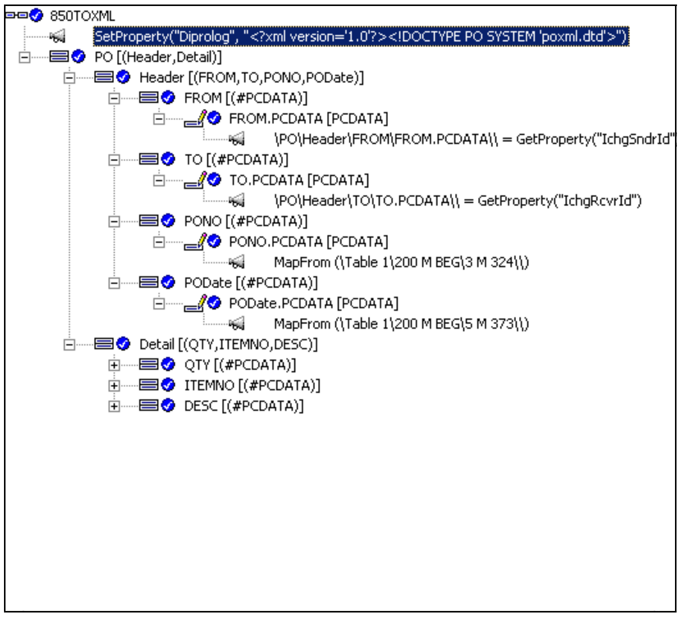
Define meta-object MO_CWXML_Config. You can copy the meta-object MO_WDIXML_Config. Rename the field POACK_POResponse to PO_PO.

Define meta-object MO_DataHandler_Inbound_Default. You can copy it from MO_DataHandler_Default. For the MIME type text/xml, set the field type to `MO_DataHandler_CWXML_Config`.

Open the connector object MQSeriesInboundConnector and switch to the tab Application Config Properties. Set the value of the property DataHandlerConfigMO to `MO_DataHandler_Inbound_Default`. Set the value of the property ConfigurationMetaObject to `MO_CWXML_Config`. Set the value of the property InputQueue to `queue://cw_studenta.queue.manager/purchase.orders`. Save the changes and restart the connector.

Verify the map in WebSphere Data Interchange

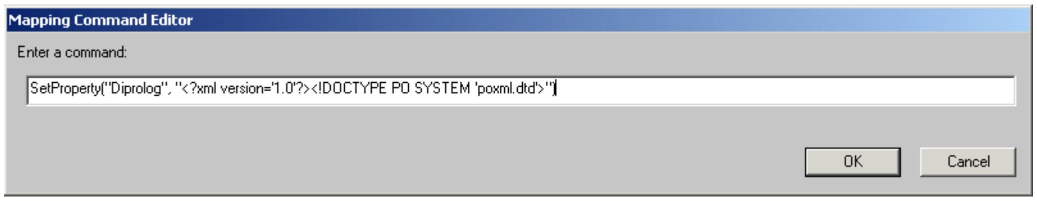
The ICS requires that the incoming XML message contains a DOCTYPE statement, that includes the name of the DTD. To make sure that the XML document contains the DTD name, review the map 850TOXML in WebSphere Data Interchange. Open the map editor and verify if you have a SetProperty call for the property Diprolog, as shown in Figure 2-31.



```
850TOXML
    SetProperty("Diprolog", "<?xml version='1.0'?><!DOCTYPE PO SYSTEM 'poxml.dtd'>")
    PO [(Header,Detail)]
        Header [(FROM,TO,PONO,PODate)]
            FROM [(#PCDATA)]
                FROM.PCDATA [PCDATA]
                    \PO\Header\FROM\FROM.PCDATA\\ = GetProperty("IchgSndrId"
            TO [(#PCDATA)]
                TO.PCDATA [PCDATA]
                    \PO\Header\TO\TO.PCDATA\\ = GetProperty("IchgRcvrId")
            PONO [(#PCDATA)]
                PONO.PCDATA [PCDATA]
                    MapFrom (\Table 1\200 M BEG\3 M 324\\)
            PODate [(#PCDATA)]
                PODate.PCDATA [PCDATA]
                    MapFrom (\Table 1\200 M BEG\5 M 373\\)
        Detail [(QTY,ITEMNO,DESC)]
            QTY [(#PCDATA)]
            ITEMNO [(#PCDATA)]
            DESC [(#PCDATA)]
```

Figure 2-31 Setting the property Diprolog

If this statement does not exist, right-click the name of the map 850TOXML and select **Insert -> Command -> SetProperty** from the context menu. A mapping command editor should appear, as shown in Figure 2-32. Update the template call of SetProperty to refer to the property Diprolog and set the value to what is required for your XML document.



Mapping Command Editor

Enter a command:

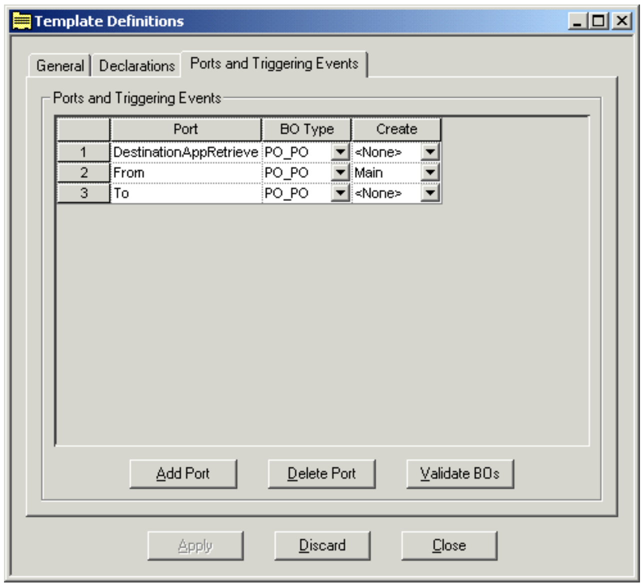
```
SetProperty("Diprolog", "<?xml version='1.0'?><!DOCTYPE PO SYSTEM 'poxml.dtd'>")
```

Figure 2-32 Adding the name of the DTD to the XML document

Save and re-compile the map.

Create the collaboration

Expand the folder Collaboration Templates in the System Manager. Copy/paste the template CollaborationFoundation and name the new template WDI_Inbound_Template. Open the new template and open its definitions. Select the tab Ports and Triggering Events (see Figure 2-33 on page 99). Set the BOType to `PO_PO` for all three ports. Set the Create field for the From port to `Main`. Apply the changes. Compile and save the template.



Ports and Triggering Events

	Port	BO Type	Create
1	DestinationAppRetrieve	PO_PO	<None>
2	From	PO_PO	Main
3	To	PO_PO	<None>

Figure 2-33 Template definitions

Create now a new collaboration WDI_Inbound from the template WDI_Inbound_Template. Bind the ports as follows:

- From port: MQSeriesInboundConnector
- To port: PortConnector
- DestinationAppRetrieve port: PortConnector

Save and start the collaboration. Check the server log and verify that you see log messages as shown in Example 2-26.

Example 2-26 ICS log

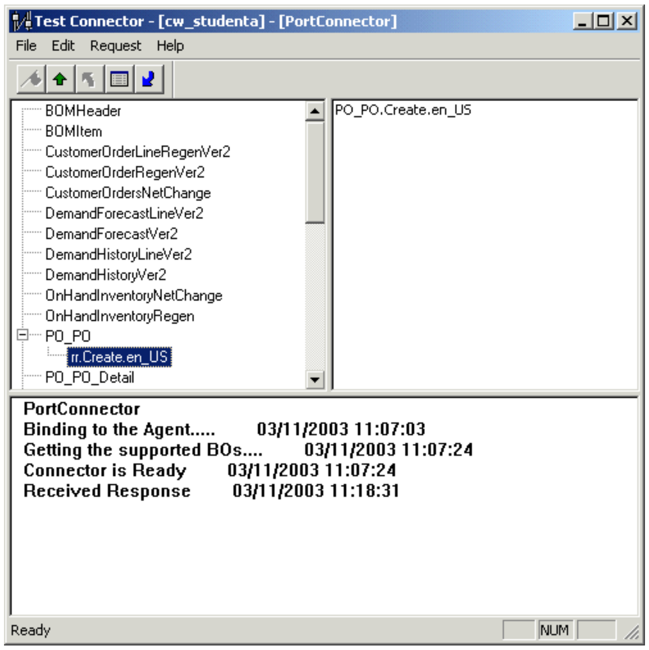
```
[System: Server] [Thread: VBJ ThreadPool Worker (#5244814)] [Type: Info] [MsgID: 31] [Mesg: Initializing
collaboration "WDI_Inbound".]
[System: Collaboration] [SS: WDI_Inbound] [Thread: VBJ ThreadPool Worker (#4968337)] [Type: Info] [MsgID:
11009] [Mesg: Subscribed to PO_PO.Create from publisher MQSeriesInboundConnector.]
[System: Collaboration] [SS: WDI_Inbound] [Thread: VBJ ThreadPool Worker (#4968337)] [Type: Info] [MsgID:
11014] [Mesg: Collaboration is active.]
```

Use the test connector

Finally, start the test connector and select the profile PortConnector. Connect to the server and create a business object of type PO_PO. Select the newly created business object and select **Request -> Accept Request**. The PortConnector acts now as an endpoint and is ready to receive PO_PO business objects.

Write now an EDI message on the queue INBOX, processed by WebSphere Data Interchange. The translated message, including a DOCTYPE, should then end on queue purchase.orders, which is monitored by the MQSeriesInbound Connector. The message will then be routed through the collaboration and end at the port connector in the test connector.

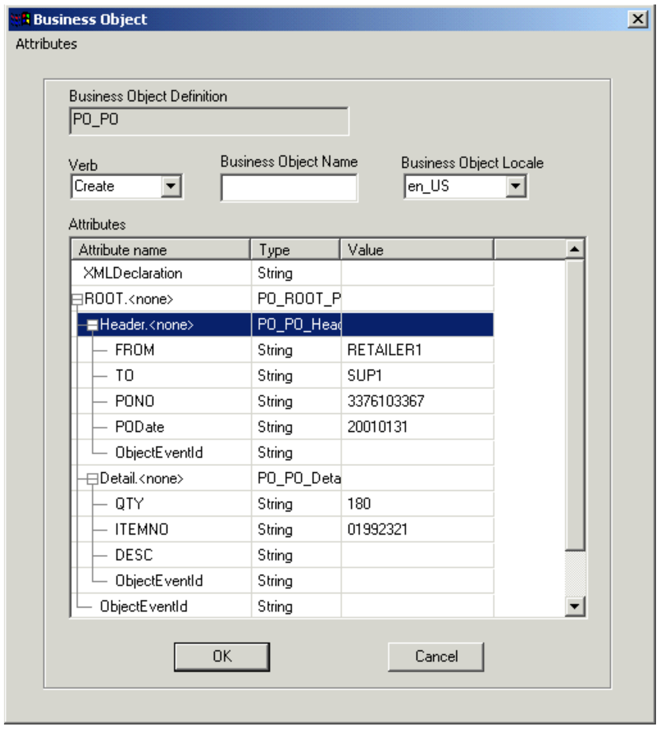
Figure 2-34 on page 100 shows the test connector window when data has arrived.



```
PortConnector
Binding to the Agent.....        03/11/2003 11:07:03
Getting the supported BOs....        03/11/2003 11:07:24
Connector is Ready        03/11/2003 11:07:24
Received Response        03/11/2003 11:18:31
```

Figure 2-34 A business object has arrived

Select the business object in the right pane to inspect the details. Figure 2-35 shows the business object representation of this XML message, which was a translation of an EDI 850 document.



Business Object Definition: PO_PO

Verb: Create

Business Object Name:

Business Object Locale: en_US

Attributes

Attribute name	Type	Value
XMLDeclaration	String	
ROOT.<none>	PO_ROOT_P	
Header.<none>	PO_PO_Head	
FROM	String	RETAILER1
TO	String	SUP1
PONO	String	3376103367
PODate	String	20010131
ObjectEventId	String	
Detail.<none>	PO_PO_Deta	
QTY	String	180
ITEMNO	String	01992321
DESC	String	
ObjectEventId	String	
ObjectEventId	String	

Figure 2-35 Details of the received business object

3

Implementing multi-product AS/2 communication with trading partners

This chapter describes the implementation of a small trading partner network with three partners where not every partner is using the same AS/2 provider. We consider a company called Supplier that is using Trading Partner Interchange (TPI) interacting with another TPI installation. We describe the setup of this network and then add a partner to it that is using iSoft's P2PAgent.

This chapter also discusses integration scenarios between TPI, WebSphere Data Interchange and the InterChange Server.

3.1 Business case

In this chapter, we will describe the use of TPI by a manufacturer of consumer products. This company, called Supplier throughout this chapter, uses TPI to receive purchase orders from a retailer, called Retailer1. Both companies use the same product for their B2B document exchange. The setup of this exchange will be discussed in detail in 3.2, “Implementing TPI between two partners” on page 103.

The company Supplier has signed a new deal with a second retailer, called Retailer2. The company Retailer2 would also like to exchange EDI documents with Supplier. However, Retailer2 imposes the use of iSoft’s P2PAgent as the preferred AS2 client. The company Supplier is not very eager to implement a second AS2 client and would like to use TPI to interact with Retailer2. In 3.3, “Communicating with an iSoft P2PAgent installation” on page 116, we describe the implementation and the interoperability of iSoft’s P2PAgent and TPI.

Figure 3-1 shows a schematic view of the infrastructure that we are going to build.

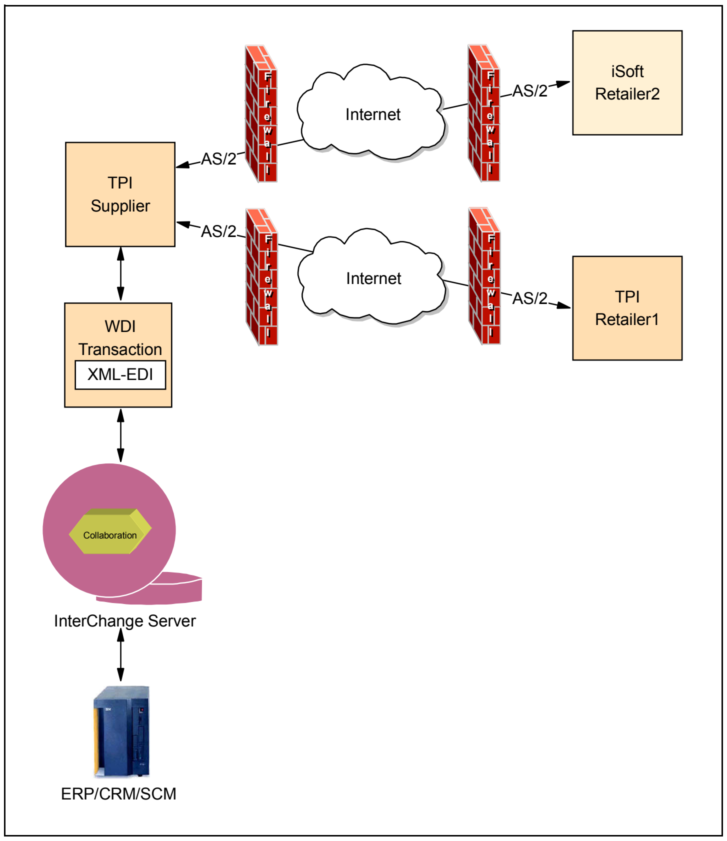


Figure 3-1 Implementation of TPI and iSoft for a supplier and two retailers

In addition to the basic implementation of B2B communication for Supplier with his two customers Retailer1 and Retailer2, we will describe several integration scenarios. Section 3.4, “Integration between WebSphere Data Interchange and TPI” on page 132 explains how the integration of TPI and WebSphere Data Interchange works. We will look at trading partner information synchronization and routing decisions. Further up the integration chain, the company Supplier might use the InterChange Server to integrate its B2B exchange with its back-office applications. Again, there are several integration scenarios, which we describe in 3.5, “Integration between the Interchange Server, WebSphere Data Interchange and TPI” on page 150.

3.2 Implementing TPI between two partners

In this section, we describe the steps to set up communication between Supplier and Retailer1 who both use TPI as their B2B gateway. Besides the basic setup of profiles, we also set up integration using WebSphere MQ for Supplier.

Figure 3-2 provides an overview of the different components within the IT infrastructure of company Supplier.

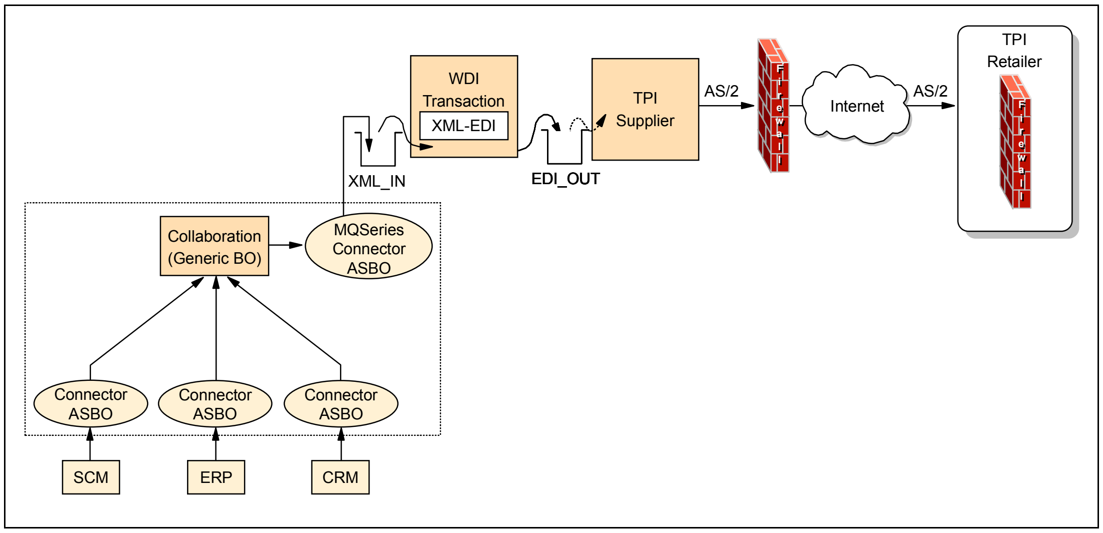


Figure 3-2 Overview of components

3.2.1 Installation of TPI for Supplier

The installation of TPI is straightforward. During the process, you will be asked for the following information:

- Registration number of the product
- Company name
- Memory limit for the TPI Server
- Destination directory

Note that the company name provided during the installation does not result in any pre-defined company profiles nor does it imply that this name should be used as the ID or profile name.

3.2.2 Company profile setup for Supplier

The very first thing you would do when using TPI is to create a company profile. Note that you might have several company profiles to represent a single physical company. Associated with a company is a public-private key pair. One can argue that you should use a different key pair for each trading partner. Thus, if you company deals with ten trading partners, this approach would result in ten different company profiles.

Start the TPI Administrator. The program will prompt you for a user ID and password. The only predefined user ID is Administrator with a blank password.

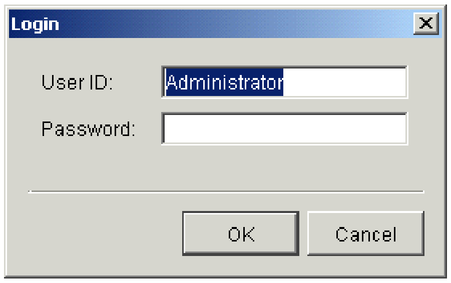


Figure 3-3 TPI login window

The initial window will appear. Select **File -> New** to create a company profile.

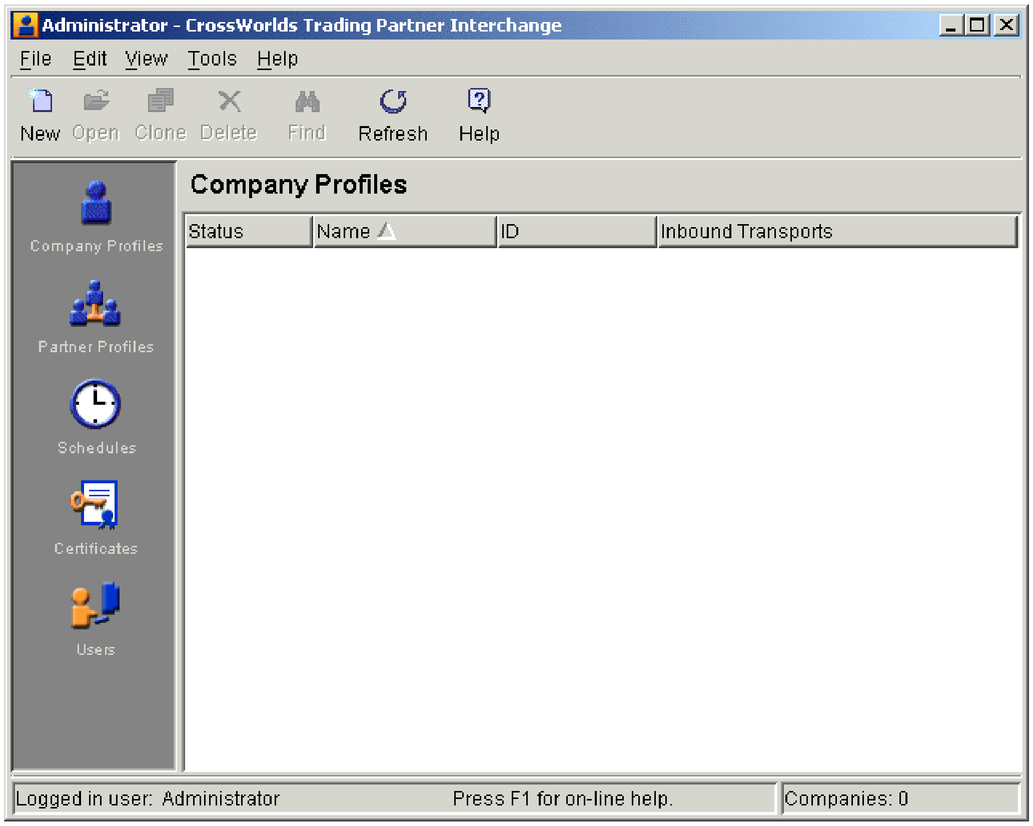
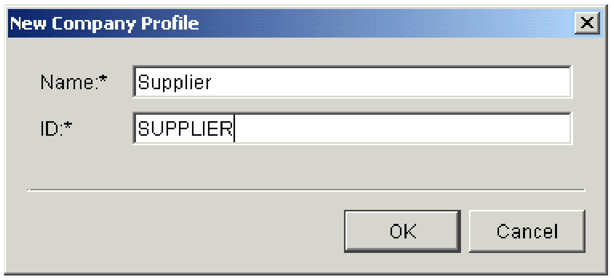

Administrator - CrossWorlds Trading Partner Interchange
File Edit View Tools Help
New Open Clone Delete Find Refresh Help
Company Profiles
Partner Profiles
Schedules
Certificates
Users
Company Profiles
Status
Name
ID
Inbound Transports
Logged in user: Administrator
Press F1 for on-line help.
Companies: 0

Figure 3-4 Initial window of TPI

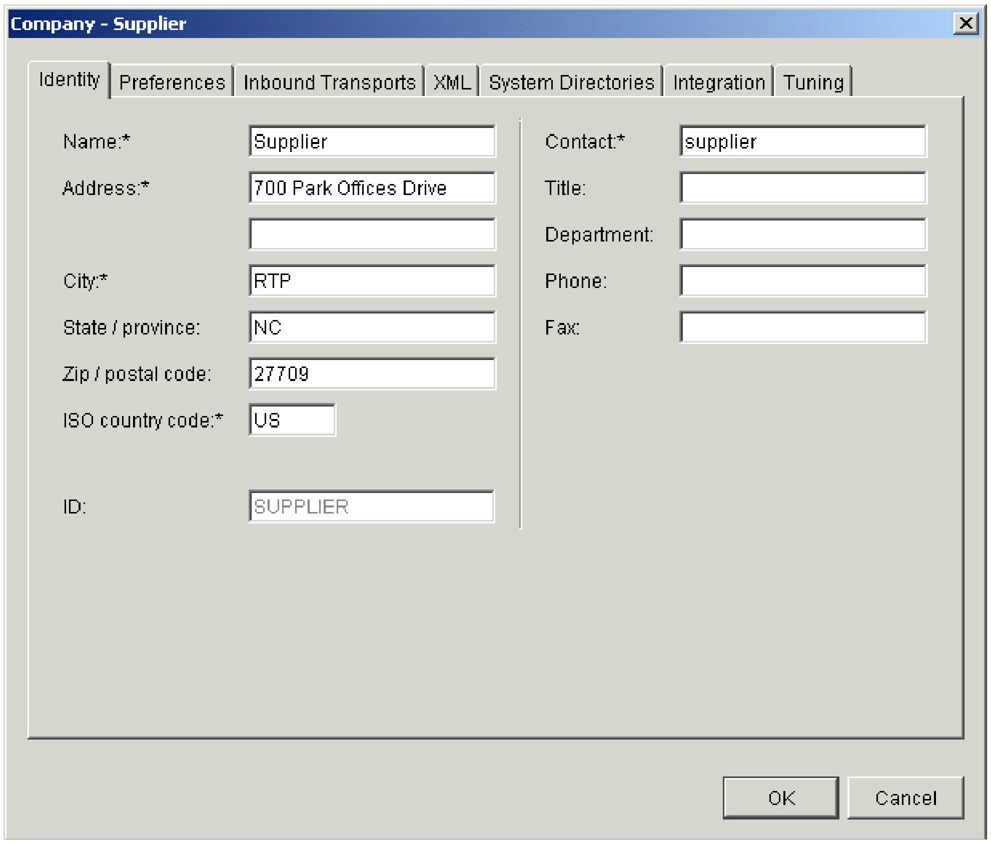
The process starts by prompting you for a trading partner name and ID, as shown in Figure 3-5 on page 105. The field ID is the concatenation of two fields in the ISA segment of an EDI document. While you can freely choose the value of the field Name, you will likely need to synchronize the value for the field ID with the configurations of other software products, such as WebSphere Data Interchange.


New Company Profile

Name:*	Supplier
ID:*	SUPPLIER

Figure 3-5 First step to create a new company profile

After you click **OK**, the process requires you to fill in a number of fields on different tabs (Figure 3-6). Provide such data as address information, contact information, and the ISO country code. The Preferences tab allows you to control document retention values and notification e-mail addresses. You can leave this tab to its default values.


Company - Supplier

Name:*	Supplier	Contact:*	supplier
Address:*	700 Park Offices Drive	Title:	
		Department:	
City:*	RTP	Phone:	
State / province:	NC	Fax:	
Zip / postal code:	27709		
ISO country code:*	US		
ID:	SUPPLIER		

Figure 3-6 Create company profile: the Identity tab

Switch to the Inbound Transports tab and click the **Add** button. From the drop-down box, select **Bundled HTTP**. Click **OK**. You will then be presented with the HTTP location that is not configurable for bundled HTTP. Click **OK** once more to return to the main window (Figure 3-7 on page 106).

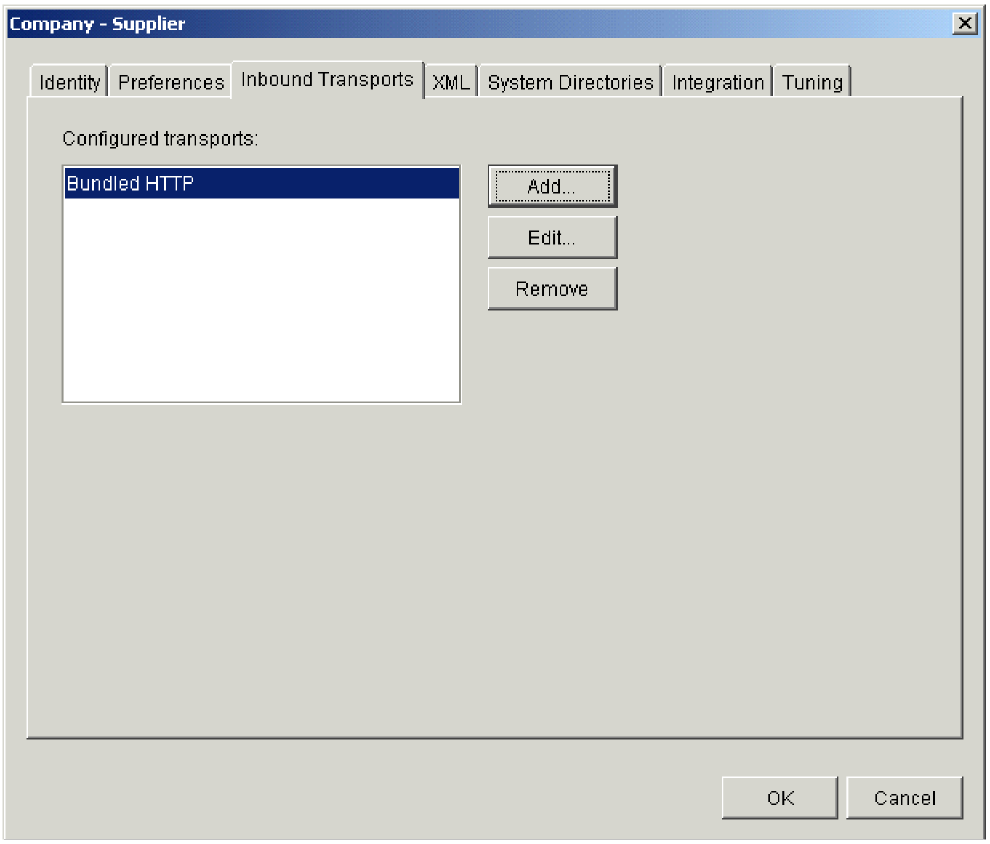
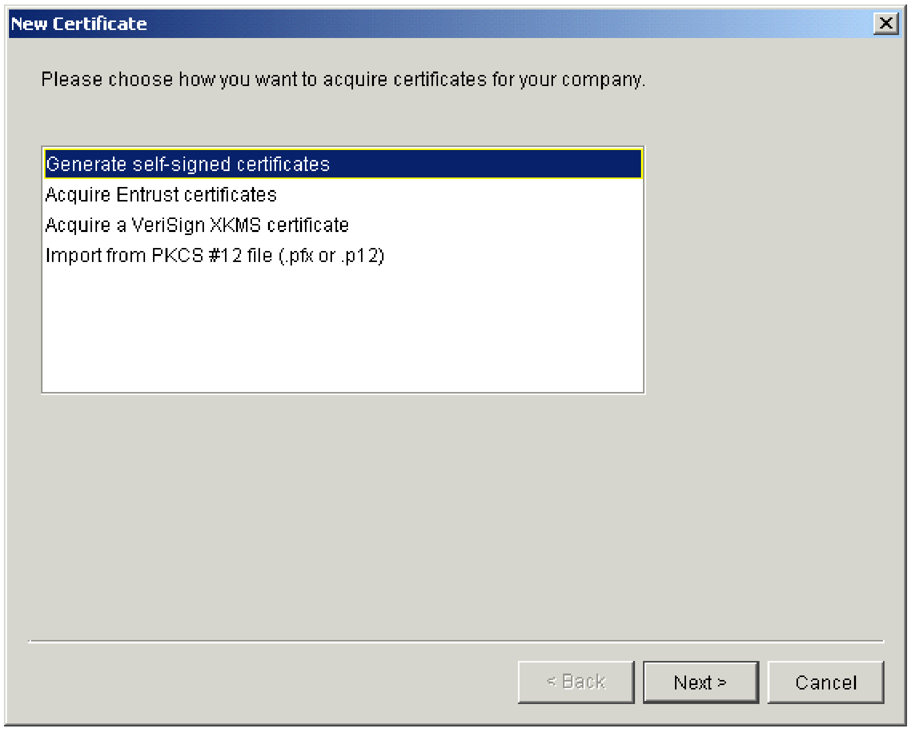


Figure 3-7 Completed inbound transport settings

For this scenario, we do not need to configure anything on the XML tab. Note that your version of TPI might disable the tabs Integration and Tuning. This depends on the type of licence that you have acquired.

For file-based integration, which we will use initially, select the tab **System Directories** and review the default directory names. If necessary, you can alter them now.

Click **OK** to save this profile. The TPI Administrator tool will prompt you as to whether you want to set up a certificate for this new company profile. Select **Yes**, which will open another window that will help you generate a new certificate (Figure 3-8 on page 107).

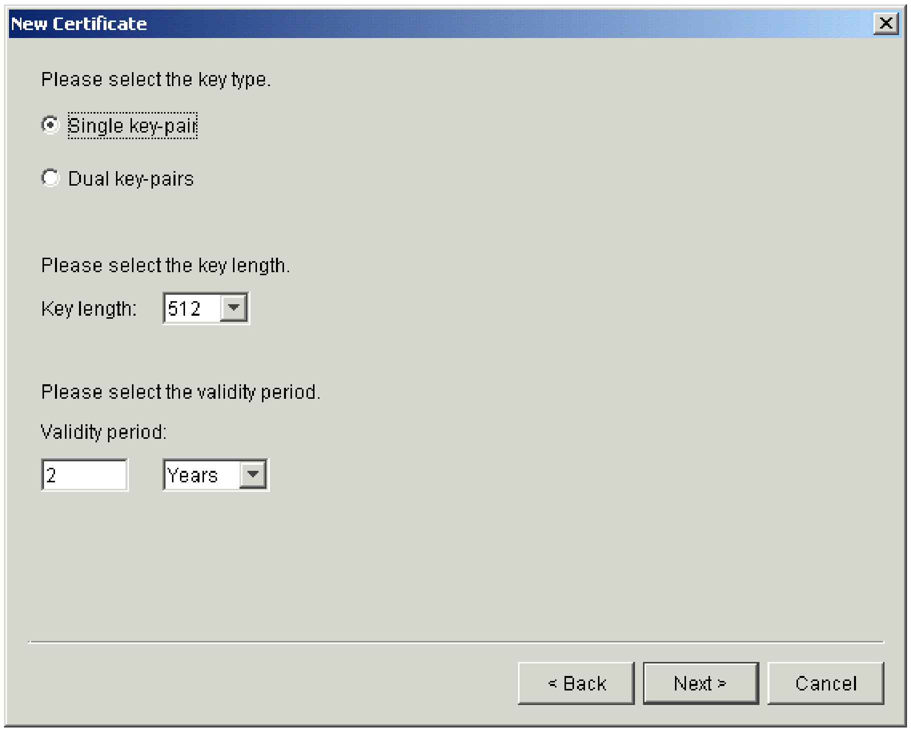


Please choose how you want to acquire certificates for your company.

- Generate self-signed certificates
- Acquire Entrust certificates
- Acquire a VeriSign XKMS certificate
- Import from PKCS #12 file (.pfx or .p12)

Figure 3-8 Create a new certificate

You can choose to generate a self-signed certificate or acquire one from a trusted authority. If you prefer to work with self-signed certificates, the next window will ask you to specify the length of the key and the validity period (Figure 3-9 on page 108). You can use keys with a length of up to 2048 bits. However, if you would like to create a setup that can interoperate with other AS2 clients, such as iSoft, you should use the default length of 512. The next window summarizes your selections. Select **Finish** to close the configuration process and start the actual generation. Note that the key generation might take a while, depending on the speed of your processor.


New Certificate

Please select the key type.

(●) Single key-pair

( ) Dual key-pairs

Please select the key length.

Key length: 512

Please select the validity period.

Validity period:

2 Years

Figure 3-9 Settings for the new certificate

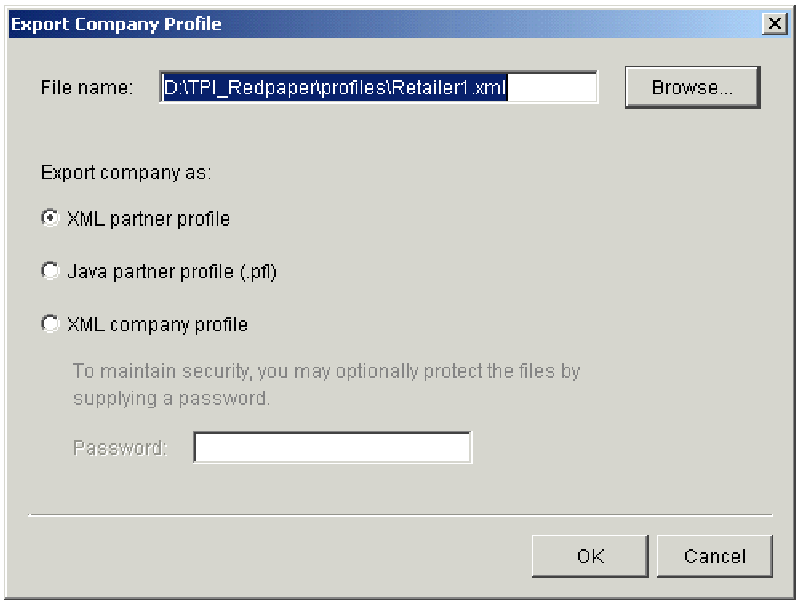
This completes the setup of a company profile. A similar procedure is required for the setup of TPI for trading partner Retailer1.

3.2.3 Partner profile setup for Retailer1 at Supplier

To connect the two TPI environments, you can either manually create a partner profile or export the company profiles on both sides and exchange the export files in a secure way. When both trading partners use TPI, it is probably easier to just export the profiles and import them on the other side.

To export the company profile, make sure that the Administrator window is currently displaying Company Profiles. If not, click the **Company Profiles** button on the left-hand side. Select **File -> Export** to initiate the export process (Figure 3-10 on page 109).

Select the radio button to export your company as an XML partner profile. This means that the export file can be used only as an import file for partners at another TPI installation. If you would like to import the company profile itself on another TPI Server (for example, to stage a profile from test to production), you should use the option XML company profile. Click the **Browse** button to specify a file name and location to store the export file. This file can now be exchanged with your trading partners.



Export Company Profile

File name: D:\TPI_Redpaper\profiles\Retailer1.xml

Export company as:

(●) XML partner profile

( ) Java partner profile (.pfl)

( ) XML company profile

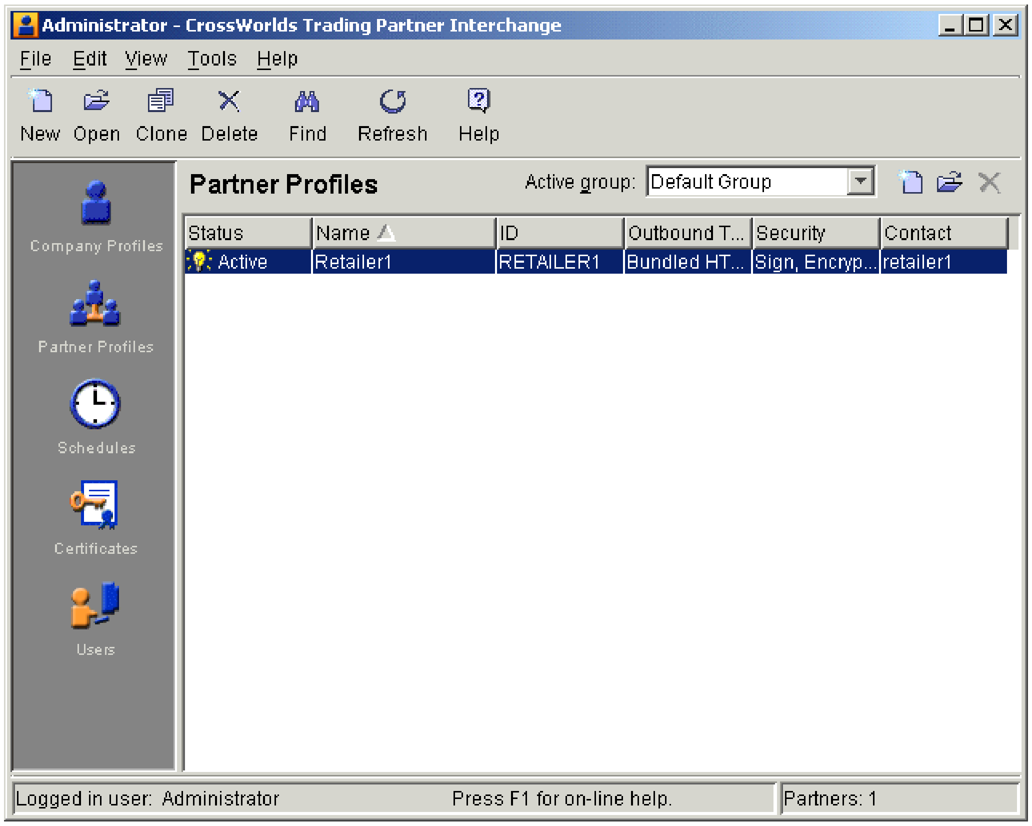
To maintain security, you may optionally protect the files by supplying a password.

Password:

Figure 3-10 Export company profile

You can now import the company profile as a partner profile. Switch to the Partner Profiles window (click **View -> Partner Profiles**) and select **File -> Import**. Select the import file on the file system and click **OK**. The profile will now be loaded.

Double-click the profile to inspect it or to make any changes to it. For example, the target HTTP location, by default, contains only the host name and not the domain name of the partner. You might need to change this on the Outbound Transports tab. Select **Bundled HTTP**, click the **Edit** button, and update the URL. Also, you might need to add information about firewalls and firewall authentication.


Administrator - CrossWorlds Trading Partner Interchange
File Edit View Tools Help
New Open Clone Delete Find Refresh Help
Company Profiles
Partner Profiles
Schedules
Certificates
Users

Partner Profiles

Active group: Default Group

Status	Name	ID	Outbound T...	Security	Contact
Active	Retailer1	RETAILER1	Bundled HT...	Sign, Encryp...	retailer1

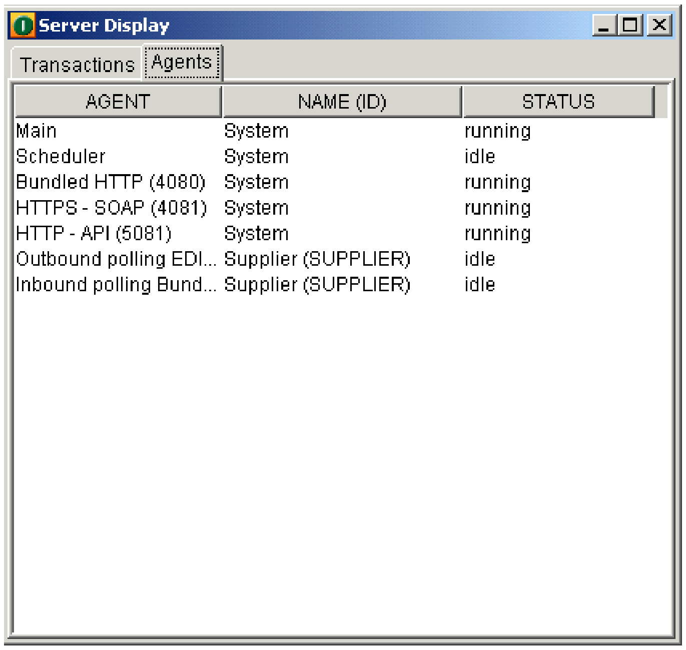
Logged in user: Administrator   Press F1 for on-line help.   Partners: 1

Figure 3-11 Partner Profiles window after import

3.2.4 Validation of the setup

Before proceeding with any type of integration, it is probably a good idea to perform some simple validation of the current setup. First, from the Start menu, start the TPI Server and the View Server Log application.

When the server is started, a window is shown with two tabs, labeled Transactions and Agents. Select the tab **Agents** and verify that the system is running. You can also inspect the log to validate this. The outbound communication is implemented by polling the outbound directories every 30 seconds. This value is tunable in certain versions of TPI using the Tuning tab for the company profile. By monitoring the tab Agents in the window Server Display for 30 seconds, you can see that the task Outbound polling EDI runs for a moment and then returns to the idle status. This is an indication that the server is operating as expected.


Server Display

Transactions | Agents

AGENT	NAME (ID)	STATUS
Main	System	running
Scheduler	System	idle
Bundled HTTP (4080)	System	running
HTTPS - SOAP (4081)	System	running
HTTP - API (5081)	System	running
Outbound polling EDI...	Supplier (SUPPLIER)	idle
Inbound polling Bund...	Supplier (SUPPLIER)	idle

Figure 3-12 TPI Server Display

A common problem is a collision on TCP ports. Usually, TPI uses port 1090, but this port can already be used by other applications such as WebSphere Application Server. To avoid this, you can start TPI first before you start WebSphere Application Server.

Assuming that all server components start well, we can proceed to the next task: the actual exchange of some EDI documents. To exchange documents quickly, you can use the Document Generator tool, which you can start from the Start menu.

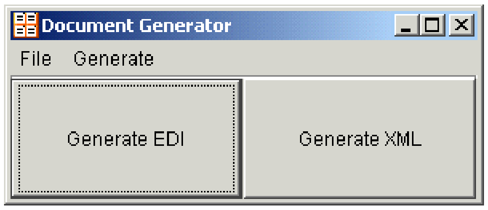
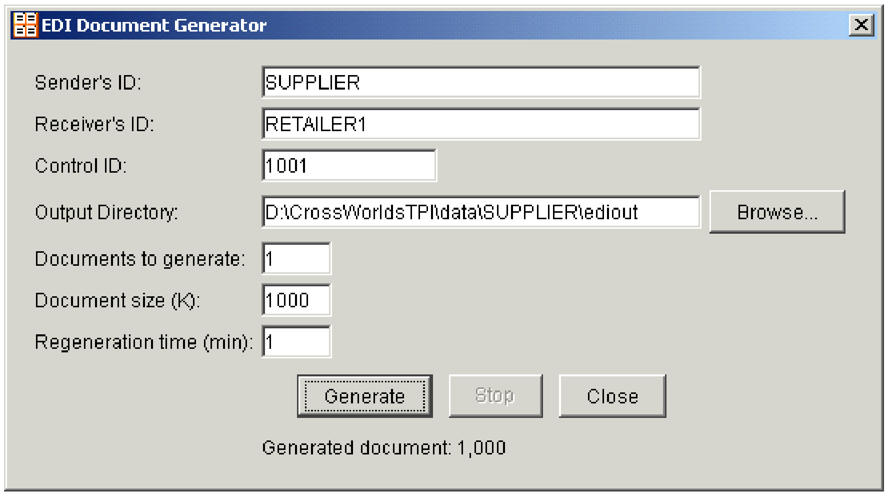


Figure 3-13 The Document Generator tool

Click the **Generate EDI** button to provide details about the document that you want to generate.

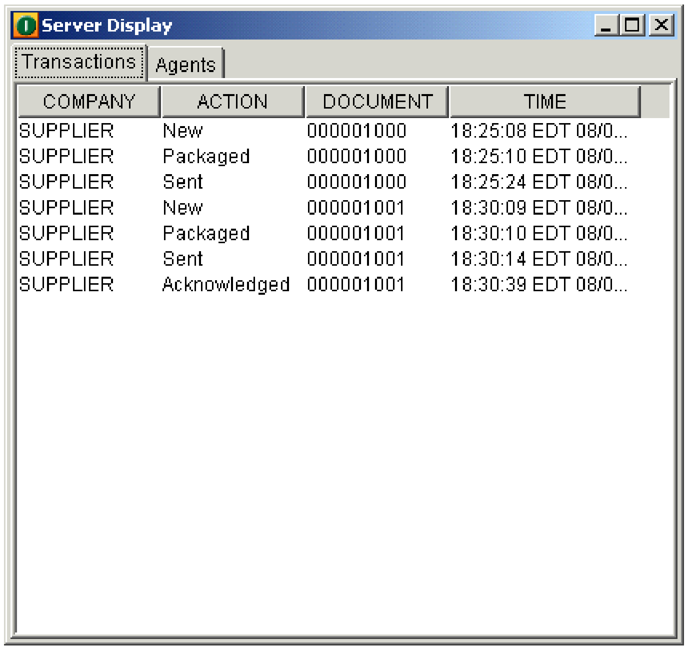

EDI Document Generator

Sender's ID:	SUPPLIER
Receiver's ID:	RETAILER1
Control ID:	1001
Output Directory:	D:\CrossWorldsTPI\data\SUPPLIER\ediout
Documents to generate:	1
Document size (K):	1000
Regeneration time (min):	1

Generated document: 1,000

Figure 3-14 EDI Document Generator

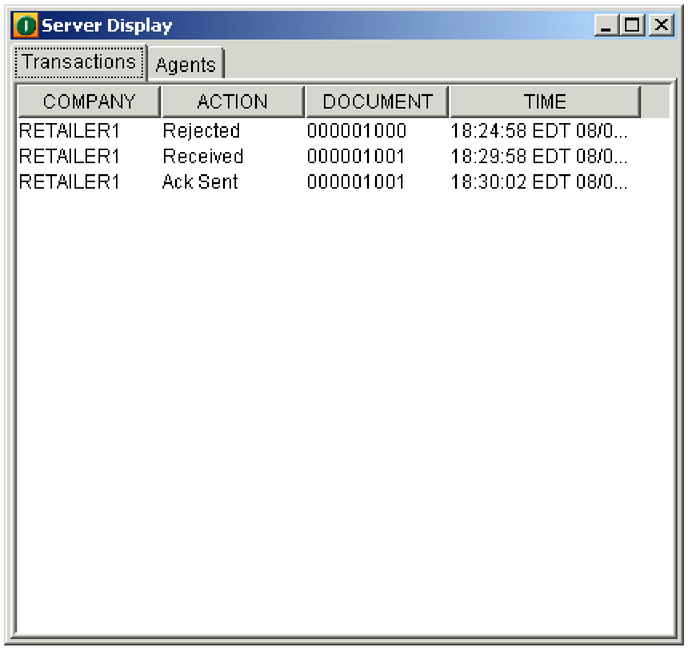
Fill in the Sender ID and Receiver ID fields. Provide a value for the Control ID field also, since this is used in the document. You can use this, for example, to find out what both partners do when the same document is sent twice. The output directory should be set to the directory that you specify in the company profile (System Directories tab). Here, the default destination was used. Click the **Generate** button. When the document is generated, click the **Stop** button. If you do not do so, the generation will continue and repeat every minute. Now switch to the Server Display window (Figure 3-15) or the Server Log window to track the send process. Note that TPI uses polling techniques. The generated file might not be processed immediately.


Server Display

COMPANY	ACTION	DOCUMENT	TIME
SUPPLIER	New	000001000	18:25:08 EDT 08/0...
SUPPLIER	Packaged	000001000	18:25:10 EDT 08/0...
SUPPLIER	Sent	000001000	18:25:24 EDT 08/0...
SUPPLIER	New	000001001	18:30:09 EDT 08/0...
SUPPLIER	Packaged	000001001	18:30:10 EDT 08/0...
SUPPLIER	Sent	000001001	18:30:14 EDT 08/0...
SUPPLIER	Acknowledged	000001001	18:30:39 EDT 08/0...

Figure 3-15 Server Display window at Supplier

Check the Server Display window at the target machine and verify that the file has been received correctly. Notice that a first document was rejected. The second document was received correctly by Retailer1 (Figure 3-16 on page 113). Usually, the Server Log will contain enough information to determine why a document was rejected. Note also that the target machine has acknowledged the receipt of the file. This reply message is sometimes called a message disposition notification (MDN).



COMPANY	ACTION	DOCUMENT	TIME
RETAILER1	Rejected	000001000	18:24:58 EDT 08/0...
RETAILER1	Received	000001001	18:29:58 EDT 08/0...
RETAILER1	Ack Sent	000001001	18:30:02 EDT 08/0...

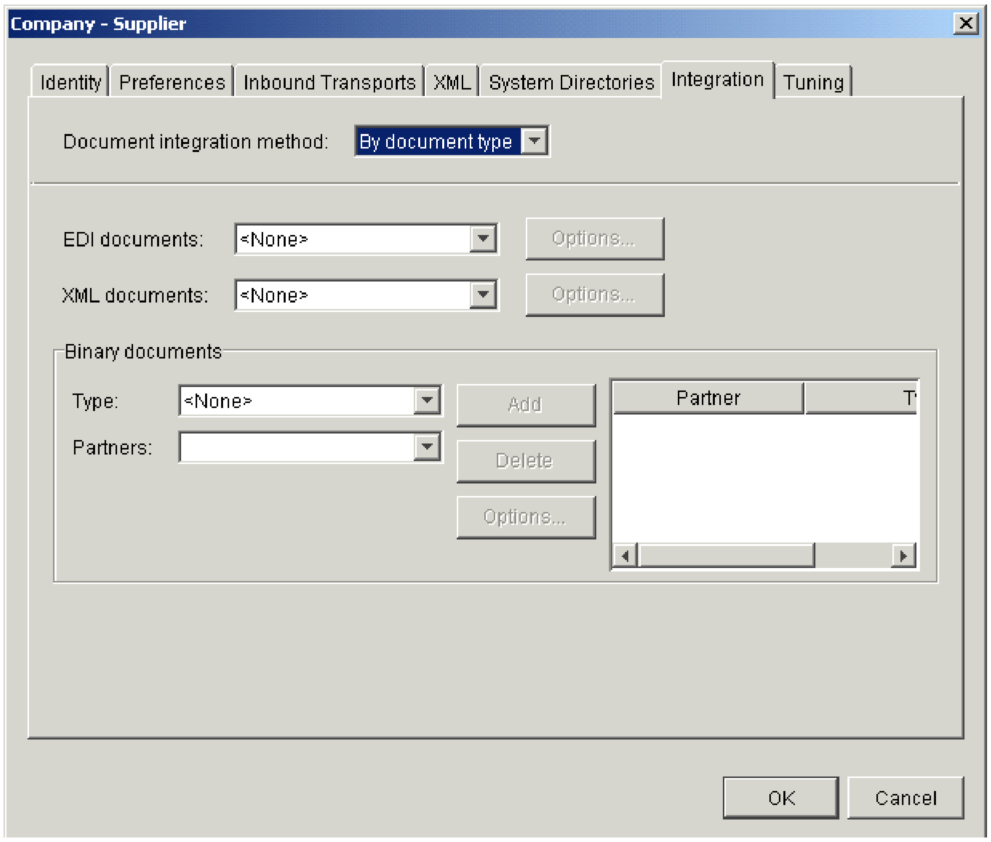
Figure 3-16 Server Display window at Retailer1

3.2.5 MQ integration and validation

Besides file-based integration, TPI also offers integration through JMS. For an example of JMS integration, refer to REDP3600, *WebSphere Data Interchange Installation and Configuration*.

Depending on your licence, TPI might also support other types of integration to the internal applications of a company. In this section, we are going to set up an integration using WebSphere MQ (MQSeries).

Start the TPI Administrator program and double-click the company profile **Supplier** which we created earlier. Select the tab **Integration**. In the drop-down box labeled Document integration method, select **By document type**. The Administrator will then show more options below this drop-down box, as shown in the next figure.


Company - Supplier

Document integration method: By document type

EDI documents: <None>

XML documents: <None>

Binary documents

Type: <None>

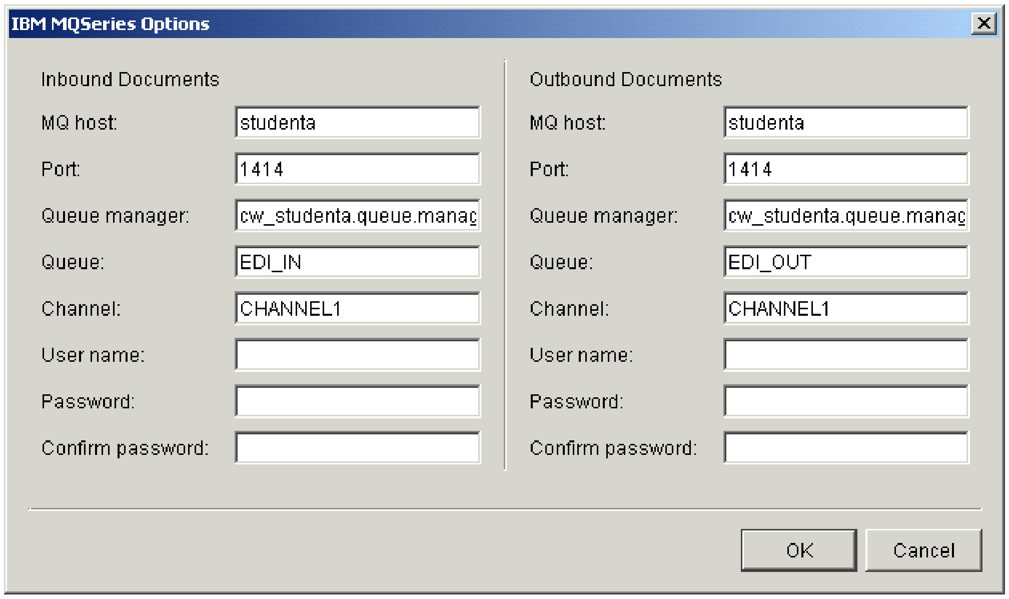
Partners:

Figure 3-17   Setting integration options

Now select the option **IBM MQSeries** from the drop-down menu labeled EDI documents. To configure the interaction with WebSphere MQ or MQSeries, click the button **Options**. TPI uses the MQ client interface to interact with a queue manager. You can set up different parameters (that is, a different queue manager) for inbound and outbound communication.

Provide the hostname of the machine that hosts the queue manager. Provide also the port that is used by the MQ listener task. You can find this port number by using the WebSphere MQ Services application.

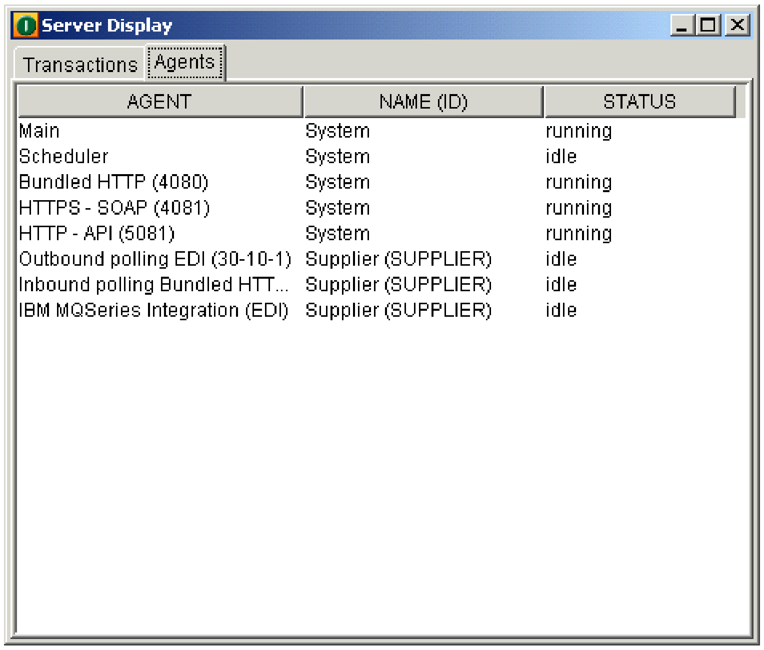
You should also provide the name of the queue manager and the names of the queues that are used to integrate with your internal systems. The queues EDI_IN and EDI_OUT are for example standard queues that are defined as part of the installation of WebSphere Data Interchange. Finally, since TPI is using the MQ client interface, you need to provide the name of an MQ channel of type Server Connection. Use the application WebSphere MQ Explorer to find the name of an existing channel or to define a new one.



Inbound Documents		Outbound Documents	
MQ host:	studenta	MQ host:	studenta
Port:	1414	Port:	1414
Queue manager:	cw_studenta.queue.manag	Queue manager:	cw_studenta.queue.manag
Queue:	EDI_IN	Queue:	EDI_OUT
Channel:	CHANNEL1	Channel:	CHANNEL1
User name:		User name:	
Password:		Password:	
Confirm password:		Confirm password:	

Figure 3-18 Settings integration options for WebSphere MQ (MQSeries)

Click **OK** to close the window to configure MQ options. Click **OK** again to close the company profile. Verify in the Server Log that the server has detected that the profile was updated. Now switch to the Server Display window and select the tab **Agents**. It should now list an additional agent for integration through WebSphere MQ (MQSeries).



AGENT	NAME (ID)	STATUS
Main	System	running
Scheduler	System	idle
Bundled HTTP (4080)	System	running
HTTPS - SOAP (4081)	System	running
HTTP - API (5081)	System	running
Outbound polling EDI (30-10-1)	Supplier (SUPPLIER)	idle
Inbound polling Bundled HTT...	Supplier (SUPPLIER)	idle
IBM MQSeries Integration (EDI)	Supplier (SUPPLIER)	idle

Figure 3-19 Additional agent for integration with WebSphere MQ

Note again that this new task is idle. This type of integration is once more a polling type of integration. By contrast, integration using JMS is immediate, that is, when a message arrives on a queue and TPI is listening on this queue via the JMS interface, the message will be picked up immediately. When TPI uses the standard MQ interface, it will open and read the queue periodically.

A quick way to verify that TPI is polling the queue EDI_OUT is to write a dummy message to this queue. You can for example use the Put Test Message option within WebSphere MQ Explorer for the queue EDI_OUT and send a “hello world” message. After a while, TPI should read this message from the queue and reject it as an invalid message. This proves that TPI is reading the queue.

An alternative way of testing is again to use the Document Generator tool. However, this time, do not save the generated document in the standard directory ediout, to prevent the server from sending the message before you can get the generated file. Use a tool like rfhutil to read the file and write it to the queue EDI_OUT. The tool rfhutil can be downloaded for free as a SupportPac™ from the following Web site:

`http://www-3.ibm.com/software/integration/support/supportpacs/individual/ih03.html`

This completes the basic setup for TPI for two partners and an integration using WebSphere MQ.

3.3 Communicating with an iSoft P2PAgent installation

This section discusses the implementation of iSoft’s P2PAgent program for the company Retailer2 and how to configure the P2PAgent and TPI to allow both companies to exchange EDI documents using a different B2B gateway product.

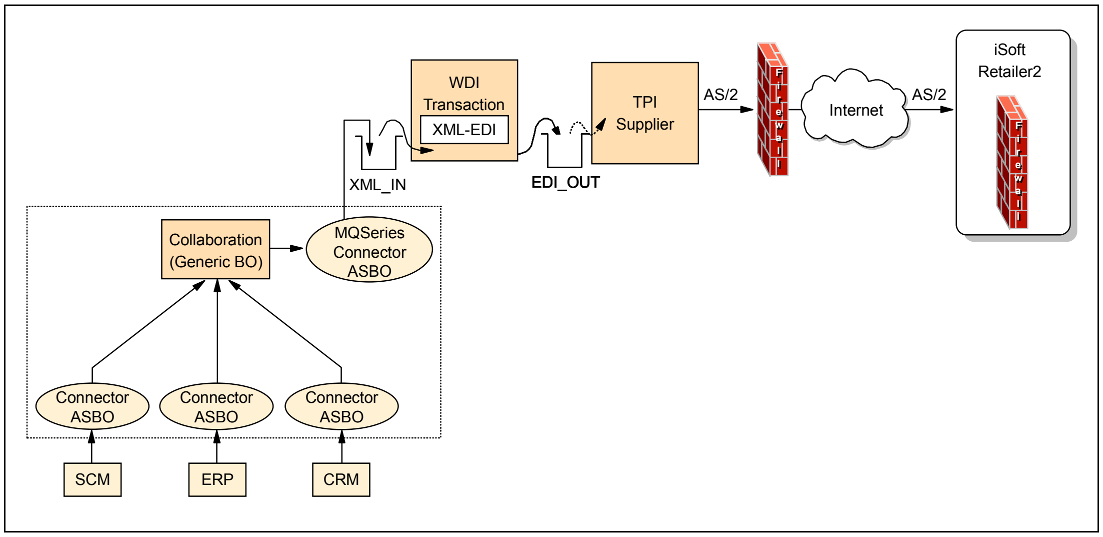
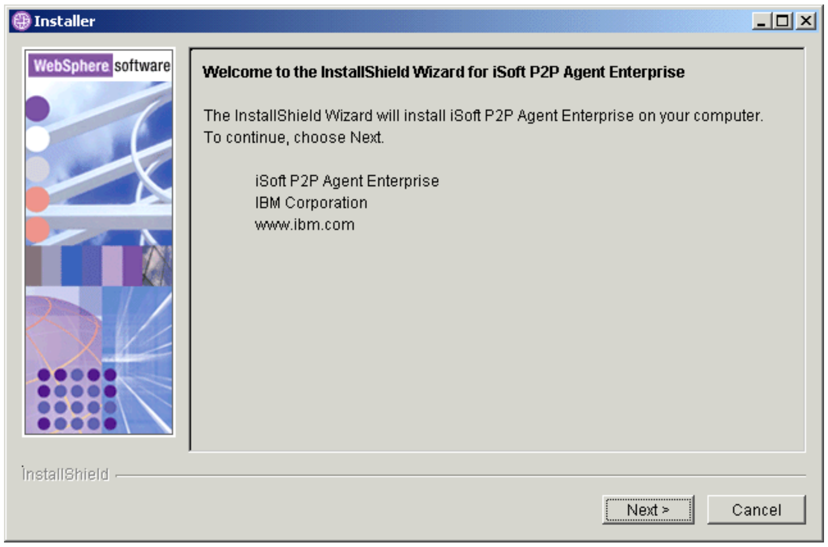


Figure 3-20 Overview of the components

3.3.1 Installation and initial configuration of iSoft’s P2PAgent

The AS2 client software is available in a few different packages from IBM or iSoft itself and with different licensing restrictions. In some packages, you may or may not find sample files and documentation. However, in all cases, the installation of the software is straightforward. Some of the steps below may not be required since they were performed during the installation itself.



Welcome to the InstallShield Wizard for iSoft P2P Agent Enterprise

The InstallShield Wizard will install iSoft P2P Agent Enterprise on your computer.
To continue, choose Next.

iSoft P2P Agent Enterprise
IBM Corporation
www.ibm.com

Figure 3-21   Installer for iSoft P2P Agent Enterprise edition

The P2PAgent program can use queues and/or files for several functions. To set up an initial environment, you should create a number of queues and/or directories to support those functions.

Example 3-1 lists a number of WebSphere MQ commands that you can use to create queues for use by the P2PAgent program. These queues can be used in the configuration file as described below.

Example 3-1   WebSphere MQ commands to create supporting queues

```
def ql('notices') defpsist(yes)
def ql('receipts’) defpsist(yes)
def ql('workorders') defpsist(yes)
def ql('outbox') defpsist(yes)
def ql('inbox') defpsist(yes)
def ql('log') defpsist(yes)
def ql('errors') defpsist(yes)
```

If you want a file-based integration for some or all of the iSoft functions, you should create the following directories. Again, these directories are used in the configuration file described below.

- inbox
- outbox
- receipts
- workorders
- pki
- errors
- log
- notices

The P2PAgent’s operations are controlled by a configuration file. The default location of this file is the install directory and the default name of this file is p2pagent.cfg. Note that it is possible to have a different name and location. You would then need to use start-up parameters to provide this information. Throughout this redbook, we will use the default name and location. The configuration file has an XML format and is basically a sequence of commands that are interpreted by the P2PAgent program at start-up time. Note again that these commands are the same as the interactive commands or the work order commands.

For all settings related to file locations and directories, you can replace the name of a directory with an MQ URI, in the format `mq://queue_manager_name/queue_name`, unless it is specified that only directory names are supported.

- Define the location to store error information

  To store error information in files in a directory named errors, use the following command:

  ```
  set -eperrors -ef
  ```

- Define the location to store logging information

  To store logging information in files in a directory named log, use the following command:

  ```
  set -lplog -lf
  ```

  The name of the log file is P2PYYYYMMDD.log, where YYYYMMDD is the current date. This implies that the P2PAgent program will automatically create a new file for each day and append information to it. The contents of this log file are basically the same as the standard output of the agent program.

- Define the location to store notices

  Notices are files or messages that indicate the results of document transactions.

  ```
  set -npmq://cw_studentc.queue.manager/notices -nf
  ```

- Define the location of certificates and private keys

  The PKI service component of the P2PAgent component will look in this directory to find any keys and certificates. The command below identifies this directory as pki.

  ```
  set -pppki
  ```

- Define the location to store receipts

  Receipts are the documents received by the sending agent when the receiving agent confirms the delivery of a given EDI document. In most case, such a receipt document will be signed using a digital signature algorithm. Receipts are delivered asynchronously.

  ```
  set -rpmq://cw_studentc.queue.manager/receipts
  ```

- Set time-out values

  The P2PAgent allows you to configure several types of time-outs. The command below sets the time-out for a first receive after accepting an inbound connection. For more information about other types of time-out values, refer to the iSoft product documentation.

  ```
  set -tr300s
  ```

- Set mailbox information

  Mailboxes can be file-based or queue-based or even stored in a database. The command below identifies a queue manager as the host of the mailboxes. Note that the location of a mailbox can still be set at the individual trading partner level.

  ```
  set -bhmq://cw_studentc.queue.manager
  ```

- Start the inbound service

  This command starts the inbound service, to listen on port 4080 for host name studentc.

  ```
  start http://studentc:4080
  ```

There are more options and settings that can be controlled in the configuration file, but for an initial deployment, these values are sufficient. In addition to a number of **`set`** commands, a configuration file typically contains a number of **`addpair`** commands and **`importkey`** commands. The **addpair** command defines a relationship between two symbolic trading partner names. The structure of the **addpair** command is as follows:

```
addpair <from> <to> <to-URL> <rcpt-URL> <notify-name> <mailbox>
```

Applied to this first occurrence of the **addpair** command in Example 3-2, this means:

When processing documents sent by partner SUPPLIER to partner RETAILER2, send this to the URL `http://studenta:4080/exchange/SUPPLIER`. Request that receipts be sent to the URL `http://studentc:4080` and address those receipts to partner RETAILER2.

The second occurrence of the **addpair** command in Example 3-2 applies to the processing of incoming documents. Here, the `rcpt-URL` parameter is set to `*` to indicate that we do not override the settings requested by the sender of the document.

The **importkey** command assigns certificates and/or private keys to a trading partner relationship for a specific function, such as encryption and signing. While many possibilities exist, a typical scenario is expressed in Example 3-2. The syntax of the **importkey** command is shown below:

```
importkey <from> <to> <usage> <options>
```

`From` and `To` identify the trading partner relationship. The `usage` is a one character code that identifies when the certificate and/or private key should be used. The `options` are used to identify the key and certificate.

The first command instructs the P2PAgent program that documents from RETAILER2 to SUPPLIER are to be signed and decrypted (option E) using the private key and certificate of the partner RETAILER2. The signing here relates to documents sent to RETAILER2. The decrypting relates to the decryption of MDNs received from RETAILER2 as a result of sending documents to RETAILER2.

The encryption for documents sent from RETAILER2 to SUPPLIER (option J) is done using the certificate of partner SUPPLIER, as shown in the second **importkey** command. In practice, this means that:

- RETAILER2 sends documents to SUPPLIER in such a way that only SUPPLIER can read them, since we can assume that only SUPPLIER has access to its own private key.
- RETAILER2 sends documents to SUPPLIER in such a way that RETAILER2 can never deny that it has sent that document. RETAILER2 will sign the document using its private key.

The reverse commands are needed too to control how documents should be received by RETAILER2 when sent by SUPPLIER.

Example 3-2 P2P agent configuration file for one bi-directional communication link

```
<xml>
<command>set -eperrors -ef </command>
<command>set -lplog -lf </command>
<command>set -npmq://cw_studentc.queue.manager/notices    -nf </command>
<command>set -opmq://cw_studentc.queue.manager/workorders -of -oswo </command>
<command>set -pppki </command>
<command>set -rpmq://cw_studentc.queue.manager/receipts </command>
<command>set -tr300s </command>
<command>set -bhmq://cw_studentc.queue.manager </command>

<command>addpair   RETAILER2 SUPPLIER http://studenta:4080/exchange/SUPPLIER/       http://studentc:4080/
RETAILER2 mq://cw_studentc.queue.manager/outbox</command>
```

```
<command>addpair   SUPPLIER RETAILER2 http://studentc:4080/      *                    SUPPLIER
mq://cw_studentc.queue.manager/inbox</command>
<command>importkey SUPPLIER RETAILER2 E -fCpki\RETAILER2-SUPPLIER.cer
-fKpki\RETAILER2-SUPPLIER.prv</command>
<command>importkey SUPPLIER RETAILER2 J -fCpki\SUPPLIER-RETAILER2.cer </command>
<command>importkey RETAILER2 SUPPLIER E -fCpki\RETAILER2-SUPPLIER.cer
-fKpki\RETAILER2-SUPPLIER.prv</command>
<command>importkey RETAILER2 SUPPLIER J -fCpki\SUPPLIER-RETAILER2.cer </command>

<command>start http://studentc:4080</command>
</xml>
```

This completes the first step of the configuration and you can now start up the agent. Open a command window and switch to the directory that holds the configuration file (shown in Example 3-2 on page 119) and the program executable. Start the program and you should see output similar to the output shown in Example 3-3.

Example 3-3 Standard output of first start-up of the P2PAgent program

```
C:\iSoft_Enterprise>p2pagent_odbc_ibm_enterprise.exe
iSoft(R) Peer-to-Peer Agent(TM) for MQSeries(R)
(C) Copyright 2001-2002 iSoft Corp.
Build: 3.1.2002.10.30.1 [Nov 27 2002 15:08:00]
IBM Enterprise Edition
Authorized License
2003.01.27 13:25:43.299      POPT OK  Error path set to [errors]
2003.01.27 13:25:43.299      POPT OK  Inbound errant will be stored
2003.01.27 13:25:43.309      POPT OK  Log path set to [log]
2003.01.27 13:25:43.309      POPT OK  Trace set to WRITE_FILE
2003.01.27 13:25:43.329      POPT OK  Notice path set to [mq://cw_studentc.queue.manager/notices]
2003.01.27 13:25:43.329      POPT OK  Notices will be written to file
2003.01.27 13:25:43.349      POPT OK  Work-order path set to [mq://cw_studentc.queue.manager/workorders]
2003.01.27 13:25:43.349      POPT OK  Work-order searching enabled
2003.01.27 13:25:43.349      POPT OK  Work-order file-spec set to [wo]
2003.01.27 13:25:43.359      POPT OK  PKI path set to [pki]
2003.01.27 13:25:43.379      POPT OK  Async. receipt path set to [mq://cw_studentc.queue.manager/receipts]
2003.01.27 13:25:43.389      POPT OK  First-receive interval set to [300000ms]
2003.01.27 13:25:43.399      POPT OK  Mailbox host set to [mq://cw_studentc.queue.manager]
2003.01.27 13:25:43.399      POPT OK  Mailbox address set to [0.0.0.0]
2003.01.27 13:25:43.409      POPT OK  Mailbox port set to [0]
2003.01.27 13:25:43.479      HPIM OK  HTTP inbound service started
2003.01.27 13:25:48.506      PIKC ERR Unable to import keys
2003.01.27 13:25:48.536      PIKC ERR Unable to import keys
2003.01.27 13:25:48.556      PIKC ERR Unable to import keys
2003.01.27 13:25:48.576      PIKC ERR Unable to import keys
```

Since we have not yet generated any keys nor received any certificates from our trading partner SUPPLIER, it is not surprising that the four **importkey** commands are failing. To generate these keys, we need to run an **addkey** command. However, since the **addkey** command only needs to be run once, it is not included in the configuration file. To generate keys, create a work order file (for example addkeys.wo) that contains an XML document, such as the one shown in Example 3-4 on page 121.

The structure of the **addkey** command is as follows:

```
addkey <from> <to> <function> <key length> <issuer> <subject>
```

Applied to the first occurrence of the **addkey** command, we ask to generate a key that is used for communication from RETAILER2 to SUPPLIER for the functions sign, encrypt, decrypt and signature verification (option O) with a key length of 1024 bits for RSA. The certificate is self-signed and has the provided subject.

Example 3-4 Addkeys.wo work order file

```
<xml>

# Create Keys

<command>addkey RETAILER2 SUPPLIER O 512 self C=US;S=TX;L=Dallas;O=iSoft;CN=RETAILER2</command>

</xml>
```

To execute the commands of the work order, you need to execute the **batch** command in an interactive session of the P2PAgent program. Type `batch addkeys.wo` and press the **Enter** key in the command window (see Example 3-5).

Example 3-5 Output of addkeys.wo work order

```
C:\iSoft_Enterprise>p2pagent_odbc_ibm_enterprise.exe
iSoft(R) Peer-to-Peer Agent(TM) for MQSeries(R)
(C) Copyright 2001-2002 iSoft Corp.
Build: 3.1.2002.10.30.1 [Nov 27 2002 15:08:00]
IBM Enterprise Edition
Authorized License
2003.01.27 13:26:52.448       POPT OK  Error path set to [error]
2003.01.27 13:26:52.448       POPT OK  Inbound errant will be stored
2003.01.27 13:26:52.458       POPT OK  Log path set to [log]
2003.01.27 13:26:52.458       POPT OK  Trace set to WRITE_FILE
2003.01.27 13:26:52.478       POPT OK  Notice path set to [mq://cw_studentc.queue.manager/notices]
2003.01.27 13:26:52.478       POPT OK  Notices will be written to file
2003.01.27 13:26:52.498       POPT OK  Work-order path set to [mq://cw_studentc.queue.manager/workorders]
2003.01.27 13:26:52.498       POPT OK  Work-order searching enabled
2003.01.27 13:26:52.508       POPT OK  Work-order file-spec set to [wo]
2003.01.27 13:26:52.518       POPT OK  PKI path set to [pki]
2003.01.27 13:26:52.528       POPT OK  Async. receipt path set to [mq://cw_studentc.queue.manager/receipts]
2003.01.27 13:26:52.538       POPT OK  First-receive interval set to [300000ms]
2003.01.27 13:26:52.558       POPT OK  Mailbox host set to [mq://cw_studentc.queue.manager]
2003.01.27 13:26:52.558       POPT OK  Mailbox address set to [0.0.0.0]
2003.01.27 13:26:52.569       POPT OK  Mailbox port set to [0]
2003.01.27 13:26:52.649       PIKC ERR Unable to import keys
2003.01.27 13:26:52.649       HPIM OK  HTTP inbound service started
2003.01.27 13:26:52.669       PIKC ERR Unable to import keys
2003.01.27 13:26:52.689       PIKC ERR Unable to import keys
2003.01.27 13:26:52.719       PIKC ERR Unable to import keys
batch addkeys.wo
ok
2003.01.27 13:27:18.125       PAKC OK  Keypair generated
```

Since we need to share the certificate with our trading partner SUPPLIER, we need to export the key in a file. Here, we again use the concept of a work order to perform these actions. Example 3-6 on page 122 shows the Exportkeys.wo work order file for all three trading partners.

Example 3-6 Exportkeys.wo work order file

```
<xml>

# Export Keys

<command>exportkey RETAILER2 SUPPLIER 0 RETAILER2-SUPPLIER.cer RETAILER2-SUPPLIER.prv</command>

</xml>
```

To run these commands in an interactive session, we again use the **batch** command, as shown in Example 3-7.

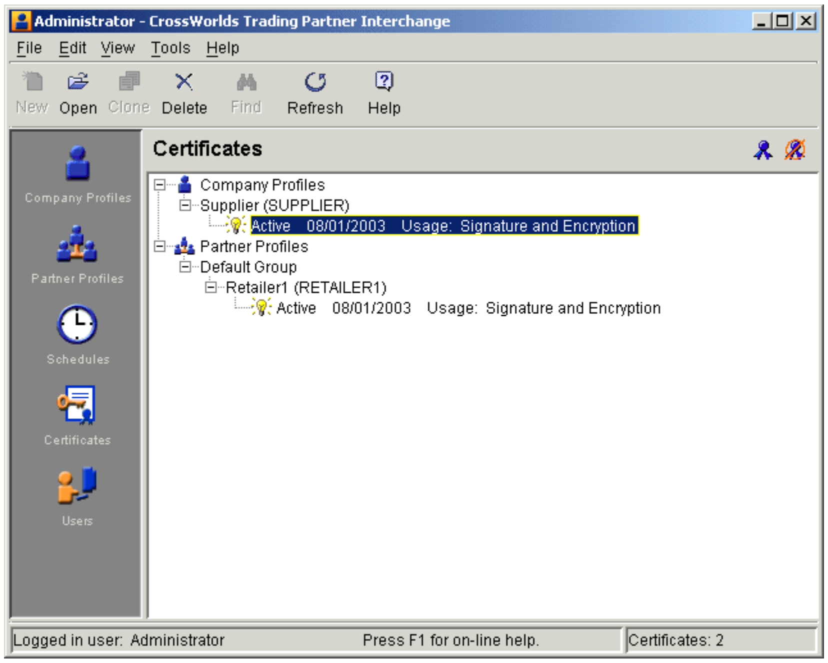
Example 3-7 Output of the exportkeys.wo work order

```
C:\iSoft_Enterprise>p2pagent_odbc_ibm_enterprise.exe
iSoft(R) Peer-to-Peer Agent(TM) for MQSeries(R)
(C) Copyright 2001-2002 iSoft Corp.
Build: 3.1.2002.10.30.1 [Nov 27 2002 15:08:00]
IBM Enterprise Edition
Authorized License
2003.01.27 13:26:52.448       POPT OK  Error path set to [error]
2003.01.27 13:26:52.448       POPT OK  Inbound errant will be stored
2003.01.27 13:26:52.458       POPT OK  Log path set to [log]
2003.01.27 13:26:52.458       POPT OK  Trace set to WRITE_FILE
2003.01.27 13:26:52.478       POPT OK  Notice path set to [mq://cw_studentc.queue.manager/notices]
2003.01.27 13:26:52.478       POPT OK  Notices will be written to file
2003.01.27 13:26:52.498       POPT OK  Work-order path set to [mq://cw_studentc.queue.manager/workorders]
2003.01.27 13:26:52.498       POPT OK  Work-order searching enabled
2003.01.27 13:26:52.508       POPT OK  Work-order file-spec set to [wo]
2003.01.27 13:26:52.518       POPT OK  PKI path set to [pki]
2003.01.27 13:26:52.528       POPT OK  Async. receipt path set to [mq://cw_studentc.queue.manager/receipts]
2003.01.27 13:26:52.538       POPT OK  First-receive interval set to [300000ms]
2003.01.27 13:26:52.558       POPT OK  Mailbox host set to [mq://cw_studentc.queue.manager]
2003.01.27 13:26:52.558       POPT OK  Mailbox address set to [0.0.0.0]
2003.01.27 13:26:52.569       POPT OK  Mailbox port set to [0]
2003.01.27 13:26:52.649       PIKC ERR Unable to import keys
2003.01.27 13:26:52.649       HPIM OK  HTTP inbound service started
2003.01.27 13:26:52.669       PIKC ERR Unable to import keys
2003.01.27 13:26:52.689       PIKC ERR Unable to import keys
2003.01.27 13:26:52.719       PIKC ERR Unable to import keys
batch addkeys.wo
ok
2003.01.27 13:27:18.125       PAKC OK  Keypair generated
batch exportkeys.wo
ok
2003.01.27 13:27:45.645       POKC OK  Key-pair exported
```

At this time, stop the P2PAgent program using the **shutdown** command. Before we can restart the P2PAgent, we need to exchange certificates.

3.3.2 Exporting the certificate from TPI

Start the TPI Administrator tool on the machine that hosts the exchange for SUPPLIER and select the view **Certificates**. As shown in Figure 3-22 on page 123, the view Certificates contains a company certificate and a partner certificate for RETAILER1. Select the company certificate and select **File -> Export**.


Administrator - CrossWorlds Trading Partner Interchange
File Edit View Tools Help
New Open Clone Delete Find Refresh Help
Company Profiles
Partner Profiles
Schedules
Certificates
Users

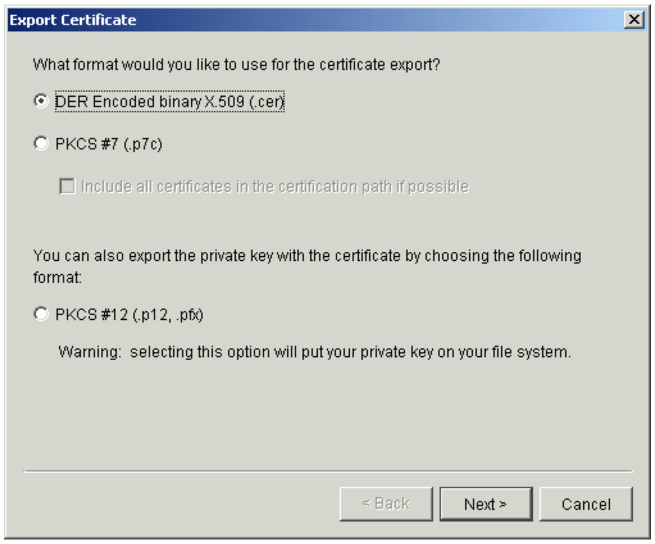
Certificates

- Company Profiles
  - Supplier (SUPPLIER)
    - Active 08/01/2003 Usage: Signature and Encryption
- Partner Profiles
  - Default Group
    - Retailer1 (RETAILER1)
      - Active 08/01/2003 Usage: Signature and Encryption

Logged in user: Administrator   Press F1 for on-line help.   Certificates: 2

Figure 3-22 The view Certificates of TPI Administrator

You can export a certificate in a few formats. Select the option **DER Encoded binary X.509** and click **Next**. Provide a file name and folder name in the next window and click **Next** again. A summary window will appear where you click **Finish**.



What format would you like to use for the certificate export?

[✓ DER Encoded binary X.509 (.cer)]

PKCS #7 (.p7c)

Include all certificates in the certification path if possible

You can also export the private key with the certificate by choosing the following format:

PKCS #12 (.p12, .pfx)

Warning: selecting this option will put your private key on your file system.

Figure 3-23 Export certificate window

On the machine that hosts the iSoft P2PAgent for RETAILER2, copy the certificate in the folder pki within the iSoft installation folder. Make sure that the file name matches the **`importkey`** statements. You can then restart the P2PAgent. This time, the start-up should no longer generate errors, as shown in Example 3-8 on page 124.

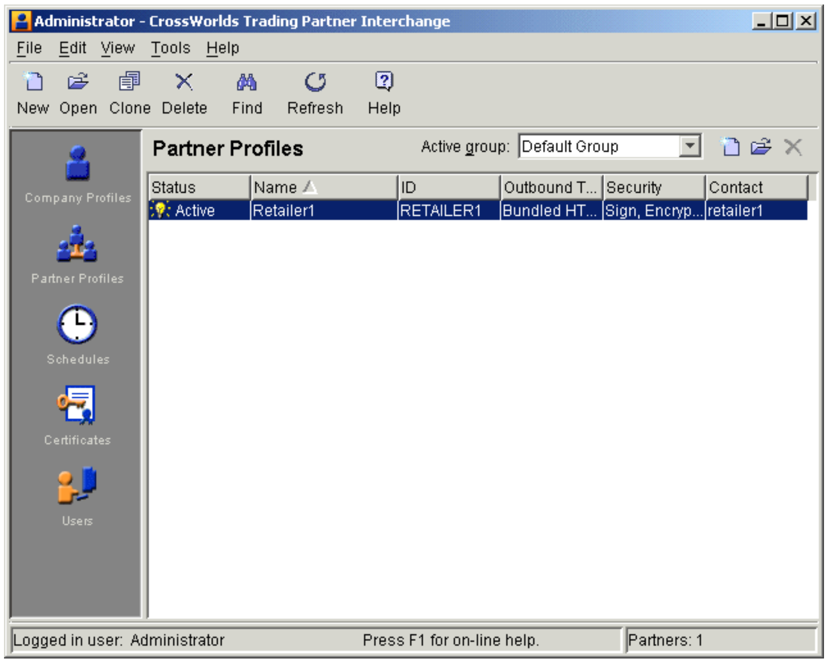
Example 3-8 Standard output of restarted P2PAgent program

```
C:\iSoft_Enterprise>p2pagent_odbc_ibm_enterprise.exe
iSoft(R) Peer-to-Peer Agent(TM) for MQSeries(R)
(C) Copyright 2001-2002 iSoft Corp.
Build: 3.1.2002.10.30.1 [Nov 27 2002 15:08:00]
IBM Enterprise Edition
Authorized License
2003.01.27 13:32:54.209       POPT OK  Error path set to [errors]
2003.01.27 13:32:54.209       POPT OK  Inbound errant will be stored
2003.01.27 13:32:54.219       POPT OK  Log path set to [log]
2003.01.27 13:32:54.229       POPT OK  Trace set to WRITE_FILE
2003.01.27 13:32:54.239       POPT OK  Notice path set to [mq://cw_studenta.queue.manager/notices]
2003.01.27 13:32:54.239       POPT OK  Notices will be written to file
2003.01.27 13:32:54.249       POPT OK  Work-order path set to [mq://cw_studenta.queue.manager/workorders]
2003.01.27 13:32:54.259       POPT OK  Work-order searching enabled
2003.01.27 13:32:54.259       POPT OK  Work-order file-spec set to [wo]
2003.01.27 13:32:54.269       POPT OK  PKI path set to [pki]
2003.01.27 13:32:54.289       POPT OK  Async. receipt path set to [mq://cw_studenta.queue.manager/receipts]
2003.01.27 13:32:54.299       POPT OK  First-receive interval set to [300000ms]
2003.01.27 13:32:54.309       POPT OK  Mailbox host set to [mq://cw_studenta.queue.manager]
2003.01.27 13:32:54.309       POPT OK  Mailbox address set to [0.0.0.0]
2003.01.27 13:32:54.319       POPT OK  Mailbox port set to [0]
2003.01.27 13:32:54.399       HPIM OK  HTTP inbound service started
```

This completes the setup for Retailer2. Before we can exchange documents, we need to create a partner profile for Retailer2 in the TPI of Supplier.

3.3.3 Creating a partner profile for Retailer2 in TPI of Supplier

On the machine that hosts the TPI exchange of Supplier, start the TPI Administrator and select the view **Partner Profiles**. As shown in Figure 3-24 on page 125, the view contains the profile of Retailer1, which was created by importing the profile. For Retailer2, we cannot use the import function, since Retailer2 uses iSoft's P2PAgent, which cannot generate an XML profile suitable for TPI.


Administrator - CrossWorlds Trading Partner Interchange
File Edit View Tools Help
New Open Clone Delete Find Refresh Help
Company Profiles
Partner Profiles
Schedules
Certificates
Users

Partner Profiles

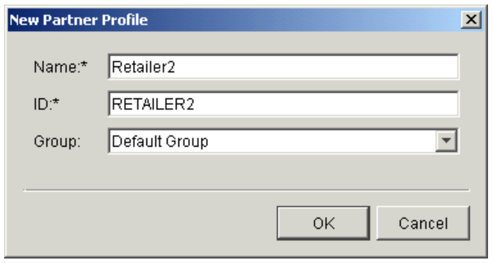
Active group: Default Group

Status	Name	ID	Outbound T...	Security	Contact
Active	Retailer1	RETAILER1	Bundled HT...	Sign, Encryp...	retailer1

Logged in user: Administrator Press F1 for on-line help. Partners: 1

Figure 3-24 Partner Profiles view of TPI

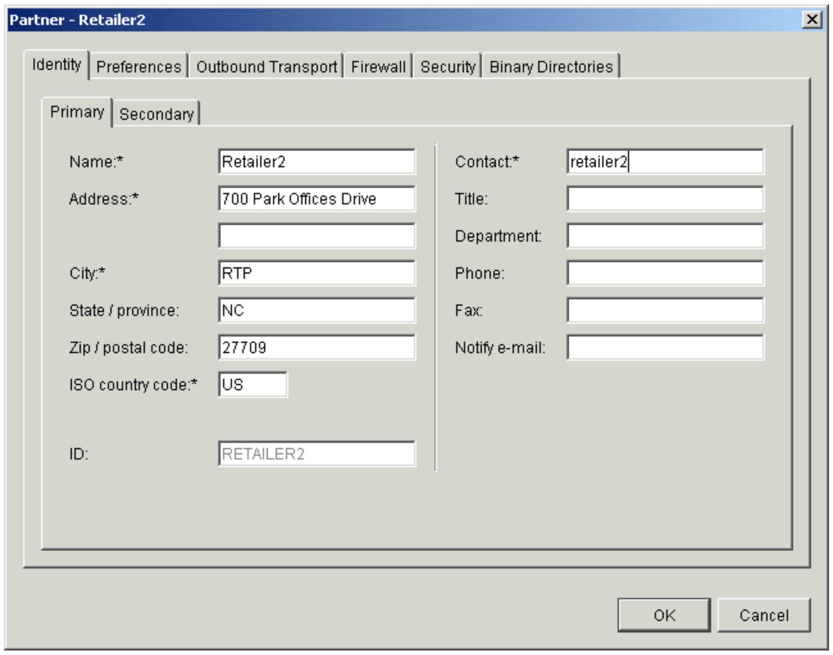
To define the profile for Retailer2 manually, select **File -> New**. The New Partner Profile window (Figure 3-25). Provide a name for this profile and specify the ID, as it will appear in the ISA segment of an EDI document for and from Retailer2. Click **OK**.



Name:*	Retailer2
ID:*	RETAILER2
Group:	Default Group

Figure 3-25 New partner profile: set name and ID

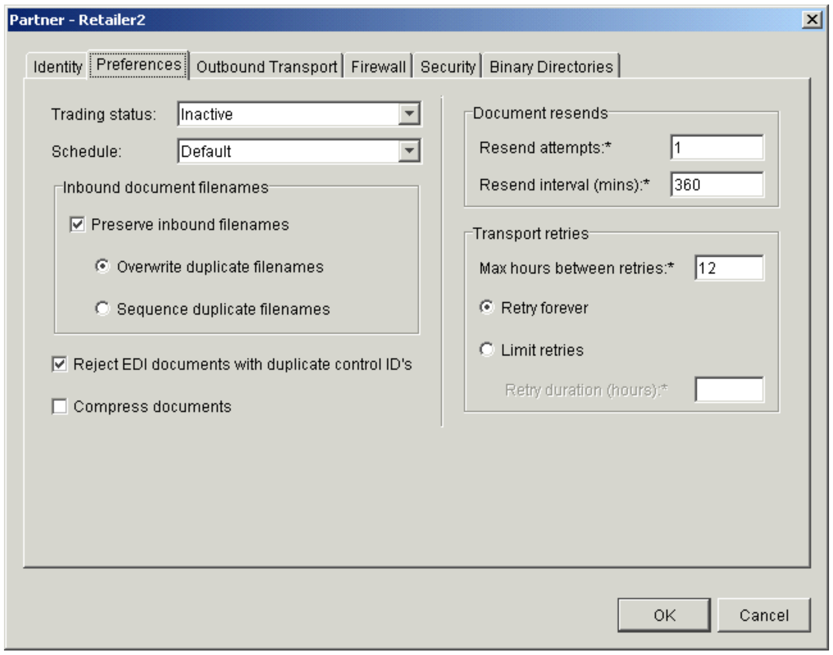
After clicking **OK**, a new window will appear that will allow you to specify different kinds of information about Retailer2 and how to communicate with it. On the Identity tab, provide address and contact information.



Name:*	Retailer2	Contact:*	retailer2
Address:*	700 Park Offices Drive	Title:	
		Department:	
City:*	RTP	Phone:	
State / province:	NC	Fax:	
Zip / postal code:	27709	Notify e-mail:	
ISO country code:*	US		
ID:	RETAILER2		

Figure 3-26 New partner profile: provide address and contact information

Select the tab **Preferences** and review the settings about retries and resends. The default values are fine for our purposes. The tab Preferences also controls whether a partner is active or not. Select **Inactive** in the list box Trading Status. We will change it to Active after importing the certificate of Retailer2. We cannot leave it active at this time, since we will require encryption and digital signatures. TPI makes sure that no partner profile is active for which there is no certificate and for which encryption and signatures are required.



Trading status: Inactive

Schedule: Default

Inbound document filenames

[✓] Preserve inbound filenames

(●) Overwrite duplicate filenames

( ) Sequence duplicate filenames

[✓] Reject EDI documents with duplicate control ID's

[ ] Compress documents

Document resends

Resend attempts:* 1

Resend interval (mins):* 360

Transport retries

Max hours between retries:* 12

(●) Retry forever

( ) Limit retries

Retry duration (hours):*

Figure 3-27 New partner profile: the tab Preferences

Now select the tab **Outbound Transports**. Click the **Add** button to set up a transport for Retailer2.

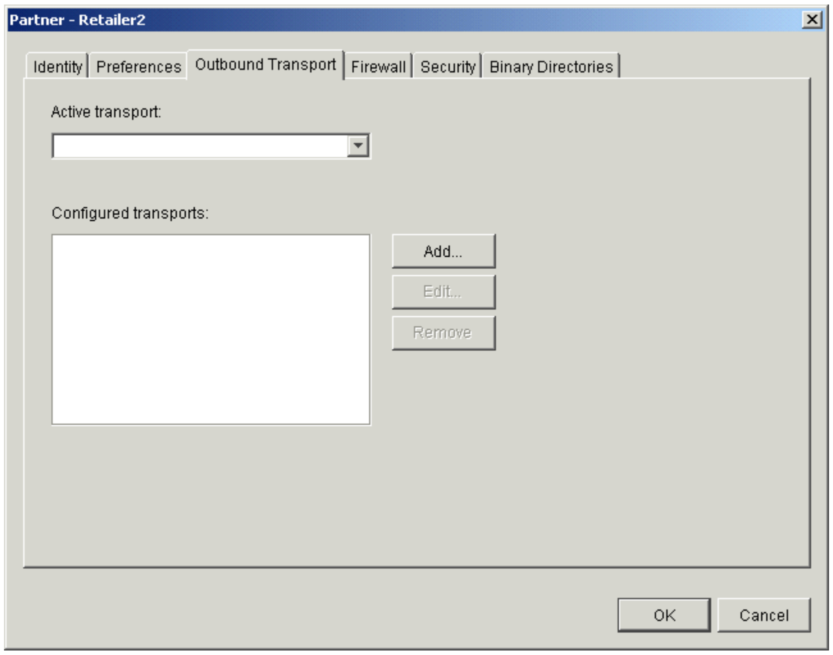
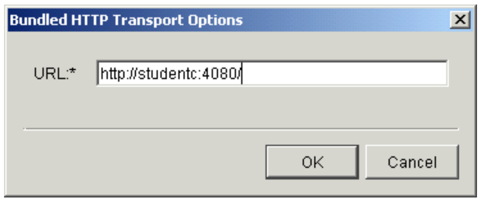


Figure 3-28 New partner profile: the tab Outbound Transport

In the Add Transport window, select **Bundled HTTP** and click **OK**.

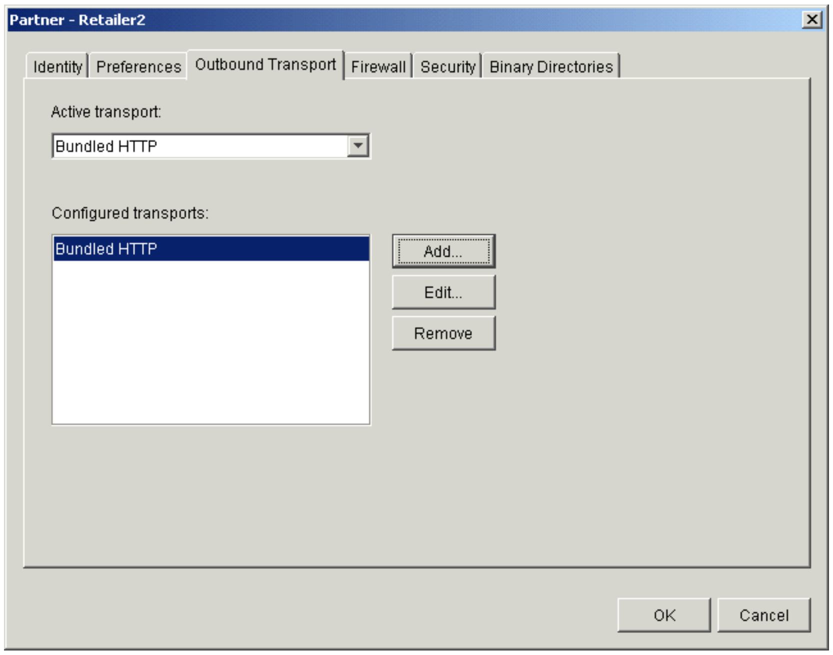
A new window will appear to provide transport options for Bundled HTTP. Provide the URL on which iSoft's P2PAgent is listening, which is `http://studentc:4080` in our case.



URL:* http://studentc:4080/

Figure 3-29 New partner profile: Bundled HTTP Transport Options

Click **OK** to close this window, which will bring you back to the main setup window for a partner profile.

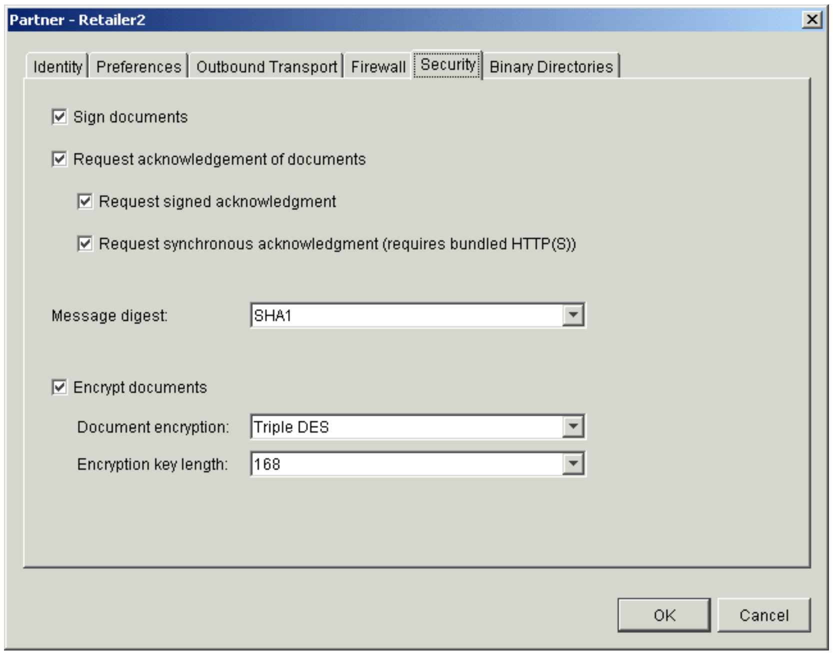

Partner - Retailer2

Active transport: Bundled HTTP

Configured transports: Bundled HTTP

Figure 3-30 New partner profile: completed outbound transport settings

If Supplier will be using a firewall, you need to provide details about the firewall on the tab Firewall. Select the tab **Security** to control the exchange of documents with Retailer2. Set the options as shown in Figure 3-31 to make sure that both documents and acknowledgments are signed and encrypted. Note that these settings will impact the **`send`** command in iSoft's P2PAgent as we see in 3.3.6, "Validation of the setup" on page 130.


Partner - Retailer2

☑ Sign documents

☑ Request acknowledgement of documents

☑ Request signed acknowledgment

☑ Request synchronous acknowledgment (requires bundled HTTP(S))

Message digest: SHA1

☑ Encrypt documents

Document encryption: Triple DES

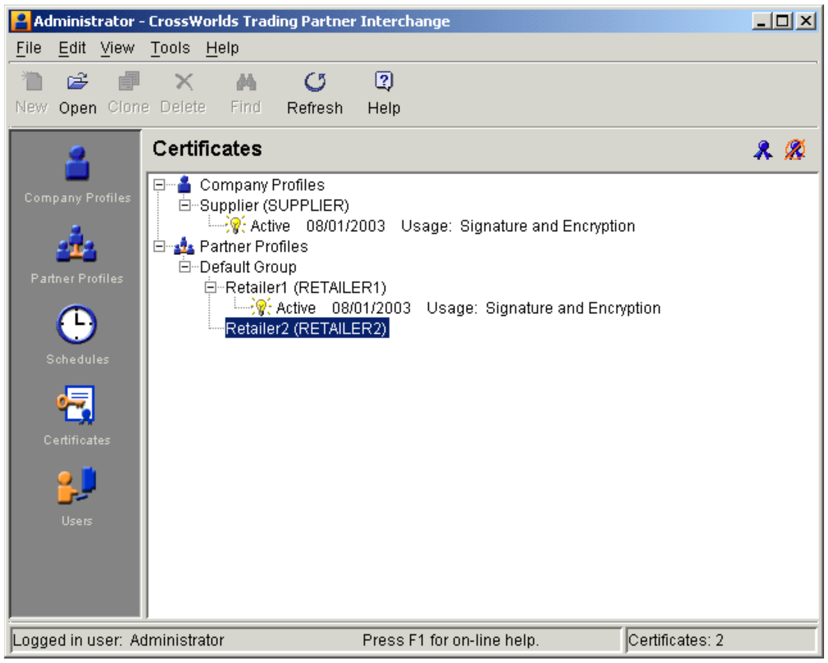
Encryption key length: 168

Figure 3-31 New partner profile: the Security tab

Since we do not exchange binary documents, we do not need to make any changes on the tab Binary Directories. Click **OK** to finish the partner profile definition.

3.3.4 Importing the certificate of Retailer2 in TPI

The next step is to import the certificate of Retailer2 in the database of TPI. We assume again that the trading partners have agreed upon a secure way to exchange the certificates. Start the TPI Administrator and select the view **Certificates**.


Administrator - CrossWorlds Trading Partner Interchange
File Edit View Tools Help
New Open Clone Delete Find Refresh Help
Company Profiles
Partner Profiles
Schedules
Certificates
Users

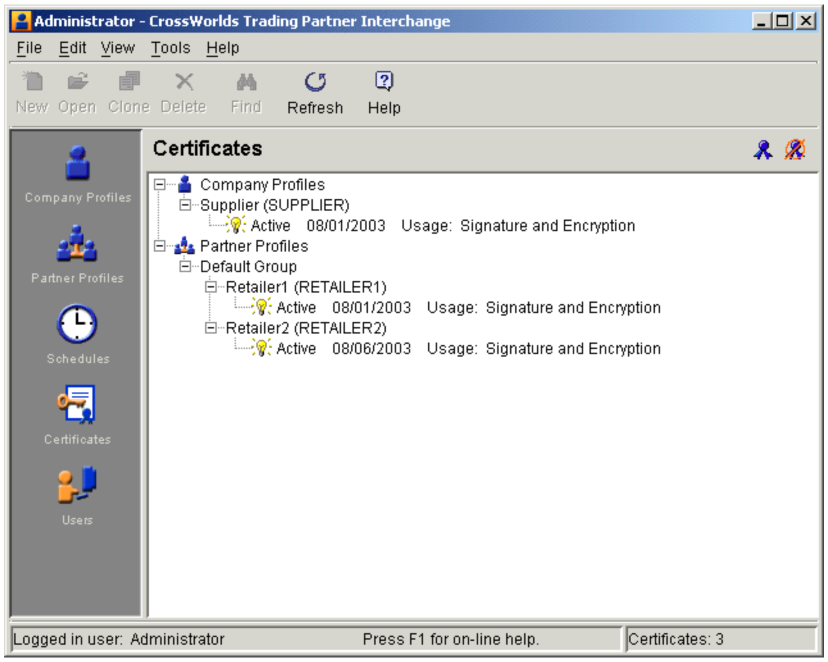
Certificates

```
Company Profiles
  Supplier (SUPPLIER)
    Active   08/01/2003   Usage:  Signature and Encryption
Partner Profiles
  Default Group
    Retailer1 (RETAILER1)
      Active   08/01/2003   Usage:  Signature and Encryption
    Retailer2 (RETAILER2)
```

Logged in user: Administrator   Press F1 for on-line help.   Certificates: 2

Figure 3-32 The view Certificates of TPI Administrator

Select the inactive partner **Retailer2** and click **File -> Import**. Provide the file name of the certificate file and select **Finish**. Figure 3-33 shows the Certificates view after the import.


Administrator - CrossWorlds Trading Partner Interchange
File Edit View Tools Help
New Open Clone Delete Find Refresh Help
Company Profiles
Partner Profiles
Schedules
Certificates
Users

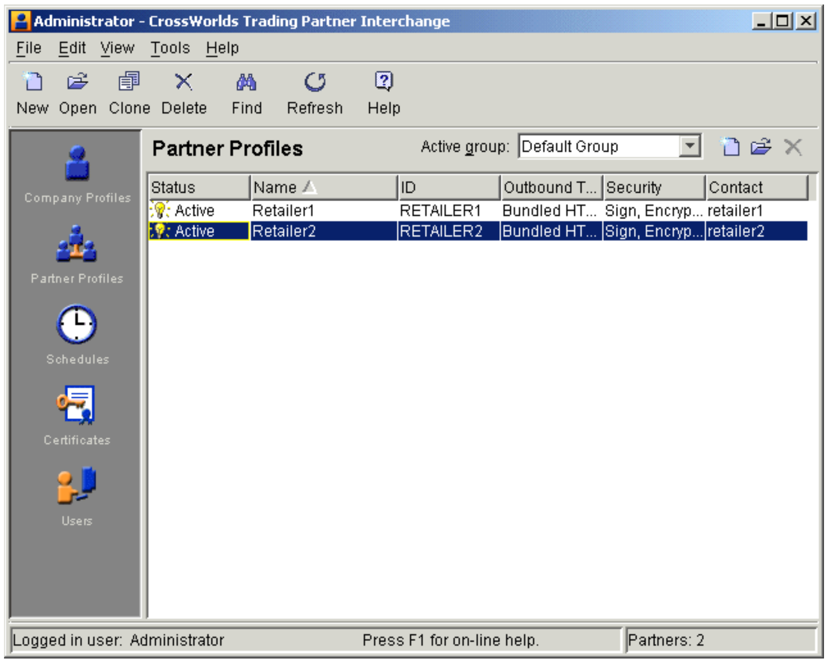
Certificates

```
Company Profiles
  Supplier (SUPPLIER)
    Active   08/01/2003   Usage:  Signature and Encryption
Partner Profiles
  Default Group
    Retailer1 (RETAILER1)
      Active   08/01/2003   Usage:  Signature and Encryption
    Retailer2 (RETAILER2)
      Active   08/06/2003   Usage:  Signature and Encryption
```

Logged in user: Administrator   Press F1 for on-line help.   Certificates: 3

Figure 3-33 The Certificates view of TPI Administrator

Switch now to the Partner Profiles view. Double-click the profile **Retailer2** and switch to the tab Preferences. Set the property Trading Status to `Active` and click **OK**.


Administrator - CrossWorlds Trading Partner Interchange
File Edit View Tools Help
New Open Clone Delete Find Refresh Help
Company Profiles
Partner Profiles
Schedules
Certificates
Users
Partner Profiles
Active group: Default Group

Status	Name	ID	Outbound T...	Security	Contact
Active	Retailer1	RETAILER1	Bundled HT...	Sign, Encryp...	retailer1
Active	Retailer2	RETAILER2	Bundled HT...	Sign, Encryp...	retailer2

Logged in user: Administrator   Press F1 for on-line help.   Partners: 2

Figure 3-34 Completed partner profile for Retailer2 in TPI

3.3.5 Upgrading TPI

TPI had some AS/2 interoperability problems that are solved by upgrading to V4.1.2.6 or later. If you use TPI V4.1, you can download a fix from the following Web site:

`http://www-1.ibm.com/support/docview.wss?uid=swg24001872`

To apply the fix, unzip the download package in a temporary file and copy the upgraded file cyclone.jar to the lib directory in the TPI installation directory.

3.3.6 Validation of the setup

Now that we are using TPI at the right level and that we have configured profiles on both sides, we can perform some validation by exchanging documents between the two partners.

Retailer2 sends documents

There are several ways within iSoft’s P2PAgent to send files. Some of those techniques are:

- Polling a directory for files with a given extension.
- Polling a queue for messages.
- Polling for workorders that contain a **`send`** command. Workorders can be stored in a directory and the extension of the file name should match a given extension, which is .wo by default.
- Sending a file as the result of a **`send`** command.

To simply send a test file for validation, we use the last technique. The workorder file sendcmd.wo is shown in Example 3-9 on page 131. The **`send`** command instructs the P2PAgent program to send files with extension .out (-fS parameter) from the directory outbox (-fP parameter) to the Supplier. When the transmission completes, the file should be renamed by adding the extension .pend (-fE parameter). The document should be encrypted using the

Triple Des algorithm (-e parameter) and encoded using the Base64 technique (-pB). The document will we signed using the SH-1 algorithm and the signature will be encoded using the Base64 technique (-sB parameter). The P2PAgent will request a synchronous MDN from the partner and the MDN needs to be signed using the SH-1 algorithm (-r1 parameter; an asynchronous MDN would be -r1A). The **send** command is performed only once (-n1 parameter). Therefore, no retries take place if it fails. The -cE parameter sets the MIME type to application/edi-x.12. Finally, the -hQ parameter instructs the P2PAgent program to quote the EDI names (RETAILER2 and SUPPLIER) in the MIME headers.

Example 3-9 The command file sendcmd.wo

```
<xml>
<command>
send http RETAILER2 SUPPLIER -fPoutbox -fSout -fE.pend -tE20032132000000 -e -pB -sB -r1 -n1 -cE -hQ
</command>
</xml>
```

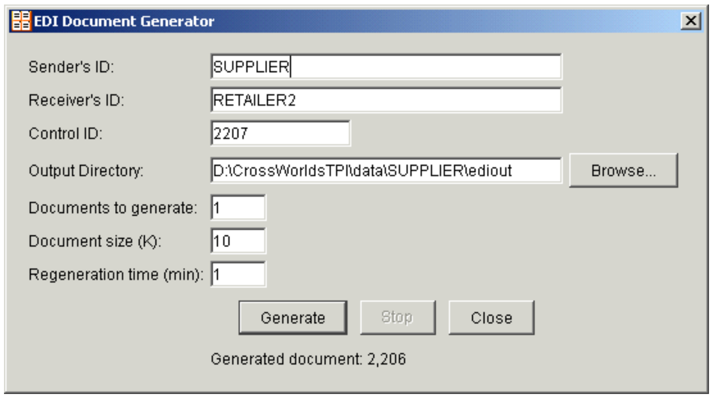
To launch the **send** command, you type the command `batch sendcmd.wo` in an interactive session of the P2PAgent, as shown in Example 3-10. The output of the interaction with TPI is the top part of what is shown in the example. The output in the TPI Server Display is shown in Figure 3-36 on page 132.

Example 3-10 Output of the P2PAgent for a send to TPI and a receive from TPI

```
batch sendcmd.wo
ok
2003.01.28 09:07:40.564 84044 HPOS OK  Outbound session started - mbox=[0] batch=[0] attempt=[1 of 1]
2003.01.28 09:07:48.335 84044 VRFY OK  ** Signature verified **
2003.01.28 09:07:48.395 84044 PMDN OK  ** Original-Content-MIC found in MDN **
2003.01.28 09:07:48.495 84044 HPOS OK  Outbound session stopping - batch=[0]
2003.01.28 09:08:48.061 64089 HPIS OK  HTTP inbound session started
2003.01.28 09:08:48.081 64089 HPIS OK  HTTP client: 9.24.104.248:3164
2003.01.28 09:08:48.201 64089 DECR OK  ** Content decrypted **
2003.01.28 09:08:48.301 64089 VRFY OK  ** Signature verified **
2003.01.28 09:08:48.462 64089 STMQ OK  Data stored
2003.01.28 09:08:48.572 64089 HPIS OK  HTTP inbound session stopping
```

Supplier sends documents

To send documents from Supplier (TPI) to Retailer2 (iSoft), we can again use the Document Generator tool, as demonstrated earlier in 3.2.4, “Validation of the setup” on page 110. Set the Sender’s ID and Receiver’s ID to the correct values and click **Generate** to generate a sample EDI document.

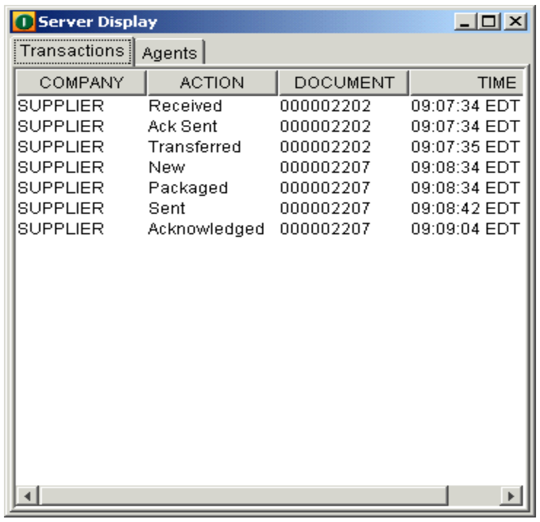


Sender's ID:	SUPPLIER
Receiver's ID:	RETAILER2
Control ID:	2207
Output Directory:	D:\CrossWorldsTPI\data\SUPPLIER\ediout
Documents to generate:	1
Document size (K):	10
Regeneration time (min):	1

Generated document: 2,206

Figure 3-35 Using the Document Generator tool

Depending on the polling rates within TPI, the server will detect the new file, package it and send it out. The output of iSoft's P2PAgent for this transaction is shown in Example 3-10 on page 131 (lower part).



COMPANY	ACTION	DOCUMENT	TIME
SUPPLIER	Received	000002202	09:07:34 EDT
SUPPLIER	Ack Sent	000002202	09:07:34 EDT
SUPPLIER	Transferred	000002202	09:07:35 EDT
SUPPLIER	New	000002207	09:08:34 EDT
SUPPLIER	Packaged	000002207	09:08:34 EDT
SUPPLIER	Sent	000002207	09:08:42 EDT
SUPPLIER	Acknowledged	000002207	09:09:04 EDT

Figure 3-36 TPI Server Display window

Figure 3-36 shows the different states for the transaction between TPI and iSoft, including the Acknowledgment from iSoft.

3.4 Integration between WebSphere Data Interchange and TPI

Business documents such as purchase orders and invoices are in general created, managed, and stored in back-end systems that are very much enterprise-specific. As a result, the structure and data format of those documents can be very different from what industry organizations have agreed upon as a standard. An EDI 850 document, for purchase orders, has a specific layout that internal applications cannot always generate. This mismatch between the internal document and the standard document to be used in B2B transactions has resulted in the development of EDI-specific document translators. An example of such a product is WebSphere Data Interchange.

Since the use of EDI translators is very common, a typical implementation of the TPI Server involves the integration with products such as WebSphere Data Integration.

Throughout this section, we are assuming that you have a working WebSphere Data Interchange environment with at least one client and one server. We are also assuming that the server is running on the same machine as the TPI Server.

For more information about configuring a WebSphere Data Interchange environment, refer to the Redpaper *WebSphere Data Interchange Installation and Configuration*, REDP3600.

3.4.1 Processing received EDI documents

In this section, we will describe the flow of inbound documents, where documents are received by the TPI Server and stored in queues or files. WebSphere Data Interchange picks up these documents and performs the required translation into an internal format, which happens to be an XML format.

Figure 3-37 shows the overall data flow. Data that is received from Retailer1 or Retailer2 arrives in a queue, while data from Retailer3 is stored in a files. Note that the configuration of the TPI Server of the MQ integration (see Figure 3-18 on page 115) has the same queue for incoming documents for both Retailer1 and Retailer2. WebSphere Data Interchange can find the appropriate trading partner information in the ISA segment of the incoming EDI document and use that information to apply the correct translation rules. Also note that a single company profile can only have one destination for a document type. If we built a TPI configuration to match Figure 3-37, we would need to create a separate company profile to handle the inbound EDI documents that need to be stored in a directory. For outbound communication, TPI can pick up files from the EDI_OUT directory and from the EDI_OUT queue in parallel. For the purposes of this section, we simply assume that EDI documents are received in files and in messages; we need to work with WebSphere Data Interchange to handle both sources of data.

The configuration of WebSphere Data Interchange involves the following steps:

1. Definition of trading partner profiles for all retailers and the supplier itself.
2. Document definition:
   a. Import of the EDI 850 document definition.
   b. Import of a DTD, matching the internal representation of a purchase order
3. Definition of the translation map.
4. Definition of mailboxes, network profiles, queue profiles and service profiles.
5. Definition of the rules associated with the map.
6. Creation of a command file to process incoming EDI files.
7. Definitions of queue and process objects in WebSphere MQ to support the automatic translation of incoming EDI documents in queues.

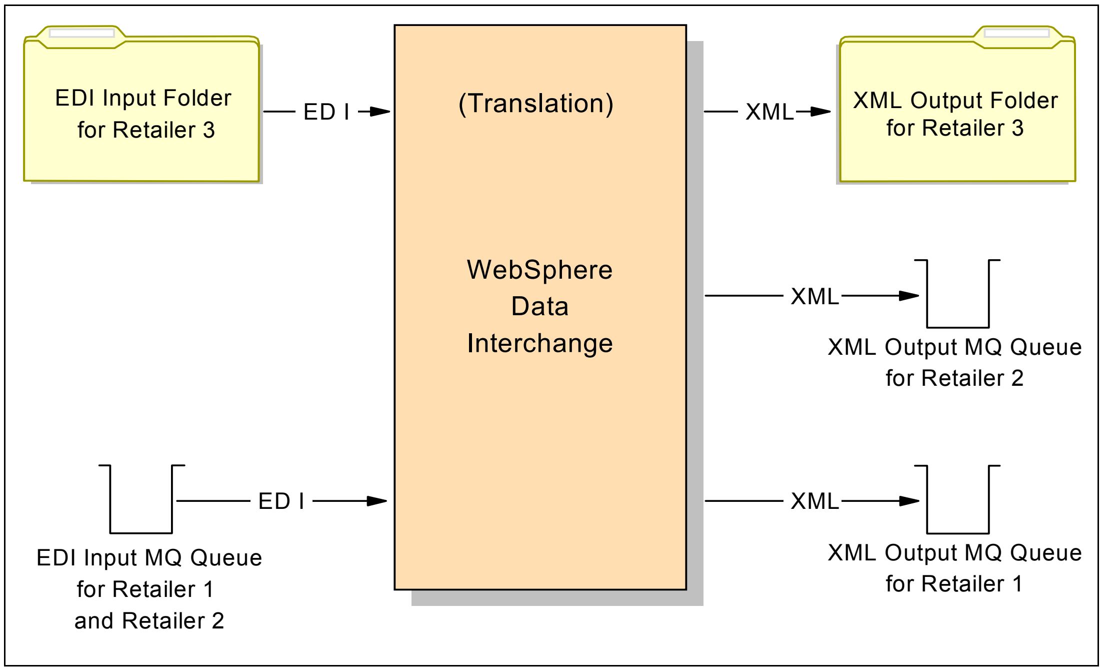
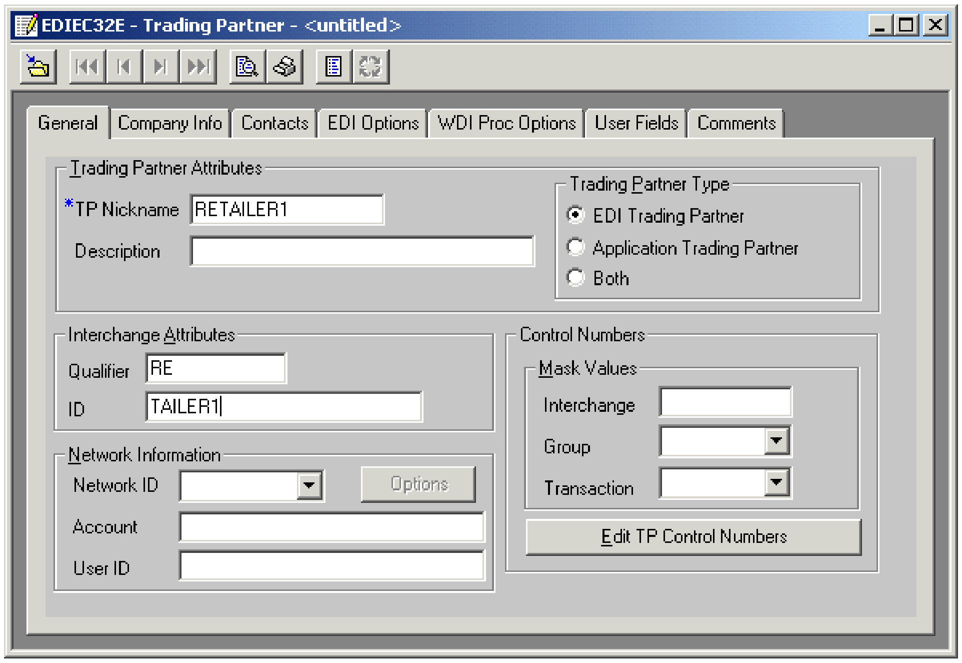

EDI Input Folder for Retailer 3
ED I
(Translation)
WebSphere Data Interchange
XML
XML Output Folder for Retailer 3
XML
XML Output MQ Queue for Retailer 2
ED I
EDI Input MQ Queue for Retailer 1 and Retailer 2
XML
XML Output MQ Queue for Retailer 1

Figure 3-37 Inbound data flow

Trading partner setup

Start the WebSphere Data Interchange client program and click the trading partner setup icon in the tool bar. A new window will appear with a list of currently defined trading partners. Click the new document button in the tool bar. On the General tab, provide a nickname for the new trading partner (RETAILER1) and fill in the values for the Interchange Qualifier and ID (see Figure 3-38), which are found in the EDI document. These fields in the ISA segment will be used by WebSphere Data Interchange as a key to locate the trading partner document (nickname) and that nickname will then be used in further processing within WebSphere Data Interchange. The value for the nickname is only relevant within the scope of WebSphere Data Interchange.



*TP Nickname: RETAILER1
Description:
Trading Partner Type: [● EDI Trading Partner] ○ Application Trading Partner ○ Both
Qualifier: RE
ID: TAILER1
Network ID:
Account:
User ID:
Interchange:
Group:
Transaction:

Figure 3-38 Trading partner definition

Repeat this process for the trading partner Retailer2.

EDI document definition

For each EDI document standard, there are a number of versions and releases. You can download import files for ANSI X12 and other standards from the following Web site:

http://www-3.ibm.com/software/integration/appconn/wdi/downloads/

For our purposes, we have used the ANSI X12 Standard Version 4 Release 3. This standard can be downloaded as an export/import file (eif) for WebSphere Data Interchange. Select **File -> Open Import File** to load the definitions in your database and point the file browser to the downloaded file X12V4R3.eif. A window of document definitions will appear that lists all the definitions that are included in this file. Select the documents that you will need, for example 850 and 855. Use the Control key to select multiple definitions. Press the **Enter** key and select the correct system (database) to import the document definitions. Note that both the 850 and 855 documents are quite large and contain many segments and fields. As a result, the import might take a while to complete.

XML document definition

WebSphere Data Interchange supports the import of DTDs to define XML documents to the translator. In general, the structure of XML documents that are used by a company's internal applications is very specific. Since this redbook is not just about EDI translation and mapping, we will use a simple DTD that helps us focus on the integration issue instead of the mapping issue. Example 3-11 lists the DTD, while Example 3-12 provides a sample message that complies with this DTD.

Example 3-11 DTD for XML document representing a purchase order

```
<?xml encoding="US-ASCII"?>

<!ELEMENT PO (Header, Detail)>

<!ELEMENT Header (FROM, TO, PONO,PODate)>
<!ELEMENT FROM (#PCDATA)>
<!ELEMENT TO (#PCDATA)>
<!ELEMENT PONO (#PCDATA)>
<!ELEMENT PODate (#PCDATA)>

<!ELEMENT Detail (QTY, ITEMNO, DESC)>
<!ELEMENT QTY (#PCDATA)>
<!ELEMENT ITEMNO (#PCDATA)>
<!ELEMENT DESC (#PCDATA)>
```

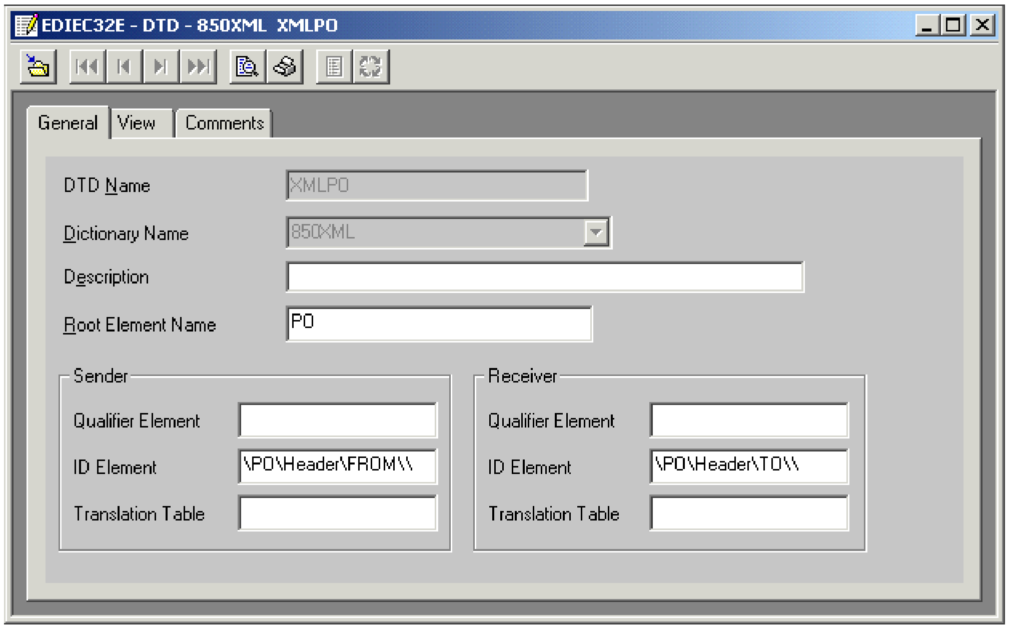
Example 3-12 Sample XML purchase order

```
<?xml version="1.0"?>
<!DOCTYPE Order SYSTEM "XMLPO.dtd">

<PO>
   <Header>
   <FROM>RETAILER1</FROM>
   <TO>SUP1</TO>
   <PONO> 12345669 </PONO>
   <PODate> 20021018 </PODate>
   </Header>

   <Detail>
     <QTY> 1000 </QTY>
     <ITEMNO> ABC2 </ITEMNO>
     <DESC> Some product </DESC>
   </Detail>
</PO>
```

To import this DTD in WebSphere Data Interchange, you need to have a dictionary to hold the DTD. If you do not already have a dictionary, or if you want to create a new one to store this DTD, open the XML window by clicking the XML button in the main tool bar of WebSphere Data Interchange. A new window will appear that lists XML dictionaries and XML DTDs. Select the tab **XML Dictionary** and click **File -> New** to create a new dictionary. Name the new dictionary 850XML, for example. Save the new dictionary by clicking **File -> Save**. Now you can import the DTD itself. Select **File -> Open Import File** to start this import process. In the file browser window, change the file type property from `Export/Import Files (*.eif)` to `XML DTD File (*.dtd)`. During the import, you will be prompted to provide the following information:

- DTD file name
- DTD object name in WebSphere Data Interchange, for example `POXML`
- Name of the dictionary, to be selected from a drop-down box
- Name of the target database, to be selected from a list
- Name of the root element: set this to `PO` for the DTD listed in Example 3-11.

When the import has completed, open the DTD document within WebSphere Data Interchange (see Figure 3-39). Within this document, you can name the XML elements that identify the sender and receiver. If the names in the XML document do not directly match the EDI names, you can provide the name of a translation table that WebSphere Data Interchange can use at runtime to look up the correct partner information. Note that this information is used when building an EDI.


EDIEC32E - DTD - 850XML XMLPO

DTD Name	XMLPO
Dictionary Name	850XML
Description	
Root Element Name	PO

Sender

Qualifier Element	
ID Element	\PO\Header\FROM\\
Translation Table	

Receiver

Qualifier Element	
ID Element	\PO\Header\TO\\
Translation Table	

Figure 3-39 Finding partner information in the XML document

The DTD that was shown in Example 3-11 on page 135 has only one field to provide information about the sender and another field to provide information about the receiver. To make the link, set:

- field ID Element for Sender: `\PO\Header\FROM\\`
- field ID Element for Receiver: `\PO\Header\TO\\`

Note the double backslash symbol at the end.

The result is that we can now build custom rules for routing and mapping based on who the target or source partner is for a given document. For an incoming EDI 850 document, the information in the ISA segment will be copied into the XML elements that were set in Figure 3-39 and we will not require specific mapping statements in the translation map.

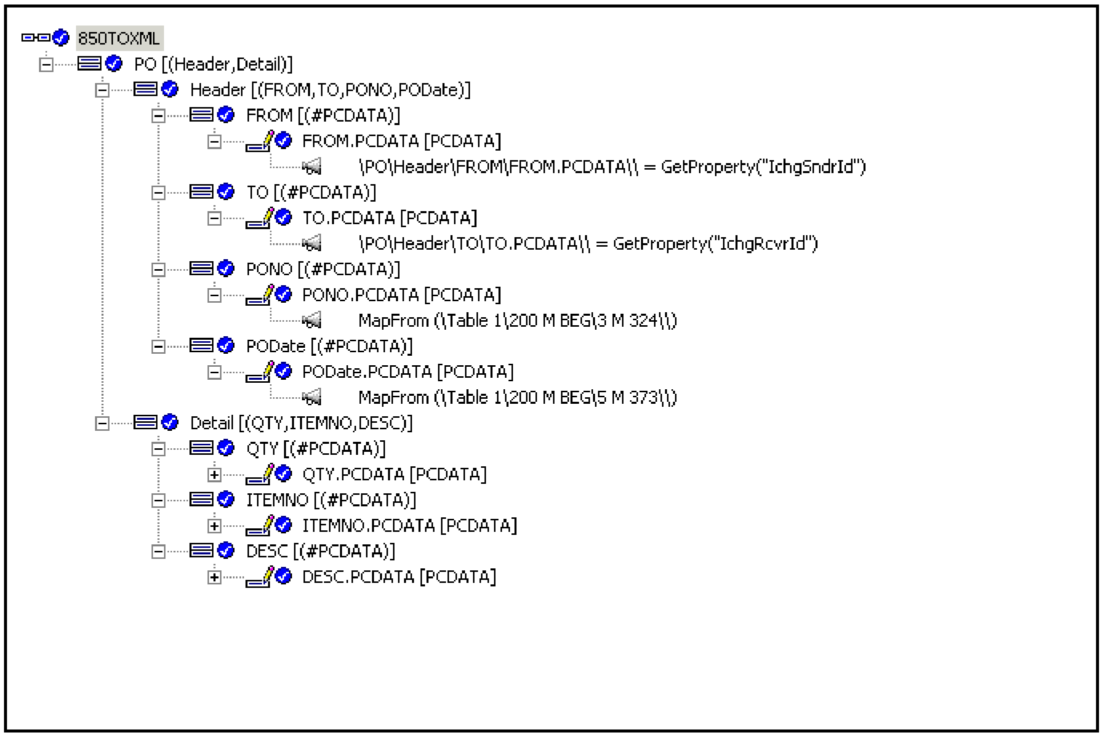
Definition of the translation map

Since the Supplier company expects to receive EDI 850 documents (purchase orders) from its customers, the retailers, we need a map that translates the incoming EDI document into an XML document that can be handled by the Supplier’s internal systems. To create this map, start the map editor by clicking the mapping button in the main tool bar of WebSphere Data Interchange. Within the map editor, either click the new document button or select **File -> New**. A map definition wizard will appear. Provide the following values for these parameters:

- Map name: `850TOXML`
- Target or source based: `Target`
- Source document definition: `EDI Standard`
- Source dictionary: `X12V4R3`

- Source EDI transaction: `850`
- Target syntax type: `XML`
- Target dictionary: `850XML`
- Target DTD: `POXML`

Given the simple structure of the XML document, the mapping itself is quite easy too. Note that the target XML document is probably too simple for most practical purposes. We do not intend to cover all options for mapping EDI documents in this redbook. Figure 3-40 shows the mapping statements between the EDI segments and fields and the target XML document. These statements are obtained by dragging the EDI field onto the XML element.



```
850TOXML
  PO [(Header,Detail)]
    Header [(FROM,TO,PONO,PODate)]
      FROM [(#PCDATA)]
        FROM.PCDATA [PCDATA]
          \PO\Header\FROM\FROM.PCDATA\\ = GetProperty("IchgSndrId")
      TO [(#PCDATA)]
        TO.PCDATA [PCDATA]
          \PO\Header\TO\TO.PCDATA\\ = GetProperty("IchgRcvrId")
      PONO [(#PCDATA)]
        PONO.PCDATA [PCDATA]
          MapFrom (\Table 1\200 M BEG\3 M 324\\)
      PODate [(#PCDATA)]
        PODate.PCDATA [PCDATA]
          MapFrom (\Table 1\200 M BEG\5 M 373\\)
    Detail [(QTY,ITEMNO,DESC)]
      QTY [(#PCDATA)]
        QTY.PCDATA [PCDATA]
      ITEMNO [(#PCDATA)]
        ITEMNO.PCDATA [PCDATA]
      DESC [(#PCDATA)]
        DESC.PCDATA [PCDATA]
```

Figure 3-40 Mapping the EDI 850 document into an XML document

The two statements in Figure 3-40 that were not created by drag-and-drop are the statements to fill in the elements FROM and TO in the Header element. To pass along the information of the EDI ISA header into the XML document, you can add an assignment command and call the function GetProperty to obtain the value of a field in the ISA header. Adding an assignment is performed by right-clicking the target field and selecting **Insert After -> Command -> Assignment** from the context menu. This will open a command editor to assist you in building the command.

Setup within WebSphere Data Interchange for SUP1 and RETAILER1

Now that we have a map, we need to tie it to a couple of other objects in WebSphere Data Interchange, by creating a rule for the map. However, before we can do that, we need to create those other objects in WebSphere Data Interchange.

The queue EDI_IN will receive EDI documents from the Retailers 1 and 2. The documents will be written in this queue by the TPI Server. To make this queue known to WebSphere Data Interchange, we need to define a queue profile. Open the setup window in WebSphere Data Interchange and select the tab **MQSeries Queues**. Select **File -> New** to create a new document. You need to name this document, for example `EDI_IN`. Set the name of the queue to `EDI_IN` and specify the name of the queue manager, `cw_studenta.queue.manager` in our example. If your documents are large, you should consider setting an appropriate value for the field Maximum Message Length. The default value is 32KB, which might not be sufficient for your environment. Select **File -> Save** to store this new document in the database.

Once WebSphere Data Interchange has translated the incoming EDI 850 document, it will need to send it to the internal system for processing. That destination might be a queue or a file in a given directory. We will describe both examples below. The target destination, either queue of the directory, can be dependent of the document and/or dependent of the trading partner. For most environments, the back-end system will not require specific locations for each trading partner. But it is quite common for purchase orders to be stored in a different location than, for example, requests for quotation.

In our example, we are going to assume that all purchase orders are to be sent to the same queue, called PURCHASE.ORDERS. As described before, create a queue profile PO_IN in WebSphere Data Interchange; set the queue name to `PURCHASE.ORDERS` and set the name of the queue manager, `cw_studenta.queue.manager` in our example. Save the document.

The next object we should define is a network profile. While it is possible to use a single network profile to describe the access to the queues EDI_IN and PURCHASE.ORDERS, it might be easier to separate these two. The queue EDI_IN might contain messages with an RFH2 header, while the messages for the queue PURCHASE.ORDERS should not have this header. To create network profiles, open the setup window in WebSphere Data Interchange and select the tab **Network Profiles**. Click **File -> New** to create a new document.

Note: The structure of the MQ message generated by TPI depends on what interface is used by TPI to interact with WebSphere MQ. If TPI is configured to use the JMS interface of WebSphere MQ, the message will have an RFH2 and this needs to be reflected in the network profile. If TPI is configured to use the standard MQ Client interface, as we have configured TPI in this chapter, the message will not contain an RFH2 header.

Create a network profile INBOX, with the following values:

- Network ID: `EDI_IN`
- Communication Routine: `VANIMQ`
- Network Program: `EDIMQSR`
- Network Parameters: `RECEIVEMQ=EDI_IN` (the name of the queue profile)

Save the document by clicking **File -> Save** and create a second network profile, with the following values:

- Network ID: `PO_IN`
- Communication Routine: `VANIMQ`
- Network Program: `EDIMQSR`
- Network Parameters: `SENDMQ=PO_IN` (the name of the queue profile)

The next step is to create mailboxes. A mailbox in WebSphere Data Interchange is a logical starting point or ending point for a translation. It can map onto a mailbox in a VAN solution or to any other resource, such as a queue or a file.

- Create mailbox EDI_IN with network ID EDI_IN.
- Create mailbox PO_IN with network ID PO_IN.

Finally, complete the setup within WebSphere Data Interchange by creating services profiles. A service profile is named after a mailbox and contains the WebSphere Data Interchange commands that need to be executed when a document arrives in a mailbox.

Create service profile EDI_IN with the following settings:

- Perform command: `PERFORM TRANSFORM WHERE INFILE(EDI_IN) SYNTAX(E) OUTFILE(PO_IN)`
- Output files: `PO_IN ..\xml\po_in.txt`

Create a service profile PO_IN with the following settings:

- Perform command: `PERFORM SEND WHERE REQID(PO_IN) OUTFILE(PO_IN) OUTTYPE(MQ) CLEARFILE(Y)`
- Input files: `PO_IN ..\xml\po_in.txt`

Creating a rule for the translation map

So far, we have created a transformation map and objects within WebSphere Data Interchange that control the flow of data. The last object to create is a rule (or usage) associated with a map. The rule will tell WebSphere Data Interchange when a map should be used.

Open the mapping window and select the map **850TOXML** that we have created previously. Select **Actions -> Usages**. Since a rule is associated with a map, some conditions are already set up front. It is clear that the map 850TOXML will only be used to transform EDI 850 documents that adhere to the standard X12V4R3, but typically, you will need more granularity. Often a map should only be used when the document comes from one partner or is going to be sent to one partner, while another map should be used for another partner. That is the kind of information that you can encode in a data transformation map. Select **File -> New** to create a new rule for the map 850TOXML. Set the following attributes:

- Usage Indicator: `Production`
- Trading Partners:
  - Sending: `ANY`
  - Receiving: `SUPPLIER`
- Make the map active by selecting the check box **Active**.

This results in a map and rule such that documents received from any partner and targeted for the internal trading partner SUPPLIER are all translated according to the same rules.

Updating WebSphere MQ resources

The translation process that we have described in the previous section assumes that messages will arrive in a queue called EDI_IN and that they can be written after translation in another queue, called PURCHASE.ORDERS. The queue EDI_IN was already created previously when we configured the TPI Server. Now, define the queue PURCHASE.ORDERS using WebSphere MQ Explorer.

To automate the translation process, we would like to use the triggering features of WebSphere MQ. Therefore, update the definition of the queue EDI_IN and set the following attributes using WebSphere MQ Explorer on the tab labeled Triggering:

- Trigger Control: `On`
- Trigger Type: `First`
- Initiation Queue: `WDI.INIT.Q`
- Process Name: `WDI.PROC`

Usually, the setup of a WebSphere Data Interchange server includes the creation of a number of WebSphere MQ objects. The commands to create these objects are available in the file wdimqcommands.txt in the samples directory of a WebSphere Data Interchange server installation. If you have executed these commands, then the queue WDI.INIT.Q already exists, as does the process WDI.PROC. If the process WDI.PROC and the queue WDI.INIT.Q do not exist for your queue manager, consult the file wdimqcommands.txt to find the requirements for these objects.

Completing the setup of WebSphere Data Interchange

The above configuration of WebSphere Data Interchange works fine for input received from Retailer1. The EDI 850 document from Retailer1 is written in the queue EDI_IN by the TPI Server and WebSphere Data Interchange will pick it up and translate it based on the rule that we created before. What needs to be done when the Retailer1 sends other types of EDI documents? Certainly, we need to make sure that the EDI definition for this document is imported and that an equivalent DTD is imported. Also, we will need to define a map between the EDI definition and the XML document. This new EDI document will arrive in the same queue called EDI_IN. WebSphere Data Interchange is able to detect that this is a different document than the EDI 850 and hence can pick up the correct map. However, will this new document be passed, after translation, to the same queue PURCHASE.ORDERS? Probably not. How is WebSphere Data Interchange going to perform this routing?

At this point, any message in the queue EDI_IN is going to be handed over to the queue PURCHASE.ORDERS. To tie the routing to the document type, and not the origin, perform the following changes to the configuration that you have created so far:

- Update the rule for 850TOXML. Set Output file to `PO_IN` and Type to `MQ` on the General tab of the data transformation rule.
- Update the service profile EDI_IN and remove the references to the output file on the **`perform`** command for the service profile EDI_IN. The updated perform command now looks like this:

```
PERFORM TRANSFORM WHERE INFILE(EDI_IN) SYNTAX(E)
```

At this time, the target destination (PO_IN) is set in the rule and can be tied to either trading partner combination and/or document combination.

For the rule associated with the map for the new EDI document, set the output file to a different file. Add a queue profile, a service profile, a mailbox profile, and a network profile to the WebSphere Data Interchange configuration.

When adding more trading partners (Retailer2, Retailer3 and so on), we need to make sure that these trading partners are known in WebSphere Data Interchange. The rule itself was not linked to a source trading partner. If you are sure that the rule for the translation of 850 documents is partner-independent, you can update it to make it an any-to-any rule. Alternatively, you need to create an additional rule, for example to set a specific output file. Or, you may need to create an additional map and rule, to handle specific translation and/or destination requirements.

Processing file-based input for WebSphere Data Interchange

When the TPI Server receives an EDI document and it is configured to store the document as a file in a given directory, it will generate a unique file name for each incoming document. The default generated name contains sender and receiver identification, followed by the document number obtained from the ISA segment and the extension *edi*. An example of such a name is shown below:

```
RETAILER1.TO.SUPPLIER_1001.edi
```

Besides variable file names, there is also the issue of how to start the EDI translation engine. When using WebSphere MQ, you can rely on MQ triggering to start the EDI translation engine. The most common solution is to use a scheduler tool and run a command file at a regular interval to process incoming EDI documents.

Both issues (variable file names and kicking off the translation process) can be solved in a variety of ways using command files and scripts. We describe here one solution that will mainly focus on the interaction between WebSphere Data Interchange and TPI and not on the scripting aspect.

Assuming that we have a configuration of TPI without MQ integration and with default system directory names, the EDI documents will be written by the TPI Server in the directory c:\CrossWorldsTPI\data\Supplier\ediin. Assume that the installation directory of WebSphere Data Interchange Server is C:\WDIServer32. In the directory C:\WDIServer32\runtime\dicmd, we created a command file, called wdi.bat with the following commands (see Example 3-13).

Example 3-13 Command file wdi.bat

```
echo off
For %%f in (c:\CrossWorldsTPI\data\Supplier\ediin\*.edi) do @translate.bat %%f
```

This command file will result in running another command file, translate.bat, as many times as there are EDI document files in the directory inbox\retailer3. We need to make sure that we process an EDI document once only. The set of file names will be built at the beginning of the execution of the command file wdi.bat. The name of each file will be passed as a parameter to translate.bat. It is clear that the command file translate.bat will have to move or rename the file when the processing is complete, otherwise it will be processed again the next time that the wdi.bat command file is executed. If a new document arrives during the period of time that wdi.bat is already running, it will not be found before the next time that wdi.bat is scheduled to run.

The contents of translate.bat are shown in Example 3-14. Since the command file knows the variable name of the file for which it is started, we can copy it to a fixed name, in this case edi.in. Then we make sure that the bin directory of WebSphere Data Interchange is part of the PATH environment variable. Next, we call the EDISERVR program, which is the WebSphere Data Interchange engine. That program is given some WDI commands via indirection. Finally, we copy the original source file to add the extension .processed to its name and delete the original file.

Since all files will be copied at some point to the file edi.in, we cannot run multiple instances of translate.bat at the same time. As a result, we cannot run multiple instances of wdi.bat at the same time. If this causes a problem for your environment, you will need to write smarter scripts to handle that.

Example 3-14 Command file translate.bat

```
echo %1
copy %1 c:\CrossWorldsTPI\data\Supplier\ediin\edi.in
set WDIRESTOREPATH=%PATH%
set PATH=C:\WDISERVER32\bin;%PATH%
ediservr < wdicmds.txt
set PATH=%WDIRESTOREPATH%
copy %1 %1.processed
del %1
```

Finally, let us look at the contents of the file wdicmds.txt, the contents of which are given in Example 3-15 on page 142.

Example 3-15 Contents of the file wdicmds.txt

```
set plan(WDIC);
init;
set file(PRTFILE,prtfile.txt);
set file(TRKFILE,trkfile.txt);
set file(EXPFILE, expfile.txt);
set file(EDI_IN,c:\CrossWorldsTPI\data\Supplier\ediin\edi.in);
set file(PO_IN,c:\CrossWorldsTPI\data\Supplier\ediin\po.in);
PERFORM TRANSFORM WHERE INFILE(EDI_IN) SYNTAX(E);
term;
```

You can easily see the correspondence between these commands and what we had configured before using the client interface of WebSphere Data Interchange. The **set file** commands correspond to the service profile settings where we had given values for similar parameters. Setting the plan to the name of the database was something we did previously in the file wdi.properties. The **PERFORM** command in Example 3-15 is the same command that we had in the service profile INBOX.

It should be noted that the translated XML document is always stored in a file called po.in. By default, WebSphere Data Interchange will append to this file, if it already exists. A single file with multiple XML documents might cause problems for other applications that are going to process this incoming order. If that is the case, an easy solution might be to add a copy command to translate.bat. For example:

```
copy c:\CrossWorldsTPI\data\Supplier\ediin\po.in %1.translated
```

3.4.2 Preparing EDI documents

The process of translating XML documents to EDI documents is conceptually not much different from the reverse process that we explained in the previous section. Figure 3-41 on page 143 shows this flow in a graphical way. Figure 3-41 on page 143 also shows that documents for Retailer2 and Retailer1 are in a different queue. However, they can just as well be stored in the same queue, called POACKQ. The translated documents should be written in the queue EDI_OUT, which is set in the company profile for Supplier in TPI.

In this section, we will use EDI 855 (purchase order acknowledgement) as the document that is being sent by the supplier to the retailer that has previously sent a purchase order.

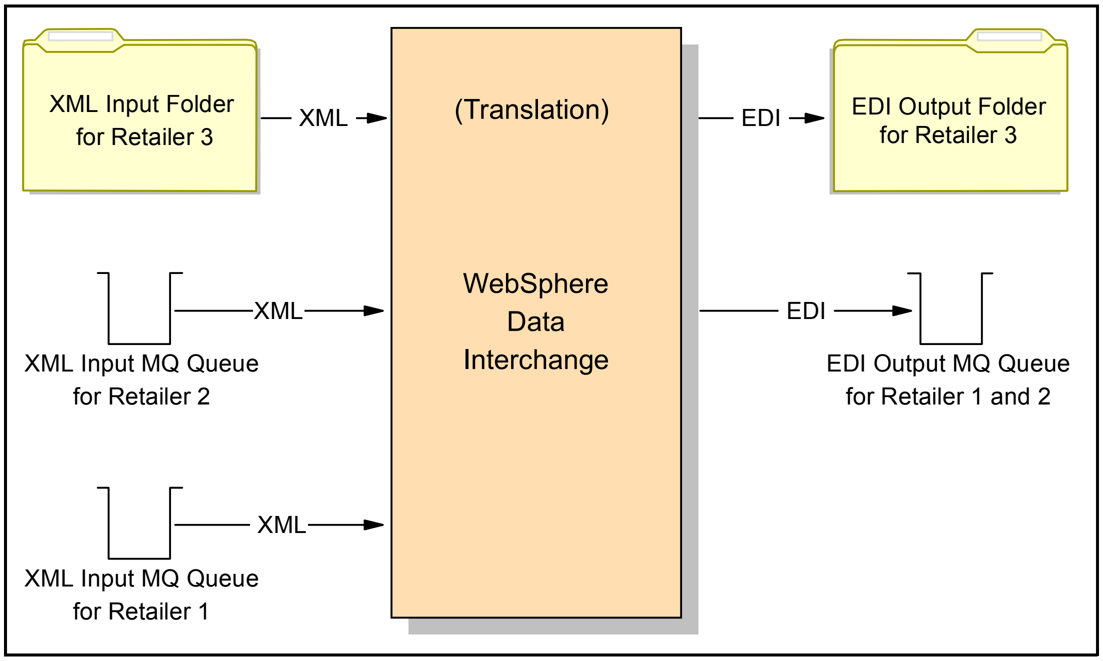

XML Input Folder for Retailer 3
XML
(Translation)
WebSphere Data Interchange
EDI
EDI Output Folder for Retailer 3
XML
XML Input MQ Queue for Retailer 2
EDI
EDI Output MQ Queue for Retailer 1 and 2
XML
XML Input MQ Queue for Retailer 1

Figure 3-41 Outbound data flow

The process of configuring WebSphere Data Interchange for this task consists of the following steps:

1. Definition of trading partner profiles for all retailers and the supplier itself. This step was completed when we described the setup for the inbound flow.
2. Document definition:
   a. Import of the EDI 855 document definition. During the import of the 850 document, we had also selected the 855 document. Thus, we can skip this step.
   b. Import of a DTD, matching the internal representation of a purchase order acknowledgment.
3. Definition of the translation map.
4. Definition of mailboxes, network profiles, queue profiles and service profiles.
5. Definition of the rules associated with the map.
6. Creation of a command file to process incoming XML files.
7. Definitions of queue and process objects in WebSphere MQ to support the automatic translation of incoming XML documents in queues.

XML document definition

In 3.4.1, "Processing received EDI documents" on page 132, we described how the company Supplier was processing incoming EDI 850 documents. In general, when a company receives such a document, it will respond with a purchase acknowledgement, which is an 855 document in EDI terminology. The internal systems of the company Supplier will likely not generate an 855 document directly. The format will be company- and application-specific. Example 3-16 on page 144 shows a simple DTD representing an XML document that contains information typically found in a PO Ack. Again, the sample DTD is simple to avoid losing focus in this redbook.

Example 3-16   DTD representing a PO acknowledgement

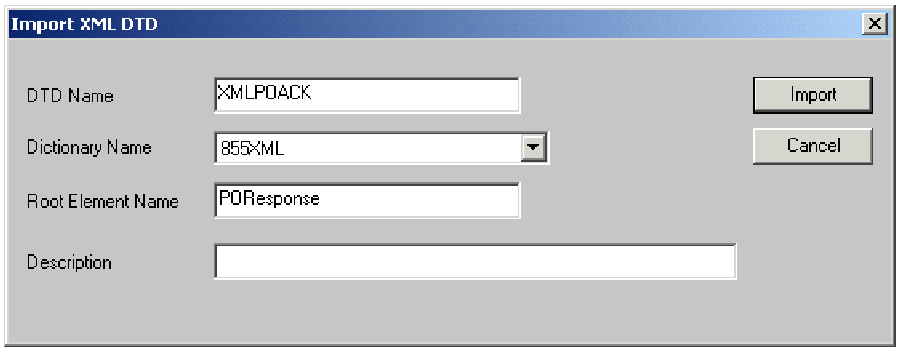
```
<?xml encoding="US-ASCII"?>
<!ELEMENT POResponse (Header,Detail)>

<!ELEMENT Header (PONumber,TargetPartnerID,Response)>

<!ELEMENT PONumber (#PCDATA)>
<!ELEMENT TargetPartnerID (#PCDATA)>
<!ELEMENT Response (#PCDATA)>

<!ELEMENT Detail (ItemNumber,Quantity,Description)>

<!ELEMENT ItemNumber (#PCDATA)>
<!ELEMENT Quantity (#PCDATA)>
<!ELEMENT Description (#PCDATA)>
```

Example 3-17 shows a sample message that complies with the DTD of Example 3-16. It contains an ID representing the target partner, a PO ID and a response field. Farther down, we also see a detailed view of the actual order.

Example 3-17   Sample XML document representing a PO acknowledgement

```
<?xml version="1.0"?>
<POResponse>
   <Header>
      <PONumber>P12347</PONumber>
      <TargetPartnerID>RETAILER2</TradingPartnerID>
      <Response>AT</Response>
   </Header>
   <Detail>
      <ItemNumber>00123</ItemNumber>
      <Quantity>10</Quantity>
      <Description>Parts</Description>
   </Detail>
</POResponse>
```

The DTD is imported in a dictionary called 855XML and the root element name is set to `POResponse` (see Figure 3-42).



DTD Name: XMLPOACK
Dictionary Name: 855XML
Root Element Name: POResponse
Description:

Figure 3-42   Import XML DTD in WebSphere Data Interchange

After importing the DTD, we need to tell WebSphere Data Interchange about the role of the field TargetPartnerID. Open the DTD again in WebSphere Data Interchange and set the field ID Element for Receiver to `\POResponse\Header\TargetTradingPartnerID\\`. Refer to Figure 3-39 on page 136 to see where we did this for the PO DTD.

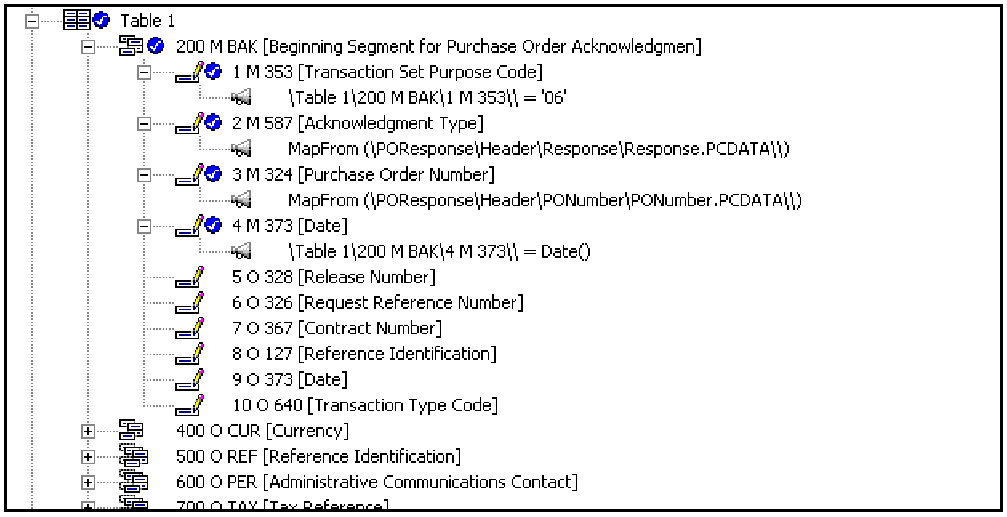
Definition of the translation map

The next step is to create the translation map to translate the XML document to the corresponding EDI 855 document. Open the map editor and select **File -> New**. Use the following values in the map definition wizard:

- Map name: `XMLTO855`
- Target or source based: `Target`
- Source document definition: `XML`
- Source dictionary name: `855XML`
- Source DTD: `XMLPOACK`
- Target document definition: `EDI`
- Target dictionary name: `X12V4R3`
- Target EDI standard transaction: `855`

After reviewing your selections in the summary window of the wizard, select **Finish** to start the actual mapping between the XML document and the EDI transaction.

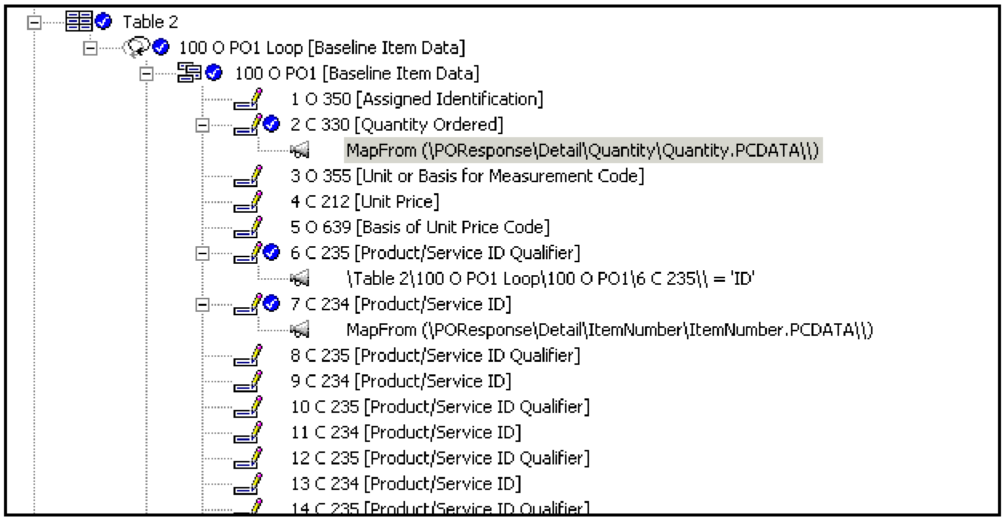
Figure 3-43 shows the first portion of the map. It shows the mappings for Table 1. The data segments of the BAK segment are filled in by dragging the corresponding elements from the XML document. The field Date is filled in using an assignment command in which we invoke the built-in function Date(). The field Transaction Set Purpose Code is filled in using an assignment in which the field is set to the constant value `06`.



```
Table 1
  200 M BAK [Beginning Segment for Purchase Order Acknowledgmen]
    1 M 353 [Transaction Set Purpose Code]
        \Table 1\200 M BAK\1 M 353\\ = '06'
    2 M 587 [Acknowledgment Type]
        MapFrom (\POResponse\Header\Response\Response.PCDATA\\)
    3 M 324 [Purchase Order Number]
        MapFrom (\POResponse\Header\PONumber\PONumber.PCDATA\\)
    4 M 373 [Date]
        \Table 1\200 M BAK\4 M 373\\ = Date()
    5 O 328 [Release Number]
    6 O 326 [Request Reference Number]
    7 O 367 [Contract Number]
    8 O 127 [Reference Identification]
    9 O 373 [Date]
    10 O 640 [Transaction Type Code]
  400 O CUR [Currency]
  500 O REF [Reference Identification]
  600 O PER [Administrative Communications Contact]
```

Figure 3-43 Building an EDI 855 document - step 1

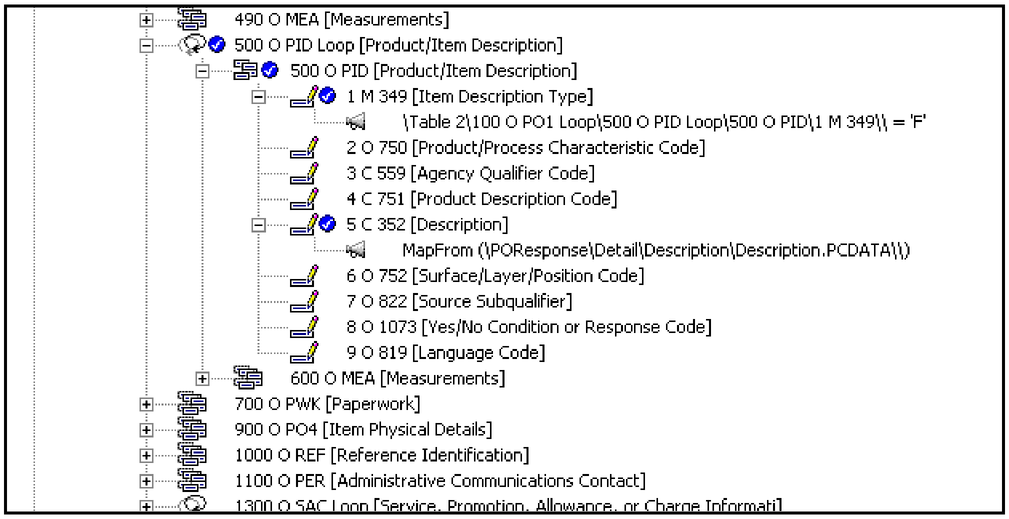
Figure 3-44 on page 146 shows the first part of the mapping for Table 2. Two data elements are mapped directly from the source XML document, while the element Product/Service ID Qualifier is filled in using an assignment.



```
Table 2
  100 O PO1 Loop [Baseline Item Data]
    100 O PO1 [Baseline Item Data]
      1 O 350 [Assigned Identification]
      2 C 330 [Quantity Ordered]
        MapFrom (\POResponse\Detail\Quantity\Quantity.PCDATA\\)
      3 O 355 [Unit or Basis for Measurement Code]
      4 C 212 [Unit Price]
      5 O 639 [Basis of Unit Price Code]
      6 C 235 [Product/Service ID Qualifier]
        \Table 2\100 O PO1 Loop\100 O PO1\6 C 235\\ = 'ID'
      7 C 234 [Product/Service ID]
        MapFrom (\POResponse\Detail\ItemNumber\ItemNumber.PCDATA\\)
      8 C 235 [Product/Service ID Qualifier]
      9 C 234 [Product/Service ID]
      10 C 235 [Product/Service ID Qualifier]
      11 C 234 [Product/Service ID]
      12 C 235 [Product/Service ID Qualifier]
      13 C 234 [Product/Service ID]
      14 C 235 [Product/Service ID Qualifier]
```

Figure 3-44 Building an EDI 855 document - step 2

Figure 3-45 shows the second part of the mapping for Table 2. One element is filled using a direct mapping from the corresponding element in the XML document, while the other element is filled in using an assignment.



```
    490 O MEA [Measurements]
    500 O PID Loop [Product/Item Description]
      500 O PID [Product/Item Description]
        1 M 349 [Item Description Type]
          \Table 2\100 O PO1 Loop\500 O PID Loop\500 O PID\1 M 349\\ = 'F'
        2 O 750 [Product/Process Characteristic Code]
        3 C 559 [Agency Qualifier Code]
        4 C 751 [Product Description Code]
        5 C 352 [Description]
          MapFrom (\POResponse\Detail\Description\Description.PCDATA\\)
        6 O 752 [Surface/Layer/Position Code]
        7 O 822 [Source Subqualifier]
        8 O 1073 [Yes/No Condition or Response Code]
        9 O 819 [Language Code]
      600 O MEA [Measurements]
    700 O PWK [Paperwork]
    900 O PO4 [Item Physical Details]
    1000 O REF [Reference Identification]
    1100 O PER [Administrative Communications Contact]
    1300 O SAC Loop [Service, Promotion, Allowance, or Charge Informati]
```

Figure 3-45 Building an EDI 855 document - step 3

Setup within WebSphere Data Interchange

As we have done for the inbound flow, we need to set up a number of objects in WebSphere Data Interchange to support this map.

Open the setup window in WebSphere Data Interchange and define an MQSeries queue profile POACKQ for the queue POACKQ. Set the name of the queue manager in this profile. Define a second MQSeries queue profile called EDI_OUT for the queue EDI_OUT.

Notice the implied naming convention for the queues. The queue POACKQ, like the queue PO_IN that was used before, has a name that implies a function, since the contents of the queue will be application-dependent. The queues EDI_IN and EDI_OUT are not application-dependent. The queue EDI_IN was used for EDI 850 documents but can be used for any type of document for which the destination is the internal system of the Supplier. The queue EDI_OUT is to be used for any type of document targeted for any trading partner.

The next step is the definition of two network profiles, which are defined in the setup window of WebSphere Data Interchange.

- Create network profile EDI_OUT with the following values:
  - Network ID: `EDI_OUT`
  - Communication Routine: `VANIMQ`
  - Network Program: `EDIMQSR`
  - Network Parameters: `SENDMQ=EDI_OUT`
- Create network profile POACKQ with the following values:
  - Network ID: `POACKQ`
  - Communication Routine: `VANIMQ`
  - Network Program: `EDIMQSR`
  - Network Parameters: `RECEIVEMQ=POACKQ`

Next, we need some mailboxes to represent the new destination and source queues. Create the following mailboxes:

- EDI_OUT: set network profile ID to `EDI_OUT`
- POACKQ: set network profile ID to `POACKQ`

Finally, we need to create service profiles that describe the actions that WebSphere Data Interchange should perform when documents are posted in a mailbox.

- Service profile EDI_OUT
  - Perform command:
    `PERFORM SEND WHERE REQID(EDI_OUT) CLEARFILE(Y)`
  - Input files: EDI_OUT - ..\xml\edi_out.txt
- Service profile POACKQ
  - Perform command:
    `PERFORM TRANSFORM WHERE INFILE(POACKQ) SYNTAX(X)`
  - Input files: EDI_OUT - ..\xml\edi_out.txt

Tip: Since these profiles are all similar, you can use the copy function of WebSphere Data Interchange by selecting the menu option **Action -> Copy**.

Creating an envelope profile

Before we can tie everything together, we need one more object in WebSphere Data Interchange: an envelope profile. EDI documents can be grouped in a single file and surrounded by an envelope. An envelope can contain documents of different transactions. In our setup, we always process (send and receive) the documents as soon as they become available. You may have situations where you want to batch the documents before enveloping them and then passing them to the trading partner. However, even when you send and receive the documents one at a time, you need to have an envelope.

Envelope profiles are used to set values for specific segments in an EDI document, such as the ISA segment. The ISA segment contains, beside other fields, the fields Sender and Receiver ID and the fields Sender and Receiver qualifier. In the DTD definition, we have made the link between the XML element TargetPartnerID and the Receiver ID. However, no information was available in the XML document to fill in the field Sender ID. We could have set a value in the map, using the setProperty built-in function. However, this would result in a value that is independent of the target partner. Given that you might have a requirement for several company profiles, we may need a different approach. An elegant solution to set the correct value (SUPPLIER) in the ISA segment is to use separate X envelope profiles.

Envelope profiles are managed in the setup window. Different types exist, corresponding to different EDI standards. Since we have used X12 documents, we will need to create an X profile.

Select the tab **X Envelope Profiles** and click **File -> New**. Name the profile 855SUP1. Select the tab **Interchange Header (ISA)** and set the field ISA06 to `PPLIER` and the ISA05 field to `SU`. Save the document. Note that TPI will combine the ISA05 and ISA06 field to map it to its company and/or partner profile names. SU concatenated with PPLIER gives us exactly the company ID that was set during the creating of the company profile in TPI (see 3.2.2, “Company profile setup for Supplier” on page 103). When you have a naming conflict between what you can store in the ISA segment and what you have configured in TPI, you can provide a secondary ID in TPI for a company profile to tie different partner names together.

Creating a rule for the map

Given that the destination of the translated document is different based on the target partner ID, we will need to define three new rules for the map XMLTO855. Open the map editor and select the map **XMLTO855** that we created before. Click **Action -> Usages** to open the rules editor. Select **File -> New** to create a rule and set these fields to the following values:

- Usage Indicator: `Production`
- Trading Partners:
  - Sending: `ANY`
  - Receiving: `RETAILER1`
- Output file and type: `EDI_OUT` and `MQ`
- Make the map active by selecting the check box **Active**.
- Select the tab Envelope Attributes
  - Envelope Type: `X`
  - Envelope Profile Name: `855SUP1`

Create a second and third rule for Retailer2 and Retailer3 with matching values.

Updating WebSphere MQ resources

The queue EDI_OUT was defined previously to support the configuration of the TPI Server. In order to validate the WebSphere Data Interchange configuration discussed above, we need to define a queue called POACKQ. This queue could be modeled after the queue EDI_IN that was defined to support the inbound data flow. The queue POACKQ should have the following values for these attributes on the tab labeled Triggering:

- Trigger Control: `On`
- Trigger Type: `First`
- Initiation Queue: `WDI.INIT.Q`
- Process Name: `WDI.PROC`

These values can be set by using WebSphere MQ Explorer. Note that it is assumed that the objects WDI.INIT.Q and WDI.PROC have been created as part of the standard configuration of WebSphere Data Interchange.

When you write an MQ message (such as the one shown in Example 3-17 on page 144) on the queue POACKQ, the WDIAdapter program should be launched by the WebSphere MQ Trigger Monitor program and the queue EDI_OUT should contain a message as shown in Example 3-18 on page 149.

Example 3-18 Sample translated EDI document

```
00000000 ISA*  *          *  *          *SU*PPLIER *RE*TAILER2 *060303*1506* *     *000000009* *P*:!
00000106 GS*PR*  *  *060303*1506*000000009* *004030!
00000149 ST*855*000000009!
00000166 BAK*06*AT*P12347*20030306!
00000192 PO1**10****ID*00123!
00000212 PID*F****Parts!
00000227 SE*5*000000009!
00000242 GE*1*000000009!
00000257 IEA*1*000000009!
00000273 .
```

Processing file-based input for WebSphere Data Interchange

As shown in Figure 3-41 on page 143, the outbound flow for Retailer3 is file-based. WebSphere Data Interchange needs to read files containing XML documents from a given directory and write them in the directory that is used by the TPI Server. As with the inbound flow, we need to make sure that script files are written in such a way that each file is processed exactly once. The TPI Server will poll the directory that was provided in the company profile (the tab System Directories).

On the input side, we need to assume again that the application that generates the XML files uses file names that are unique so that a file is not overwritten by this application before WebSphere Data Interchange has translated the XML document into an EDI document.

And finally, there is again the issue of automating the translation process, since we cannot rely on WebSphere MQ triggering to start the translation engine.

To handle all these issues, we will again present some script files. The first script file is called by a scheduler program at regular intervals and looks for files in the directory C:\WDIServer32\outbox\retailer3 with an extension of .file.txt. For each found file, the command file translate_out.bat is called, passing it the name of the XML file. Such a command file is shown in Example 3-19.

Example 3-19 Command file wdi_out.bat

```
echo off
For %%f in (c:\WDIServer32\outbox\retailer3\*.txt) do @translate_out.bat %%f
```

The second command file, translate_out.bat, prepares the WebSphere Data Interchange environment by setting the PATH environment variable correctly. It also copies the current file (passed to the command file as the first argument) to the intermediate file xml.in and then calls the actual translation engine. When the engine returns, the output file is copied to the ediout directory where TPI is polling for new files.

Example 3-20 Command file translate_out.bat

```
echo %1
copy %1 c:\WDIServer32\outbox\retailer3\xml_in
set WDIRESTOREPATH=%PATH%
set PATH=C:\WDISERVER32\bin;%PATH%
ediservr < wdi_out_cmds.txt
copy C:\WDIServer32\outbox\retailer3\edi_out c:\CrossWorldsTPI\data\Supplier\ediout\%1.edi
copy %1 %1.processed
del %1
set PATH=%WDIRESTOREPATH%
```

The actual WebSphere Data Interchange commands are stored in the file wdi_out.cmds, shown in Example 3-21. Similar to the inbound flow, the commands consist of a series of environment setup commands followed by the familiar **PERFORM** command.

Example 3-21 WebSphere Data Interchange commands

```
set plan(WDIC);
init;
set file(PRTFILE,prtfile.txt);
set file(TRKFILE,trkfile.txt);
set file(EXPFILE, expfile.txt);
set file(XML_IN,c:\WDIServer32\outbox\retailer3\xml_in);
set file(POACK,c:\WDIServer32\outbox\retailer3\edi_out);
PERFORM TRANSFORM WHERE INFILE(XML_IN) SYNTAX(X) OUTFILE(POACK);
term;
```

The generated files, with the extension .edi, are now ready for transmission by the TPI Server program and will be picked up by it at the next polling interval.

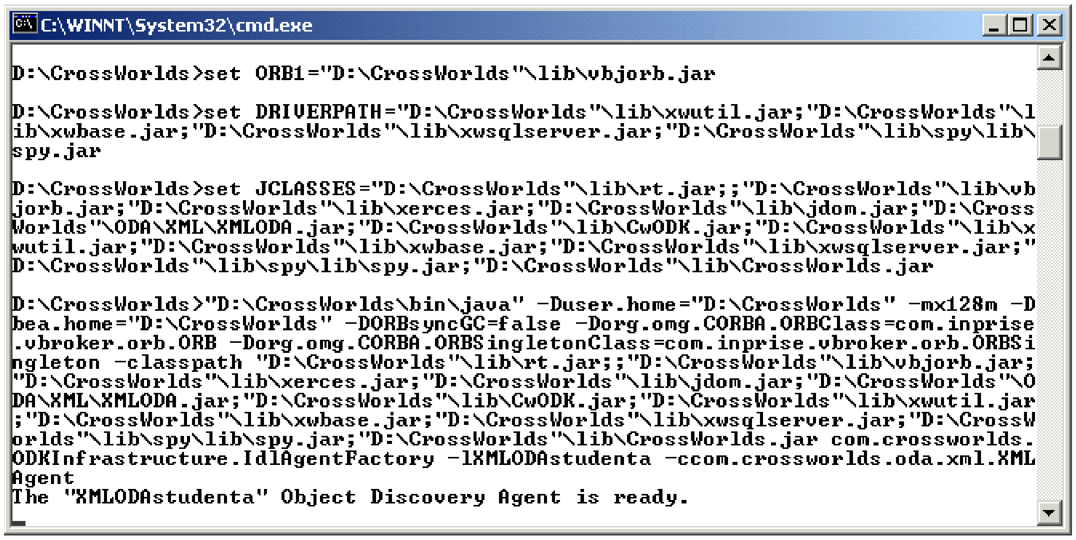
3.5 Integration between the Interchange Server, WebSphere Data Interchange and TPI

The Interchange Server (ICS) is often used as a platform for integrating applications within an enterprise. While we cannot cover all aspects of using this technology in a single redbook, this section will describe some typical operations that allow the ICS to interact with WebSphere Data Interchange. We will cover the use of the MQSeries Connector to send and receive data to and from products such as WebSphere Data Interchange. The use of other connectors, such as JText Connector, is very similar. One can also use the TPI Connector for a close integration between TPI and the ICS. For an example of such a setup, refer to the redbook *B2B Solutions using WebSphere Business Connection*, SG24-6197.

3.5.1 Creating business objects

The first step would be the creation of a business object matching the DTD that we used previously in WebSphere Data Interchange. You can use the Business Object Designer and define the fields manually. However, for a more realistic DTD representing a purchase order, there would be many more fields than what we use here. Defining the business object manually would then become an error-prone operation.

Tools are provided to make the definition of a business object easier. An optional installation component of the Interchange Server is the XMLODA, XML Object Discovery Agent. Launch the agent from the ODA\XML directory. When it is started, you should see a command window as shown in Figure 3-46 on page 151.



```
C:\WINNT\System32\cmd.exe

D:\CrossWorlds>set ORB1="D:\CrossWorlds"\lib\vbjorb.jar

D:\CrossWorlds>set DRIVERPATH="D:\CrossWorlds"\lib\xwutil.jar;"D:\CrossWorlds"\l
ib\xwbase.jar;"D:\CrossWorlds"\lib\xwsqlserver.jar;"D:\CrossWorlds"\lib\spy\lib\
spy.jar

D:\CrossWorlds>set JCLASSES="D:\CrossWorlds"\lib\rt.jar;;"D:\CrossWorlds"\lib\vb
jorb.jar;"D:\CrossWorlds"\lib\xerces.jar;"D:\CrossWorlds"\lib\jdom.jar;"D:\Cross
Worlds"\ODA\XML\XMLODA.jar;"D:\CrossWorlds"\lib\CwODK.jar;"D:\CrossWorlds"\lib\x
wutil.jar;"D:\CrossWorlds"\lib\xwbase.jar;"D:\CrossWorlds"\lib\xwsqlserver.jar;"
D:\CrossWorlds"\lib\spy\lib\spy.jar;"D:\CrossWorlds"\lib\CrossWorlds.jar

D:\CrossWorlds>"D:\CrossWorlds\bin\java" -Duser.home="D:\CrossWorlds" -mx128m -D
bea.home="D:\CrossWorlds" -DORBsyncGC=false -Dorg.omg.CORBA.ORBClass=com.inprise
.vbroker.orb.ORB -Dorg.omg.CORBA.ORBSingletonClass=com.inprise.vbroker.orb.ORBSi
ngleton -classpath "D:\CrossWorlds"\lib\rt.jar;;"D:\CrossWorlds"\lib\vbjorb.jar;
"D:\CrossWorlds"\lib\xerces.jar;"D:\CrossWorlds"\lib\jdom.jar;"D:\CrossWorlds"\O
DA\XML\XMLODA.jar;"D:\CrossWorlds"\lib\CwODK.jar;"D:\CrossWorlds"\lib\xwutil.jar
;"D:\CrossWorlds"\lib\xwbase.jar;"D:\CrossWorlds"\lib\xwsqlserver.jar;"D:\CrossW
orlds"\lib\spy\lib\spy.jar;"D:\CrossWorlds"\lib\CrossWorlds.jar com.crossworlds.
ODKInfrastructure.IdlAgentFactory -lXMLODAstudenta -ccom.crossworlds.oda.xml.XML
Agent
The "XMLODAstudenta" Object Discovery Agent is ready.
_
```

Figure 3-46 XML Object Discovery Agent is running

Now launch the Business Object Designer and select **File -> New Using ODA** from the menu, as shown in Figure 3-47.

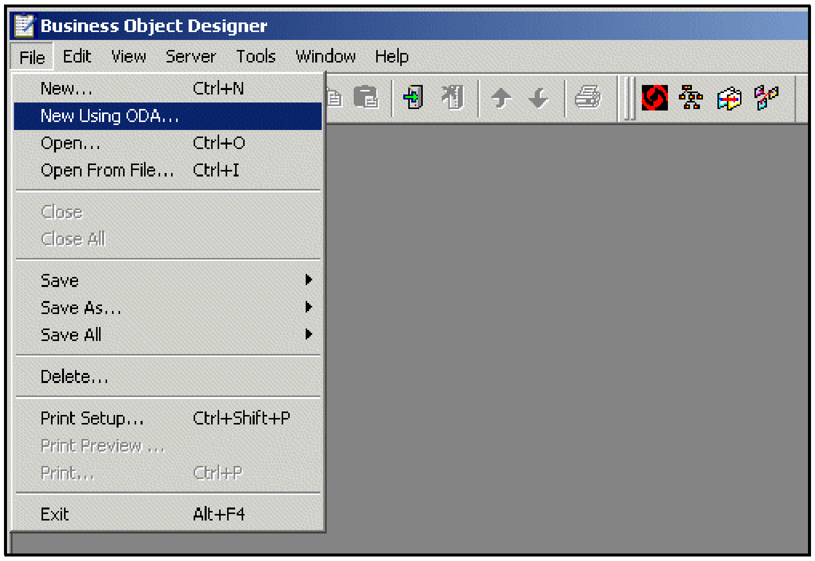
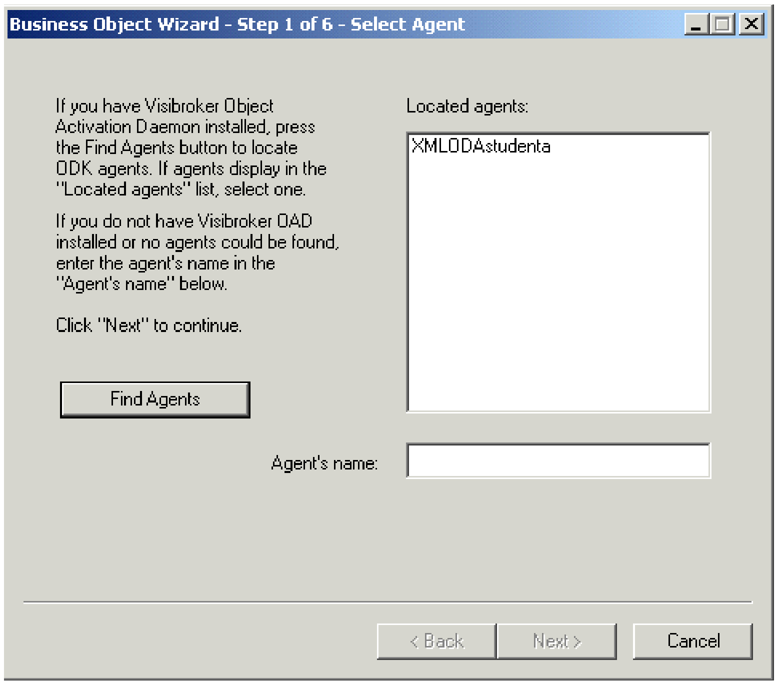


Figure 3-47 Using the Business Object Designer

A new window will appear to guide you through the definition process. Click the button **Find Agents** to populate the right pane with available agents and select the **XML ODA** agent from the list. Select **Next** to continue (Figure 3-48 on page 152).

Note: The Visibroker component should be running to get this list of available agents.



If you have Visibroker Object Activation Daemon installed, press the Find Agents button to locate the ODK agents. If agents display in the "Located agents" list, select one.

If you do not have Visibroker OAD installed or no agents could be found, enter the agent's name in the "Agent's name" below.

Click "Next" to continue.

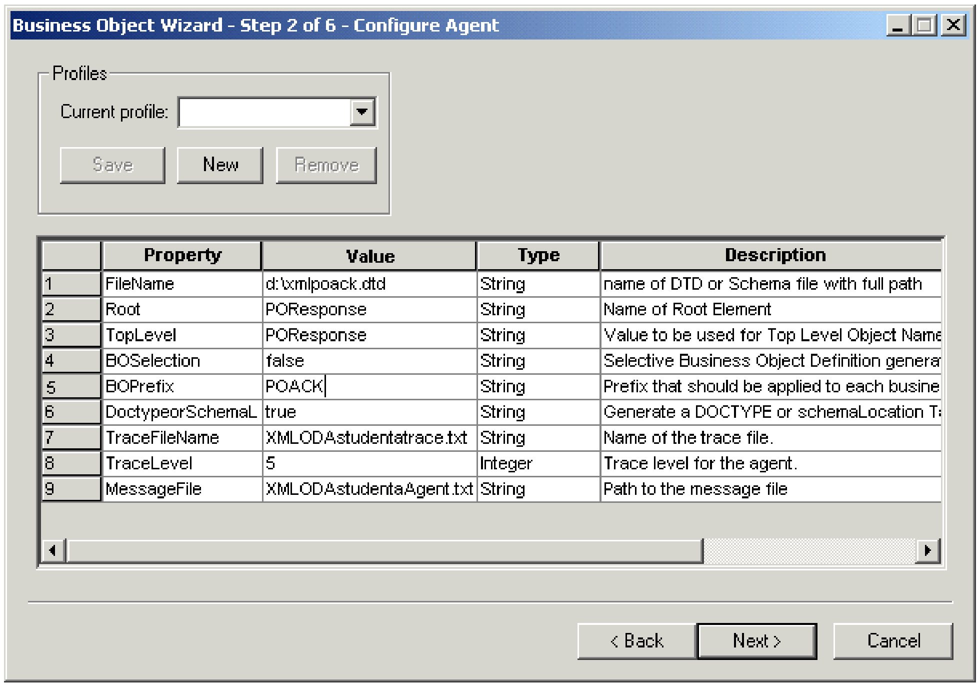
Located agents: XMLODAstudenta

Figure 3-48   Business Object wizard - Step 1

Most of the fields in Step 2 are populated by default. Provide the following information:

- Name of the file that contains the DTD
- Root element
- Top Level element
- BOPrefix

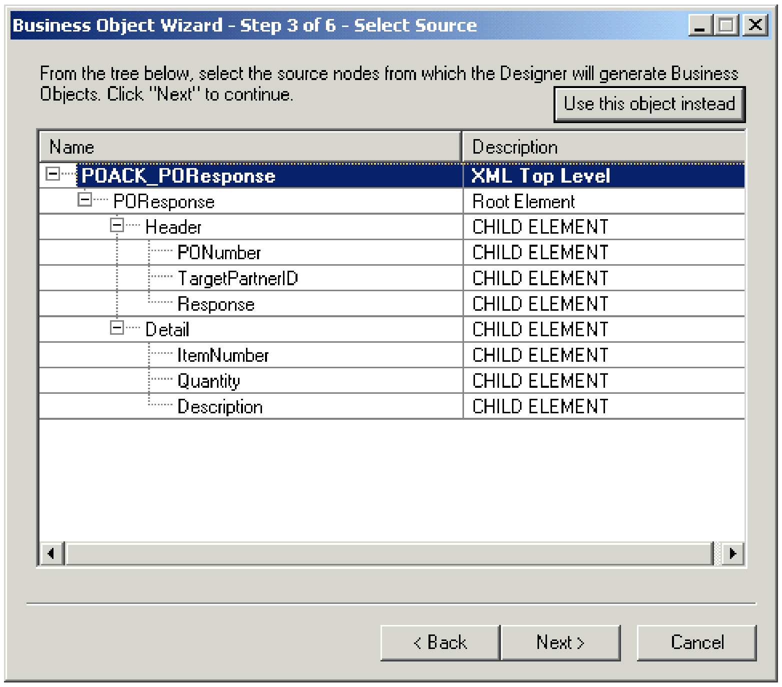
Then select **Next** to continue (Figure 3-49).



	Property	Value	Type	Description
1	FileName	d:\xmlpoack.dtd	String	name of DTD or Schema file with full path
2	Root	POResponse	String	Name of Root Element
3	TopLevel	POResponse	String	Value to be used for Top Level Object Name
4	BOSelection	false	String	Selective Business Object Definition genera
5	BOPrefix	POACK	String	Prefix that should be applied to each busine
6	DoctypeorSchemaL	true	String	Generate a DOCTYPE or schemaLocation T
7	TraceFileName	XMLODAstudentatrace.txt	String	Name of the trace file.
8	TraceLevel	5	Integer	Trace level for the agent.
9	MessageFile	XMLODAstudentaAgent.txt	String	Path to the message file

Figure 3-49   Business Object wizard - Step 2

The next step allows you to select other levels (or nodes) in the XML document for which you would like to create a business object definition. You might, for example, require an object to represent a single Detail element. For our purposes, this is not required. Therefore, we select the top node and click **Next** to continue (Figure 3-50).

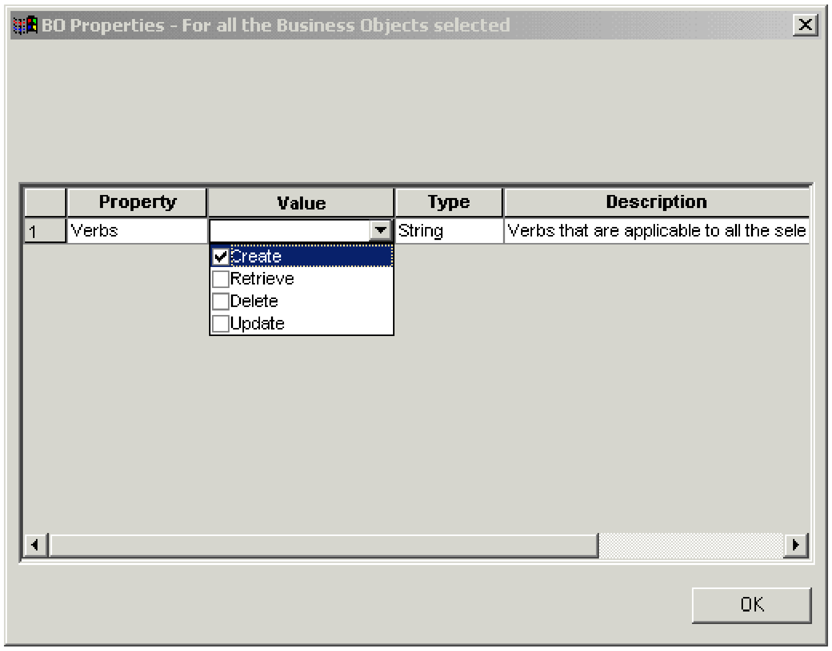

Business Object Wizard - Step 3 of 6 - Select Source

From the tree below, select the source nodes from which the Designer will generate Business Objects. Click "Next" to continue.

Name	Description
POACK_POResponse	XML Top Level
POResponse	Root Element
Header	CHILD ELEMENT
PONumber	CHILD ELEMENT
TargetPartnerID	CHILD ELEMENT
Response	CHILD ELEMENT
Detail	CHILD ELEMENT
ItemNumber	CHILD ELEMENT
Quantity	CHILD ELEMENT
Description	CHILD ELEMENT

Figure 3-50 Business Object wizard - Step 3

Step 4 summarizes your selections so far. At Step 5, you need to select a verb to go with the business objects. Figure 3-51 shows the selection of the **Create** verb. Click **OK** to continue.

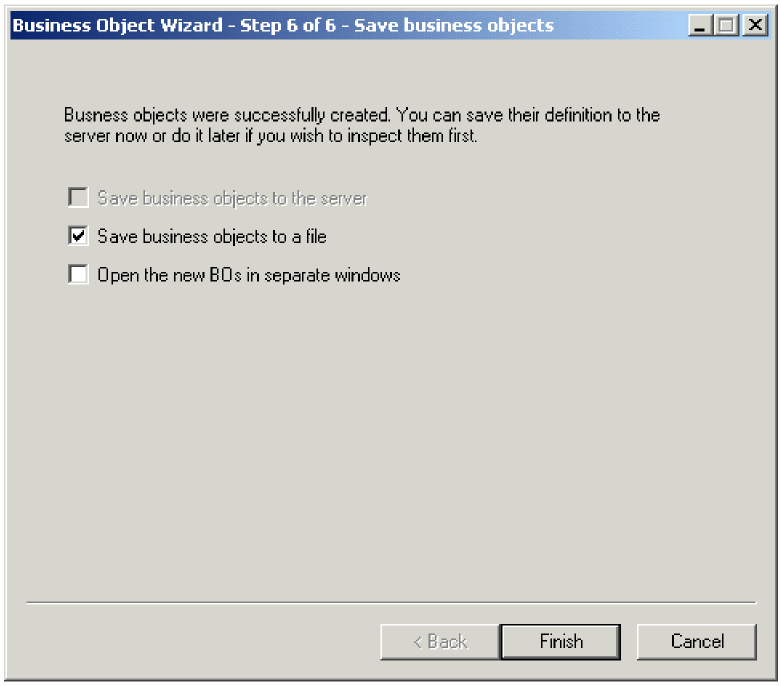

BO Properties - For all the Business Objects selected

	Property	Value	Type	Description
1	Verbs		String	Verbs that are applicable to all the sele

[✓] Create
[ ] Retrieve
[ ] Delete
[ ] Update

Figure 3-51 Business Object wizard - Step 5

Finally, in Step 6, you can choose where to save the business object. If the ICS is running and you are connected to it, the first option, *Save business objects to the server*, should be available. Alternatively, save the business object to an import file (selected option in Figure 3-52) and open the file later in the CrossWorlds System Manager by clicking **File -> Open from File**.


Business Object Wizard - Step 6 of 6 - Save business objects
< Back
Finish
Cancel

Busness objects were successfully created. You can save their definition to the server now or do it later if you wish to inspect them first.

☐ Save business objects to the server

☑ Save business objects to a file

☐ Open the new BOs in separate windows

Figure 3-52   Business Object wizard - Step 6

3.5.2 Configuring the MQSeries connector

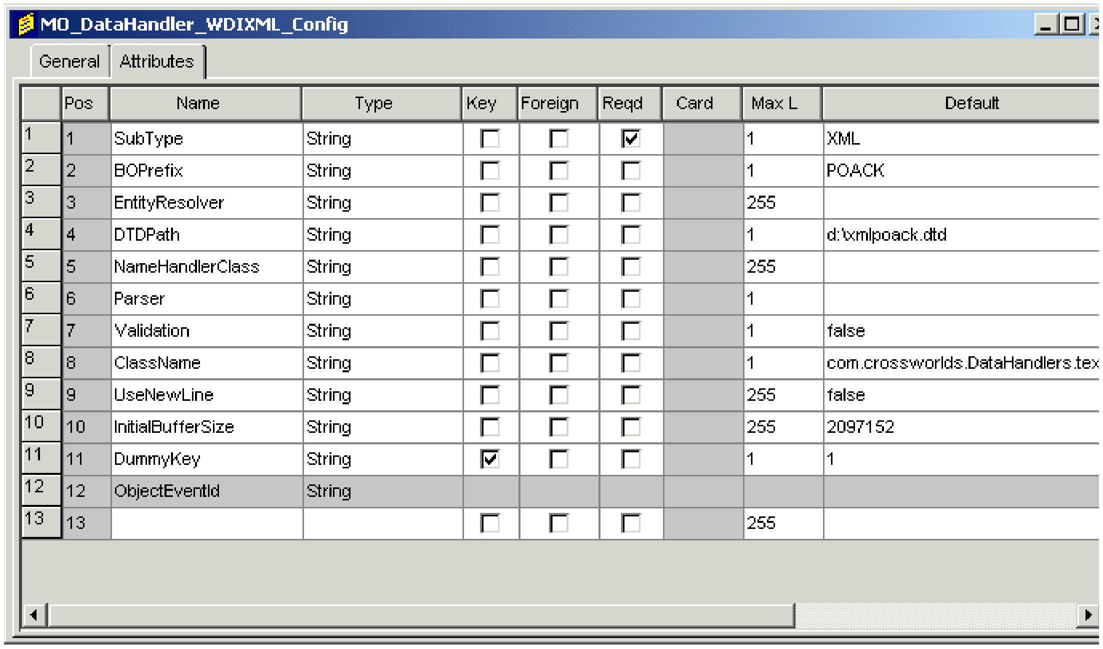
Depending on the version of ICS with which you are working, the MQSeries connector might also be called the WebSphere MQ connector, reflecting the name change of WebSphere MQ itself.

Updating the XML meta-object

Open the business object MO_DataHandler_DefaultXMLConfig and save it as MO_DataHandler_WDIXML_Config. Make the following changes to this business object:

- Set the attribute DTDPath to the directory that holds the DTD for the POResponse XML document.
- Set the BOPrefix to `POACK`, which is the prefix used during the creation of the business object (see Figure 3-49 on page 152).

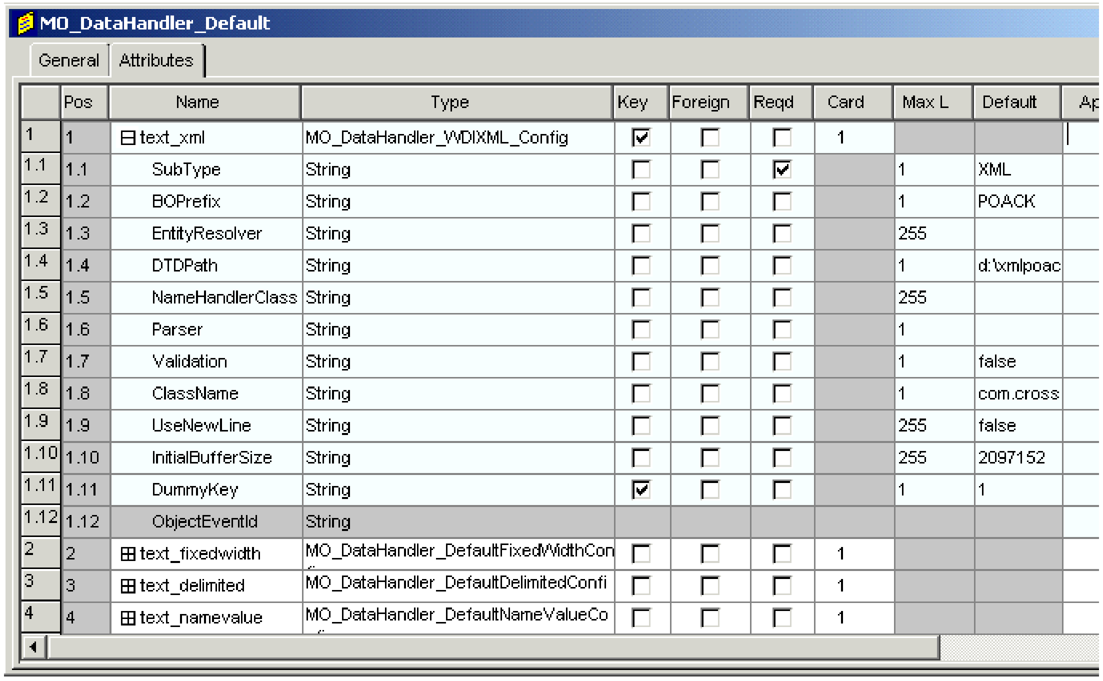
Figure 3-53 on page 155 shows the completed meta-object. Save the object to the server.



	Pos	Name	Type	Key	Foreign	Reqd	Card	Max L	Default
1	1	SubType	String	☐	☐	☑		1	XML
2	2	BOPrefix	String	☐	☐	☐		1	POACK
3	3	EntityResolver	String	☐	☐	☐		255	
4	4	DTDPath	String	☐	☐	☐		1	d:\xmlpoack.dtd
5	5	NameHandlerClass	String	☐	☐	☐		255	
6	6	Parser	String	☐	☐	☐		1	
7	7	Validation	String	☐	☐	☐		1	false
8	8	ClassName	String	☐	☐	☐		1	com.crossworlds.DataHandlers.tex
9	9	UseNewLine	String	☐	☐	☐		255	false
10	10	InitialBufferSize	String	☐	☐	☐		255	2097152
11	11	DummyKey	String	☑	☐	☐		1	1
12	12	ObjectEventId	String						
13	13			☐	☐	☐		255	

Figure 3-53 Data Handler business object

Now open the business object MO_DataHandler_Default. Update the Type field for element text_xml and set it to the XML Data Handler object `MO_DataHandler_WDIXML_Config` which we created before. Figure 3-54 shows the completed meta-object. Save this business object to the server.



	Pos	Name	Type	Key	Foreign	Reqd	Card	Max L	Default	Ap
1	1	⊟ text_xml	MO_DataHandler_WDIXML_Config	☑	☐	☐	1			
1.1	1.1	SubType	String	☐	☐	☑		1	XML	
1.2	1.2	BOPrefix	String	☐	☐	☐		1	POACK	
1.3	1.3	EntityResolver	String	☐	☐	☐		255		
1.4	1.4	DTDPath	String	☐	☐	☐		1	d:\xmlpoac	
1.5	1.5	NameHandlerClass	String	☐	☐	☐		255		
1.6	1.6	Parser	String	☐	☐	☐		1		
1.7	1.7	Validation	String	☐	☐	☐		1	false	
1.8	1.8	ClassName	String	☐	☐	☐		1	com.cross	
1.9	1.9	UseNewLine	String	☐	☐	☐		255	false	
1.10	1.10	InitialBufferSize	String	☐	☐	☐		255	2097152	
1.11	1.11	DummyKey	String	☑	☐	☐		1	1	
1.12	1.12	ObjectEventId	String							
2	2	⊞ text_fixedwidth	MO_DataHandler_DefaultFixedWidthCon	☐	☐	☐	1			
3	3	⊞ text_delimited	MO_DataHandler_DefaultDelimitedConfi	☐	☐	☐	1			
4	4	⊞ text_namevalue	MO_DataHandler_DefaultNameValueCo	☐	☐	☐	1			

Figure 3-54 Default Data Handler business object

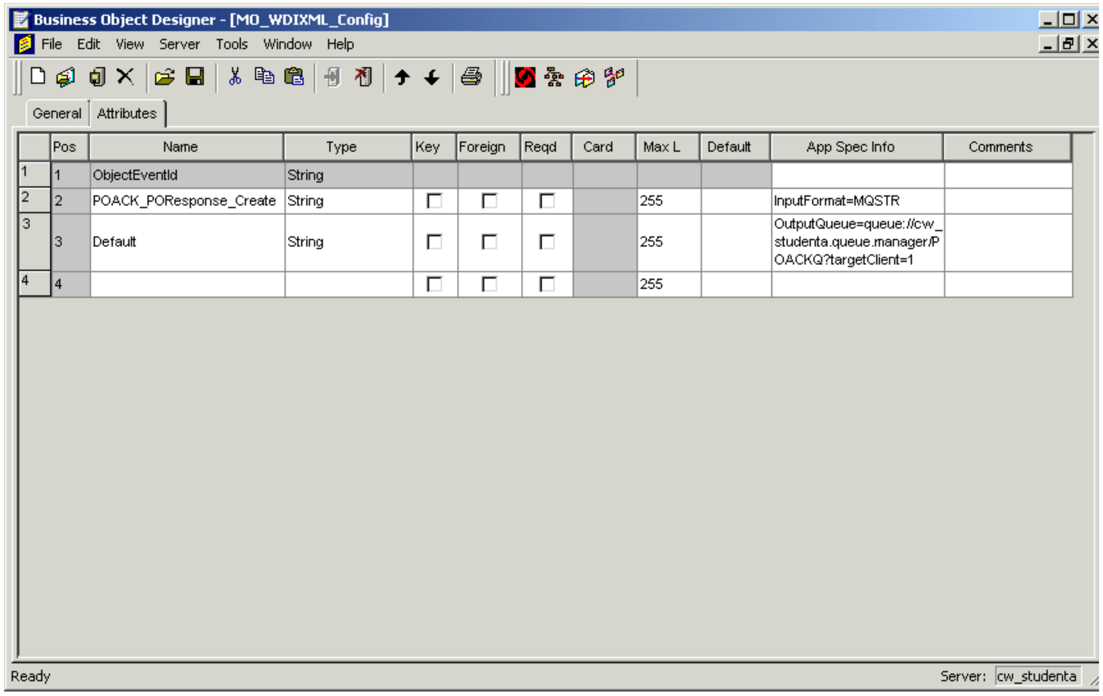
Defining the meta-object MO_WDIXML_config

The connector requires a meta-object that describes how to convert the business object to an XML message in a queue. Open the Business Object Designer and create a new business object, named MO_WDIXML_Config. When the Object Designer window appears, select the tab **Attributes** and make the following changes:

- In the name field, add `POACK_POResponse_Create`
- In the field App Spec Info, type `InputFormat=MQSTR`
- Add another attribute. In the name field, type `Default`
- Select the check box **Key** for this attribute
- In the field App Spec Info, type:

```
OutputQueue=queue://cw_studenta.queue.manager/POACKQ?targetClient=1
```

POACKQ is the name of the triggered queue for which WebSphere MQ will launch the WDIAdapter program, as configured in 3.4.2, “Preparing EDI documents” on page 142. Replace cw_studenta.queue.manager with the name of your queue manager. The option `targetClient=1` instructs the ICS to generate a standard WebSphere MQ message, instead of a JMS message. Figure 3-55 shows the completed meta-object.


Business Object Designer - [MO_WDIXML_Config]
File Edit View Server Tools Window Help
General Attributes
Ready
Server: cw_studenta

	Pos	Name	Type	Key	Foreign	Reqd	Card	Max L	Default	App Spec Info	Comments
1	1	ObjectEventId	String								
2	2	POACK_POResponse_Create	String	☐	☐	☐		255		InputFormat=MQSTR	
3	3	Default	String	☐	☐	☐		255		OutputQueue=queue://cw_studenta.queue.manager/POACKQ?targetClient=1	
4	4			☐	☐	☐		255			

Figure 3-55 Business object MO_WDIXML_Config

Configuring the MQSeries connector

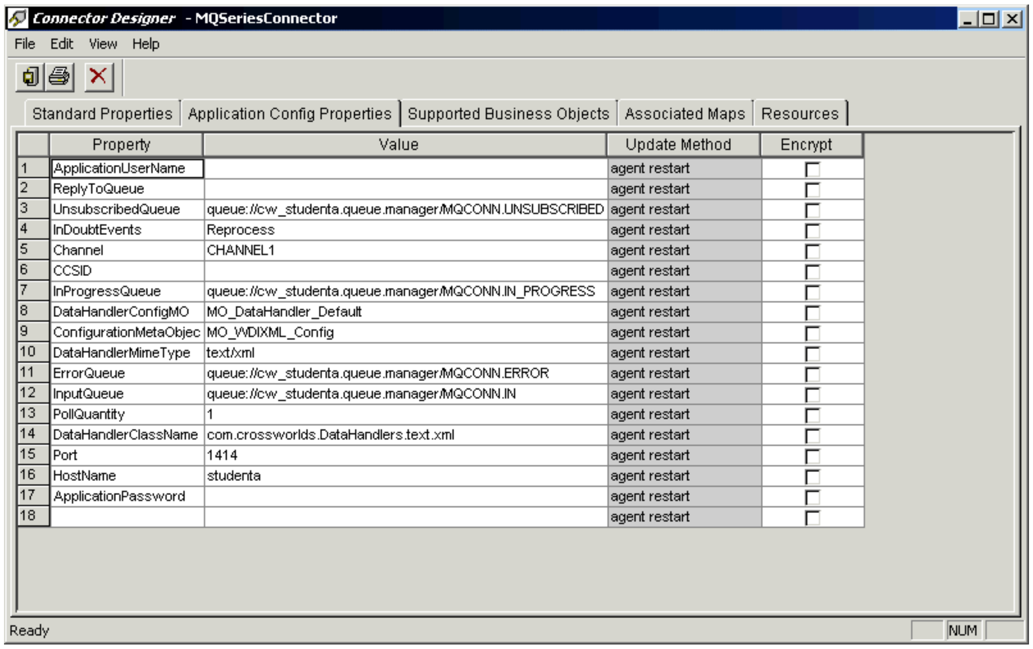
Expand the folder Connectors in the System Manager and double-click the object **MQSeries Connector**. Click the tab **Connector Agent** to specify the connector-specific properties, as detailed in Table 3-1.

Table 3-1 Connector properties

Property	Value
InDoubtEvents	Reprocess
Channel	CHANNEL1
InProgressQueue	queue://cw_studenta.queue.manager/MQ-CONN.IN_PROGRESS

Property	Value
DataHandlerConfigMO	MO_DataHandler_Default
ConfigurationMetaObject	MO_WDIXML_Config
DataHandlerMimeType	text/xml
Port	1414
Hostname	studenta

Figure 3-56 shows the Connector Designer window where you need to specify the values listed in Table 3-1 on page 156.


Connector Designer - MQSeriesConnector
File
Edit
View
Help
Standard Properties
Application Config Properties
Supported Business Objects
Associated Maps
Resources
Ready
NUM

	Property	Value	Update Method	Encrypt
1	ApplicationUserName		agent restart	☐
2	ReplyToQueue		agent restart	☐
3	UnsubscribedQueue	queue://cw_studenta.queue.manager/MQCONN.UNSUBSCRIBED	agent restart	☐
4	InDoubtEvents	Reprocess	agent restart	☐
5	Channel	CHANNEL1	agent restart	☐
6	CCSID		agent restart	☐
7	InProgressQueue	queue://cw_studenta.queue.manager/MQCONN.IN_PROGRESS	agent restart	☐
8	DataHandlerConfigMO	MO_DataHandler_Default	agent restart	☐
9	ConfigurationMetaObjec	MO_WDIXML_Config	agent restart	☐
10	DataHandlerMimeType	text/xml	agent restart	☐
11	ErrorQueue	queue://cw_studenta.queue.manager/MQCONN.ERROR	agent restart	☐
12	InputQueue	queue://cw_studenta.queue.manager/MQCONN.IN	agent restart	☐
13	PollQuantity	1	agent restart	☐
14	DataHandlerClassName	com.crossworlds.DataHandlers.text.xml	agent restart	☐
15	Port	1414	agent restart	☐
16	HostName	studenta	agent restart	☐
17	ApplicationPassword		agent restart	☐
18			agent restart	☐

Figure 3-56 Configuring the MQSeries connector

Select the tab **Supported Business Objects**. Click the blank cell under the heading Business Object Name. A drop-down box will appear. Select **POACK_POResponse** from the list and select the check box **Agent Support**. Also add the meta-object MO_DataHandler_default to this table and select the check box **Agent Support** again.

When finished, select **File -> Save to Server**. During the save, you may receive warnings about the need to restart the connector. You can accept those warnings. When the save process is finished, switch to the System Manager and right-click the **MQSeries** connector in the folder Connectors; stop and restart the connector.

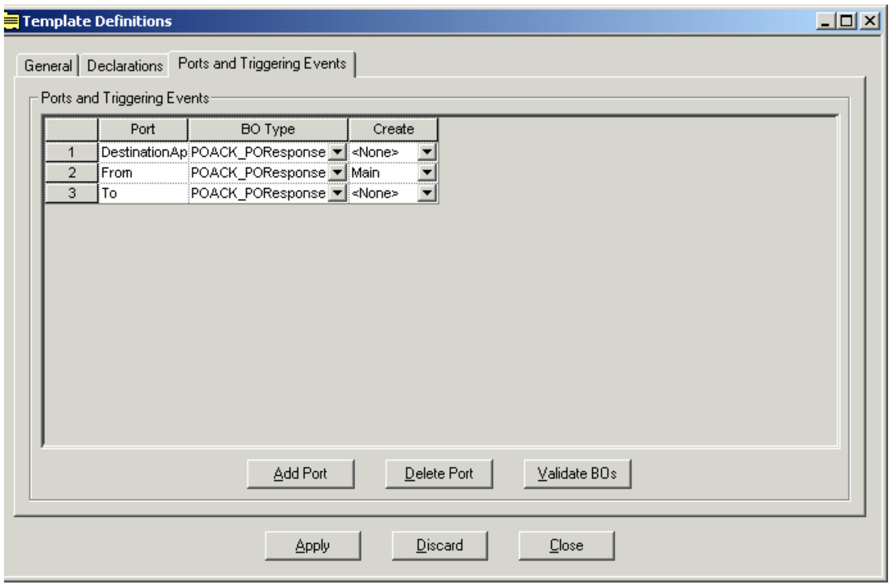
3.5.3 Developing a test collaboration

The next step is the development of a collaboration that will generate the POResponse document for processing by WebSphere Data Interchange. Open the CrossWorlds System Manager and expand the folder Collaboration Templates. Locate the template CollaborationFoundation and copy and paste it in the folder Collaboration Templates.

Note: If your installation of the Interchange Server does not have this template, you can find an import file for this template as part of the additional material for this redbook. Refer to Appendix B, "Additional material" on page 217.

Name the copied template WDI_Outbound_Template. Open the new template in the Process Designer by double-clicking it. Select **Template -> Open Template Definitions** to update the template. Select the tab **Ports and Triggering Events**. Update the BO Name for each port and set it to `POACK_POResponse`. Change the field Create for the From row (2) to `Main` (see Figure 3-57).

Apply the changes and compile the updated template.You can delete the port DestinationAppRetrieve, but that is not required.



	Port	BO Type	Create
1	DestinationAp	POACK_POResponse	<None>
2	From	POACK_POResponse	Main
3	To	POACK_POResponse	<None>

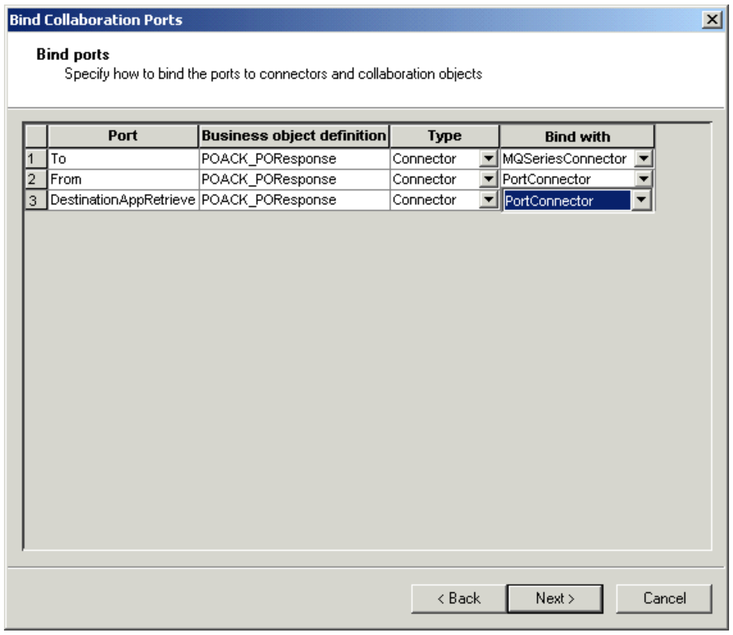
Figure 3-57 Update the template definitions

Now that we have a template that fits our needs, we can create a collaboration object. Right-click the folder **Collaborations** and select **New collaboration object** from the context menu. Select the template **WDI_Outbound_Template** and name the new collaboration WDI_Outbound. Click **Next**. Now bind the ports to the connectors, as shown in Figure 3-58 on page 159.

Set the From port to `PortConnector` and the To port to `MQSeriesConnector`. Also set the DestinationAppRetriever port to `PortConnector`, if you have not deleted this port in the template.

Note: If you do not see the PortConnector as an available choice, you need to update the list of supported business objects for the PortConnector and include the business object POACK_POResponse.

Click **Next** twice, then click **Finish** to complete the definition of this collaboration.


Bind Collaboration Ports

Bind ports
Specify how to bind the ports to connectors and collaboration objects

	Port	Business object definition	Type	Bind with
1	To	POACK_POResponse	Connector	MQSeriesConnector
2	From	POACK_POResponse	Connector	PortConnector
3	DestinationAppRetrieve	POACK_POResponse	Connector	PortConnector

Figure 3-58 Bind the ports of the collaboration

If all steps were performed without problems, the System Manager will now show a graphical representation of the collaboration, as shown in Figure 3-59.

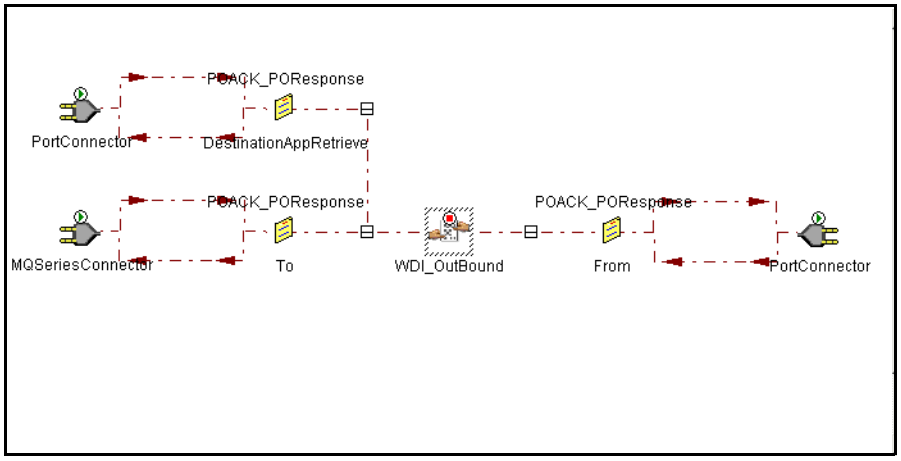


Figure 3-59 WDI_Outbound collaboration

Finally, start the collaboration by selecting **Component -> Start WDI_Outbound**. Or, select the `start` command from the context menu of the collaboration.

3.5.4 Using the Test Connector

The final step is to use the Test Connector tool to drive the collaboration. Before you start this tool, you should perform a number of validation tasks.

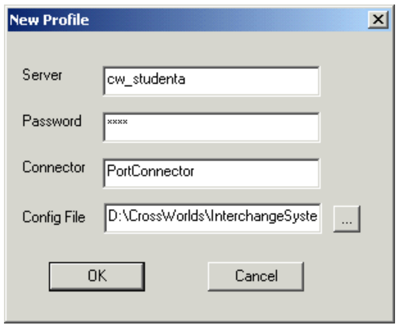
- Expand the folder Connectors in the System Manager and verify that the MQSeriesConnector is started.
- Verify that the PortConnector is started. Sometimes this connector is in a paused state, which is usually caused by a missing queue in WebSphere MQ. If you cannot get the PortConnector to start, verify that the queue AP/PORTCONNECTOR/CW_STUDENTA

exists. Replace CW_STUDENTA with the name of your InterChange Server. If this queue does not exist, create it with default attributes.

- Start the MQSeries Connector that we configured previously by selecting **Start -> Programs -> IBM CrossWorlds -> Connectors -> MQSeries Connector**. Verify in the output that the connector has started correctly. You can also start the System Monitor (from the Tools menu in the CrossWorlds System Manager) and verify that the agent state of the MQSeries Connector is Active.

Now start the Test Connector by selecting **Start -> Programs -> IBM CrossWorlds -> Connectors -> Test Connector**. When the tool is started, select **File -> New Profile**, which will bring up a profile selection window. Select **Add** to create a new profile.

Provide the name of the server, the password of the admin user ID (usually the word `null`) and the name of the connector that we are going to simulate: `PortConnector`. You can leave the field Config File blank. The test connector should be able to locate the configuration file by itself. Alternatively, you should point to the ICS configuration file manually (usually D:\CrossWorlds\InterchangeSystem.cfg). Click **OK** to close this window.

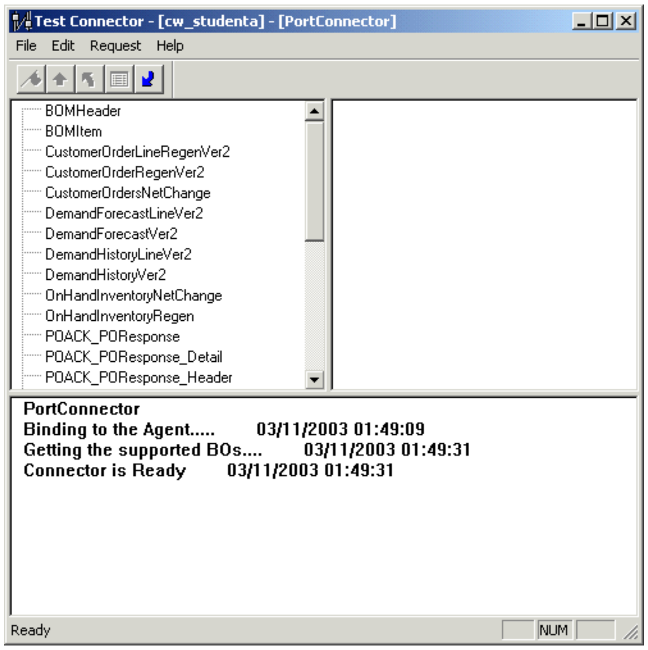


Server	cw_studenta
Password	xxxx
Connector	PortConnector
Config File	D:\CrossWorlds\InterchangeSyste

Figure 3-60 Create a new profile for the test connector

Select the new profile in the profile list window and click **OK**. The test connector is now loaded with the correct profile. Select **File -> Connect Agent** to connect to the ICS.

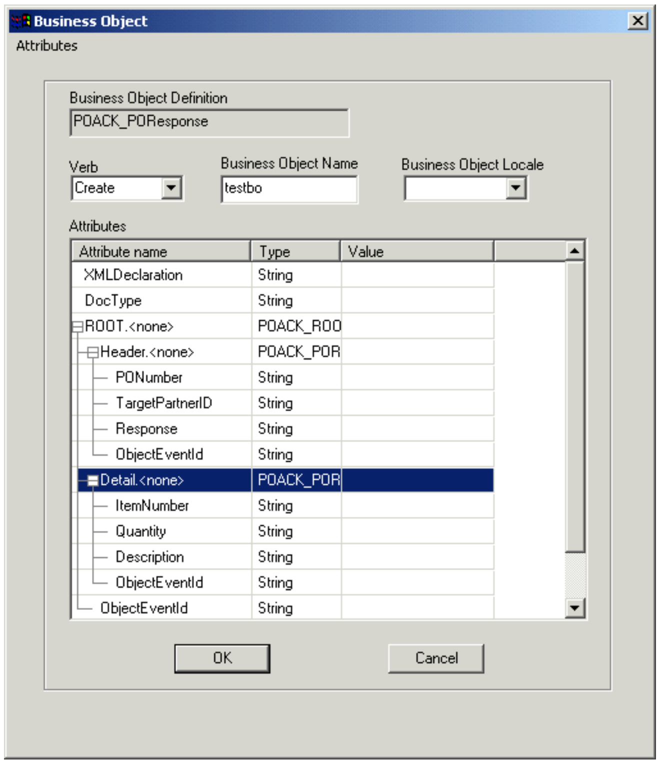
When the connection succeeds, the test connector will list all business objects that are supported by the port connector, as shown in Figure 3-61 on page 161.



```
PortConnector
Binding to the Agent.....        03/11/2003 01:49:09
Getting the supported BOs....        03/11/2003 01:49:31
Connector is Ready        03/11/2003 01:49:31
```

Figure 3-61   Running the test connector

Double-click the business object **POACK_POResponse** to bring up the next window. Select the verb **Create** and name the instance of the business object (for example `testbo`). Right-click the element **ROOT** and select **Add Instance**. Now expand the ROOT element which will list two child elements, Header and Detail. Right-click these elements too and select **Add Instance** each time. The business object window should now look as in Figure 3-62.



Business Object Definition: POACK_POResponse

Verb: Create   Business Object Name: testbo   Business Object Locale:

Attributes

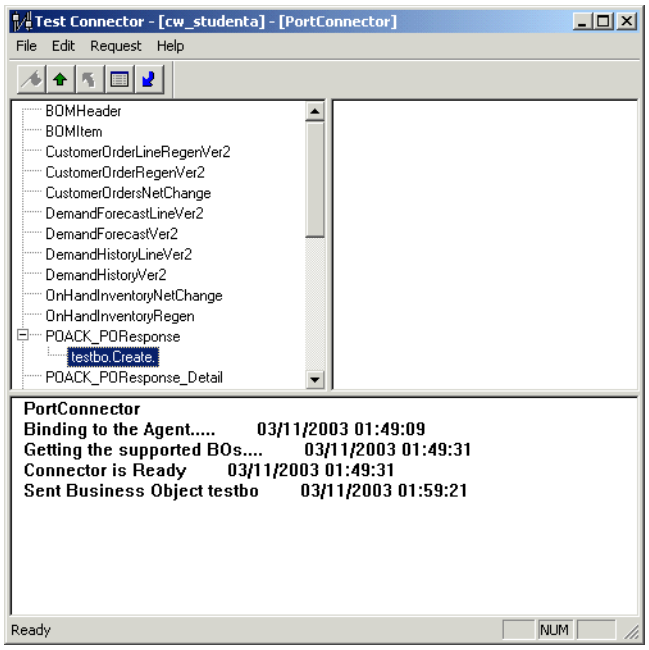
Attribute name	Type	Value
XMLDeclaration	String	
DocType	String	
ROOT.<none>	POACK_ROO	
Header.<none>	POACK_POR	
PONumber	String	
TargetPartnerID	String	
Response	String	
ObjectEventId	String	
Detail.<none>	POACK_POR	
ItemNumber	String	
Quantity	String	
Description	String	
ObjectEventId	String	
ObjectEventId	String	

Figure 3-62   Setting values for the business object

Provide values for the six data elements of this business object and click **OK**.

Important: Since this business object will result in a message that will be processed by WebSphere Data Interchange, you should provide data that makes sense for WebSphere Data Interchange. Setting a random value for TargetPartnerID will likely result in an unprocessed document within WebSphere Data Interchange.

This will bring you back to the main window of the test connector. Select the new business object in the tree structure and click **Request -> Send**.



```
PortConnector
Binding to the Agent.....       03/11/2003 01:49:09
Getting the supported BOs....       03/11/2003 01:49:31
Connector is Ready       03/11/2003 01:49:31
Sent Business Object testbo       03/11/2003 01:59:21
```

Figure 3-63 Business object being sent

Open WebSphere MQ Explorer and browse the queue POACKQ, which should contain an XML message representing this business object. However, if the WebSphere MQ Trigger Monitor is still running, your message might be consumed already and you may need to inspect the output queues of WebSphere Data Interchange. And, if TPI Server is still running, your message might be gone to a trading partner.

This completes the basic integration of the InterChange Server in a solution with WebSphere Data Interchange. The collaboration can now be extended to include ports to real back-office applications and at this time, you will probably need to develop some maps to map business objects.

3.5.5 Inbound flow

The previous sections described in detail the integration process for the outbound flow. The steps to integrate CrossWorlds in the inbound flow are quite similar.

Business object

First, we again need a business object to represent the incoming purchase order. The DTD, listed in Example 3-11 on page 135, can be imported in the InterChange Server using the XML ODA as described in 3.5.1, “Creating business objects” on page 150.

Specify the following values for the PO DTD:

- Root element: `PO`
- Top Level element: `PO`
- BOPrefix: `PO`

The MQSeriesInbound Connector

The next step is to create an additional MQSeries Connector. Perform the following steps:

1. Copy and paste the existing MQSeriesConnector object in the folder Connectors in the CrossWorlds System Manager. Name it MQSeriesInboundConnector.
2. Open a file browser and find the directory MQSeries in the connectors directory of the CrossWorlds installation. Copy the whole directory and name it MQSeriesInbound.
3. Open the folder Connectors in the Start menu and copy and paste the existing shortcut MQSeries Connector as MQSeriesInbound Connector.
4. Open the properties of this new short-cut and update the field Target:

   ```
   D:\CrossWorlds\connectors\MQSeriesInbound\start_MQSeries.bat MQSeriesInbound cw_studenta
   -cD:\CrossWorlds\connectors\MQSeriesInbound\MQSeriesAgentConfig.cfg
   ```

   MQSeries has been replaced three times with MQSeriesInbound. Also update the field *Start in* to the name of the new directory:

   ```
   D:\CrossWorlds\connectors\MQSeriesInbound\
   ```

5. Define a new queue AP/MQSERIESINBOUNDCONNECTOR/CW_STUDENTA on the queue manager used by the ICS. Replace CW_STUDENTA with the name of your ICS.
6. Restart the ICS. After the restart, verify that the connector is running using the CrossWorlds System Manager.
7. Start the connector agent using the shortcut in the Programs folder and verify that the Agent State in the System Monitor is Active.

Open the PortConnector object and update the supported business objects. Include business object PO_PO in the list and be sure to check the field Agent Support.

Creating meta-objects

Once you have verified that the new connector can be started, proceed with the definition of meta-objects. Open the meta-object MO_DataHandler_WDIXML_Config and save it as MO_DataHandler_CWXML_Config. Make the following changes:

- Provide the path to the location of the DTD and the file name
- Set the BOPrefix to `PO`

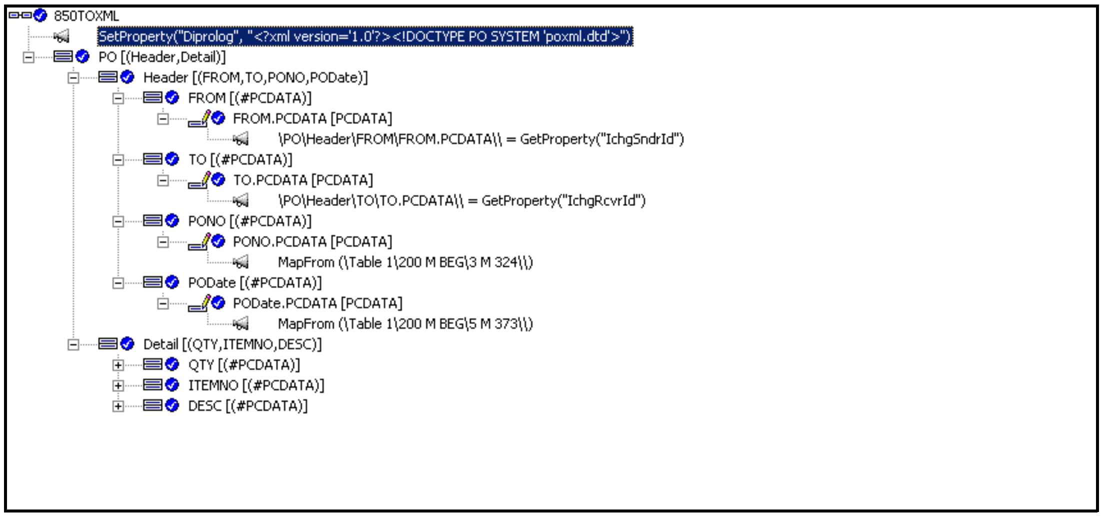
Define meta-object MO_CWXML_Config. You can copy the meta-object MO_WDIXML_Config. Rename the field POACK_POResponse to PO_PO.

Define meta-object MO_DataHandler_Inbound_Default. You can copy it from MO_DataHandler_Default. For the MIME type text/xml, set the field type to `MO_DataHandler_CWXML_Config`.

Open the connector object MQSeriesInboundConnector and switch to the tab Application Config Properties. Set the value of the property DataHandlerConfigMO to `MO_DataHandler_Inbound_Default`. Set the value of the property ConfigurationMetaObject to `MO_CWXML_Config`. Set the value of the property InputQueue to `queue://cw_studenta.queue.manager/purchase.orders`. Save the changes and restart the connector.

Verifying the map in WebSphere Data Interchange

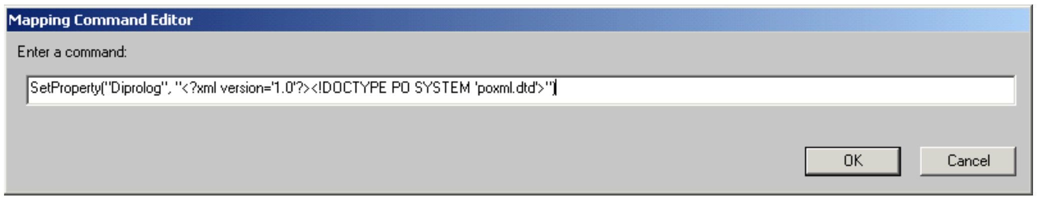
The ICS requires that the incoming XML message contain a DOCTYPE statement that includes the name of the DTD. To make sure that the XML document contains the DTD name, review the map 850TOXML in WebSphere Data Interchange. Open the map editor and verify that you have a SetProperty call for the property Diprolog, as shown in Figure 3-64.



```
850TOXML
    SetProperty("Diprolog", "<?xml version='1.0'?><!DOCTYPE PO SYSTEM 'poxml.dtd'>")
  PO [(Header,Detail)]
    Header [(FROM,TO,PONO,PODate)]
      FROM [(#PCDATA)]
        FROM.PCDATA [PCDATA]
          \PO\Header\FROM\FROM.PCDATA\\ = GetProperty("IchgSndrId")
      TO [(#PCDATA)]
        TO.PCDATA [PCDATA]
          \PO\Header\TO\TO.PCDATA\\ = GetProperty("IchgRcvrId")
      PONO [(#PCDATA)]
        PONO.PCDATA [PCDATA]
          MapFrom (\Table 1\200 M BEG\3 M 324\\)
      PODate [(#PCDATA)]
        PODate.PCDATA [PCDATA]
          MapFrom (\Table 1\200 M BEG\5 M 373\\)
    Detail [(QTY,ITEMNO,DESC)]
      QTY [(#PCDATA)]
      ITEMNO [(#PCDATA)]
      DESC [(#PCDATA)]
```

Figure 3-64 Setting the property Diprolog

If this statement does not exist, right-click the name of the map **850TOXML** and select **Insert -> Command -> SetProperty** from the context menu. A mapping command editor should appear, as shown in Figure 3-65. Update the template call of SetProperty to refer to the property Diprolog and set the value to what is required for your XML document.



Mapping Command Editor

Enter a command:

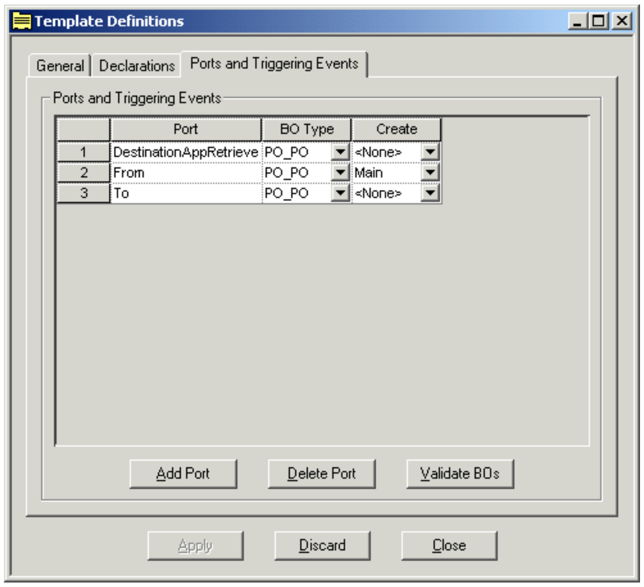
```
SetProperty("Diprolog", "<?xml version='1.0'?><!DOCTYPE PO SYSTEM 'poxml.dtd'>")
```

Figure 3-65 Adding the name of the DTD to the XML document

Save and re-compile the map.

Creating the collaboration

Expand the folder Collaboration Templates in CrossWorlds System Manager. Copy and paste the template CollaborationFoundation and name the new template WDI_Inbound_Template. Open the new template and its definitions. Select the tab **Ports and Triggering Events** (see Figure 3-66 on page 165). Set the BOType to `PO_PO` for all three ports. Set the Create field for the From port to `Main`. Apply the changes. Compile and save the template.



	Port	BO Type	Create
1	DestinationAppRetrieve	PO_PO	<None>
2	From	PO_PO	Main
3	To	PO_PO	<None>

Figure 3-66 Template definitions

Now create a new collaboration WDI_Inbound from the template WDI_Inbound_Template. Bind the ports as follows:

- From port: `MQSeriesInboundConnector`
- To port: `PortConnector`
- DestinationAppRetrieve port: `PortConnector`

Save and start the collaboration. Check the server log and verify that you see log messages, as shown in Example 3-22.

Example 3-22 ICS log

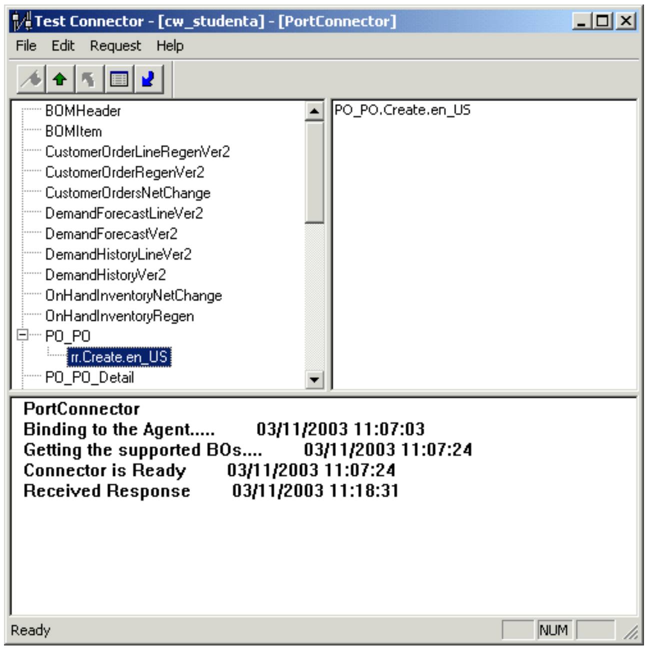
```
[System: Server] [Thread: VBJ ThreadPool Worker (#5244814)] [Type: Info] [MsgID: 31] [Mesg: Initializing
collaboration "WDI_Inbound".]
[System: Collaboration] [SS: WDI_Inbound] [Thread: VBJ ThreadPool Worker (#4968337)] [Type: Info] [MsgID:
11009] [Mesg: Subscribed to PO_PO.Create from publisher MQSeriesInboundConnector.]
[System: Collaboration] [SS: WDI_Inbound] [Thread: VBJ ThreadPool Worker (#4968337)] [Type: Info] [MsgID:
11014] [Mesg: Collaboration is active.]
```

Using the test connector

Finally, start the test connector and select the profile **PortConnector**. Connect to the server and create a business object of type PO_PO. Select the newly created business object and select **Request -> Accept Request**. The PortConnector now acts as an endpoint and is ready to receive PO_PO business objects.

Write an EDI message on the queue INBOX, processed by WebSphere Data Interchange. The translated message, including a DOCTYPE, should then end up on queue purchase.orders, which is monitored by the MQSeriesInbound Connector. The message will then be routed through the collaboration and end up at the port connector in the test connector.

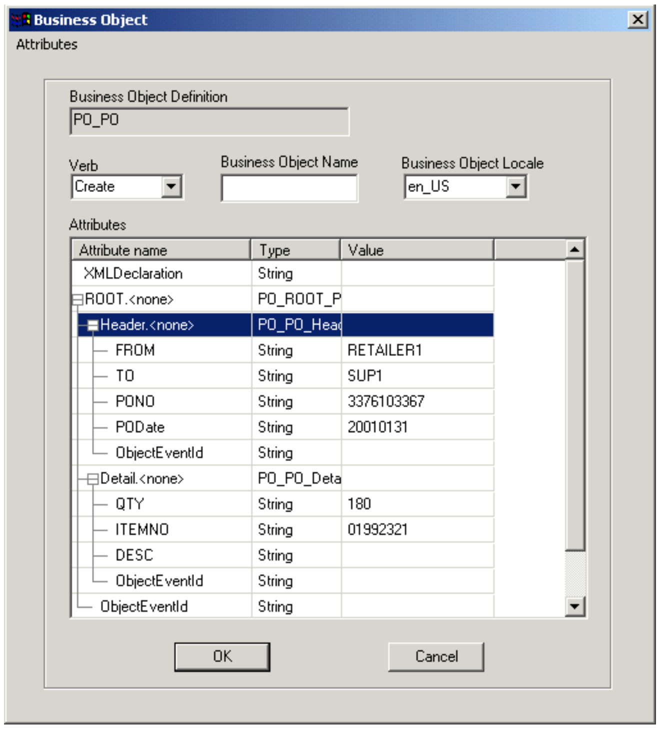
Figure 3-67 on page 166 shows the test connector window when data has arrived.



```
PortConnector
Binding to the Agent.....        03/11/2003 11:07:03
Getting the supported BOs....       03/11/2003 11:07:24
Connector is Ready        03/11/2003 11:07:24
Received Response         03/11/2003 11:18:31
```

Figure 3-67 A business object has arrived

Select the business object in the right pane to inspect the details. Figure 3-68 shows the business object representation of this XML message, which was a translation of an EDI 850 document.



Business Object Definition: PO_PO

Verb: Create

Business Object Name:

Business Object Locale: en_US

Attributes

Attribute name	Type	Value
XMLDeclaration	String	
ROOT.<none>	PO_ROOT_P	
Header.<none>	PO_PO_Hea	
FROM	String	RETAILER1
TO	String	SUP1
PONO	String	3376103367
PODate	String	20010131
ObjectEventId	String	
Detail.<none>	PO_PO_Deta	
QTY	String	180
ITEMNO	String	01992321
DESC	String	
ObjectEventId	String	
ObjectEventId	String	

Figure 3-68 Details of the received business object

4

UCCnet and item synchronization via iSoft and TPI

The previous chapter demonstrated that we can combine TPI and iSoft AS/2 providers. Companies using one of those two products can interact with each other using AS/2. This chapter focuses on interoperability between the two AS/2 providers and existing integration solutions.

As an example integration solution, we use the UCCnet Item Sync collaboration. This collaboration is usually implemented on top of the Interchange Server and interacts with UCCnet via TPI. ICS and TPI interact via the TPI connector.

In two steps, we demonstrate that connectors can be changed, from a TPI connector to an MQ connector, without impacting the overall solution. We also demonstrate that the Item Sync solution can work over iSoft's P2PAgent as well.

4.1 Overview of UCCnet

UCCnet, a not-for-profit, tax-exempt organization, aims to synchronize item information among all trading partners within the 23 industries served by the Uniform Code Council, Inc. (UCC). The mission of UCCnet is to provide value by enabling the formation of collaborative relationships through an electronic trading capability, allowing trading partners served by the UCC to have synchronized EAN·UCC item information, as well as access to compliant business applications and services.

UCCnet provides a universal foundation for electronic commerce, ensuring the integrity of fast, clean business transactions across the Internet, backed with the trustmark of the UCC.

Through its support of the global EAN•UCC standards, UCCnet delivers an open platform for collaborative commerce services, including standards-compliance verification, synchronization of product information, registry and lifecycle management of synchronized products, user locations and user trade capabilities.

UCCnet has been developed to provide an essential backbone that links trade exchange members, peer-to-peer partners, industry-driven e-marketplaces and alliance partners to one open, seamless repository of information.

As a subsidiary of the UCC, the global standards organization, UCCnet leverages its position to enable the rapid, clear development of industry Internet standards and practices.

In addition, UCCnet provides an implementation methodology, allowing small, medium and large enterprises to integrate e-commerce into their business in a swift and efficient manner.

4.2 The IBM solution

WebSphere Business Integration for Retail Distribution is a portfolio that provides retailers and their suppliers with industry-specific business process templates, a portfolio of connectors to leading applications, and an integration hub to help them automate their current business processes.

This overview focuses on the portfolio’s first solution offering, the collaboration for UCCnet Item Synchronization for Suppliers, which automatically integrates a supplier’s item business processes with those of its trading partners. The process solution uses the IBM WebSphere BI Collaboration for UCCnet Item Synchronization, along with application connectors, the IBM InterChange Server (ICS) and the Trading Partner Interchange (TPI) connector and server to provide affordable, extensible, scalable, and secure access to the UCCnet standard registry. The resulting efficiencies to supply chains can significantly drive down costs and improve profits for suppliers.

The Item Synchronization for Suppliers process solution meets these challenges by helping suppliers automate the item synchronization process in compliance with UCCnet standards. In fact, the solution is certified for interoperability with UCCnet V2.0 and V2.1.

The collaboration, or pre-built template, enables suppliers to automatically add items to, update or delist items within, or withdraw items from UCCnet when item updates are made in their Enterprise Resource Planning (ERP) applications. When an item is updated in a supplier’s ERP system, item data is automatically validated, reformatted, and sent to the UCCnet standard registry. This collaboration also provides a single process for communicating item information to trading partners via UCCnet. Thus, a supplier’s enterprise data is synchronized with item data sent outside the enterprise.

Figure 4-1 and the description following it show high-level components of the IBM WebSphere BI Collaboration for UCCnet Item Synchronization and how one of the possible business scenarios (the publication of a new item) is played out.

In this ItemAdd/ItemChange scenario, new item information is passed to UCCnet. The source of the flow is the creation of a new item (it could also be a change to an existing item) in the source ERP application. The end result of the flow processing is an ItemAdd (or an ItemChange) message that is received by UCCnet through the TPI connector.

The numbered steps in the graphic correspond to the description that follows.

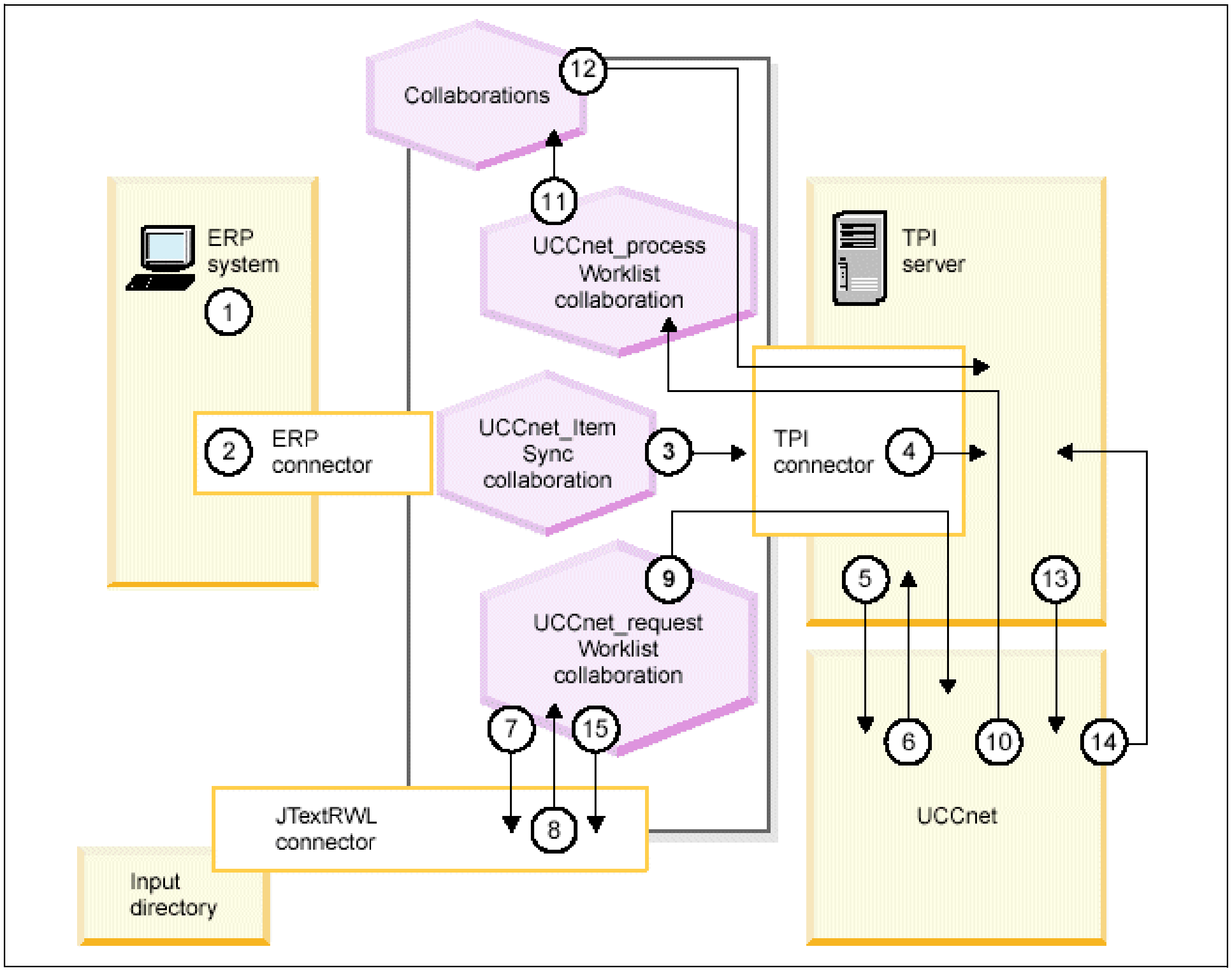


Figure 4-1 The Item Sync collaboration for UCCnet

1. A trigger from the ERP source provides the item (for example, an IDOC from SAP) to the WebSphere BI ERP-specific connector.
2. The connector transforms the data into an application-specific business object, initiates mapping from the application-specific business object to the generic business object ItemBasic and then passes the ItemBasic business object to the UCCnet_ItemSync collaboration.
3. The UCCnet_ItemSync collaboration delivers the ItemBasic business object to the TPI connector.

4. The TPI connector initiates mapping from the generic ItemBasic business object to the application-specific UCCnet_envelope business object, builds a UCCnet ItemAdd XML document from the UCCnet_envelope business object, and sends the ItemAdd document to the TPI Server.
5. The TPI Server uses the trading partner profile for UCCnet, creates the digest, encrypts, and transmits the ItemAdd document to UCCnet.
6. The Message Disposition Notification (MDN) is generated by UCCnet and returned to the TPI Server.
7. The UCCnet_requestWorklist collaboration uses the JTextRWL sample connector to poll a worklist directory on the InterChange Server for an XML message containing a response Item_Add notification from UCCnet.
8. When the JTextRWL sample connector finds an XML file in its input folder for events, it transforms and maps the message to a generic UCCnetGBO_envelope business object, and delivers the business object to the UCCnet_requestWorklist collaboration.
9. The UCCnet_requestWorklist collaboration delivers the business object to the TPI connector. The TPI connector initiates mapping from the generic UCCnetGBO_envelope sample business object to the application-specific UCCnet_envelope business object, builds an XML document from the UCCnet_envelope business object, and sends the XML document to the TPI Server. The TPI Server uses the trading partner profile for UCCnet, creates the digest, encrypts, and transmits the XML document to UCCnet.
10. UCCnet generates an MDN and returns this worklist response message to the TPI Server. The TPI connector transforms the message into a UCCnet_envelope business object, maps it to a UCCnetGBO_envelope sample business object, which it then sends to the UCCnet_processWorklist collaboration.
11. The UCCnet_processWorklist collaboration processes the generic UCCnetGBO_envelope sample business object. It identifies the business object as representing an ITEM_ADD notification response. It is dispatched to the specific ITEM_ADD_CHANGE sub-collaboration.
12. This collaboration then sends a UCCnetGBO_envelope sample business object for ItemPublicationAdd to the TPI connector. The TPI connector builds the UCCnet XML document from the UCCnet_envelope business object and delivers this ItemPublication document to the TPI Server.
13. The TPI Server uses the trading partner profile for UCCnet, creates the digest, encrypts, and transmits the document to UCCnet.
14. The MDN is generated by UCCnet and returned to the TPI Server.
15. The UCCnet_requestWorklist collaboration sends another request for notifications.

4.3 Installation of Item Sync collaboration

The UCCnet Item Sync Collaboration represents synchronization activities between a supplier and UCCnet. The architectural components involved are the supply-side trading partner legacy systems, the InterChange Server (ICS), various specific connectors (which communicate between legacy systems and the ICS), and the Trading Partner Interchange Server (TPI), which provides Internet access to the UCCnet system. These components are shown in Figure 4-2 on page 171.

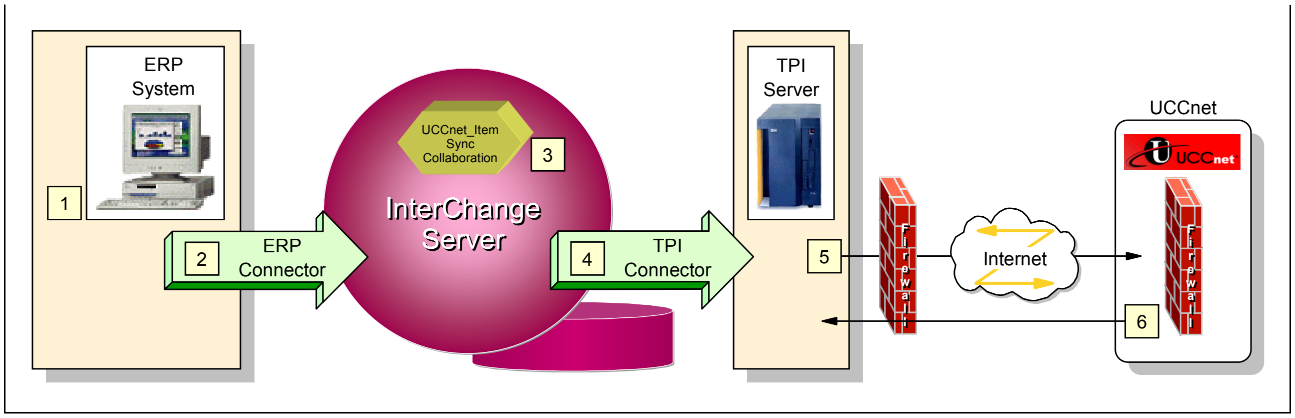


Figure 4-2 UCCnet solution with TPI Server

This chapter describes the following scenarios to deploy the UCCnet Item Sync collaboration:

- Using TPI for the supplier and TPI to simulate UCCnet
- Replacing the TPI connector with an MQ connector
- Replacing the TPI Server with iSoft's P2PAgent

4.3.1 Product installation

For the installation of the Item Sync collaboration and the required customizations, we assume that a configured Interchange Server, including a queue manager and a working TPI Server, are available.

Before describing the actual scenarios, we need to install and import the required components of the Item Sync solution.

The installation program for the Item Sync collaboration prompts you for the directory within which the installation program will create a folder called collaborations and a folder called repository. The folder collaborations contains a JAR file and a few database scripts, while the folder repository contains repository import files for collaboration templates, maps, relationships and business objects. You may want to merge those folders into your existing installation directory for the InterChange Server or you may opt for the InterChange Server installation directory as the installation directory for the Item Sync collaboration.

4.3.2 Importing the solution components

To import these repository files, you can use the command line tool repos_copy or you can import them by using the System Manager. Select **File -> Open from file** in the System Manager and select each file in the folder repository that relates to the UCCnet Item Sync collaboration. Import first the business objects, followed by the maps, the relationships and the collaboration templates.

The import of maps will generate warning messages, which you can ignore. The warnings are resolved by compiling the maps in the System Manager.

During the import of a relationship, you may get a warning about creating a schema. You can select **Ignore** at this time.

When the import is finished, open the Relationship Designer for each relationship and save it. This will trigger the creation of the schema in the database, as shown in Figure 4-3.

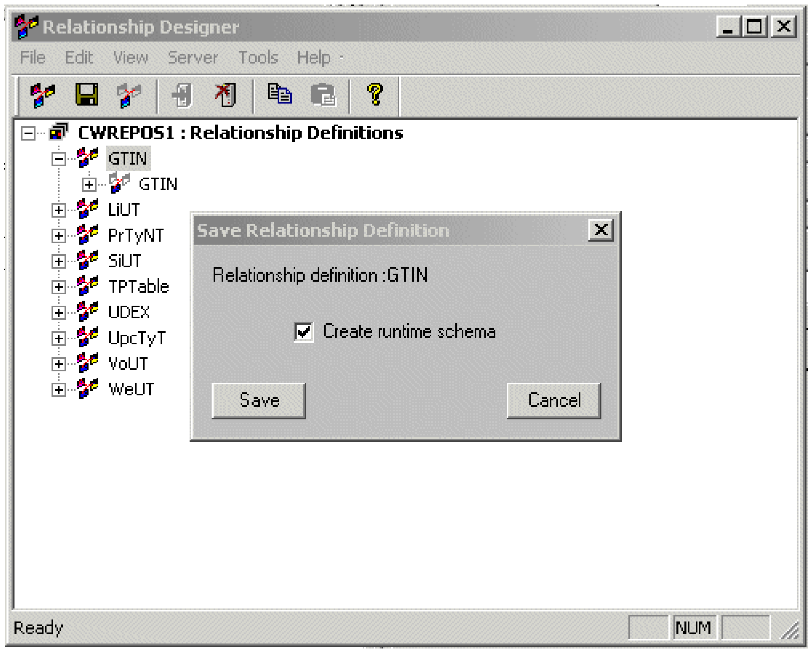

Relationship Designer
File Edit View Server Tools Help
CWREPOS1 : Relationship Definitions
GTIN
GTIN
LiUT
PrTyNT
SiUT
TPTable
UDEX
UpcTyT
VoUT
WeUT
Save Relationship Definition
Relationship definition :GTIN
Create runtime schema
Save
Cancel
Ready
NUM

Figure 4-3 Loading relationships in the database

Note: While you create the schema for the relationships, you may get an error message saying that nmake.exe was not found. Install Microsoft® Visual Studio to provide a C compiler to compile the stored procedures. Also, when installing Visual Studio, make sure that the environment variables are updated correctly.

4.3.3 Database customization

The Item Sync collaboration requires a number of changes and add-ons to the InterChange Server repository database. The folder collaborations\dependencies\UCCnet\db2 contains three DB2® scripts that you need to run to update the ICS database. Before we run those scripts, open the file InitializeRelationshipTables.sql in a text editor and update the ALTER TABLE statements. The file contains:

```
ALTER TABLE "CROSSWORLDS"."PROCESSED_GTIN" ADD("GTIN" VARCHAR2(100) NOT NULL,
"WITHDRAWN" VARCHAR2(1) NOT    NULL)
```

Assuming that the ICS connects to DB2 using the user db2admin, replace the above statement with:

```
ALTER TABLE DB2ADMIN.PROCESSED_GTIN ADD COLUMN GTIN VARCHAR(100);
ALTER TABLE DB2ADMIN.PROCESSED_GTIN ADD COLUMN WITHDRAWN VARCHAR(1);
```

Save and close this SQL file. Open a DB2 command window, change to the directory that holds the SQL scripts and execute the following commands:

```
DB2 connect to ICSREPOS user db2admin using password
DB2 -tvf audit_los.sql
DB2 -tvf InitializeRelationshipTables.sql
DB2 -tvf trading_partner.sql
DB2 connect reset
```

Stop and restart the ICS after you have executed these database commands.

Note: The commands above assume that the schemas for the relationships are created. Replace `ICSREPOS` with the name of the ICS repository. Replace `db2admin` and `password` with the correct user ID and password that you use for the ICS to connect to its database repository. This user ID and password are set during initial installation of the ICS and can be changed by using the InterChange Server Configuration Wizard.

4.3.4 Installing additional samples for the UCCnet Item Sync collaboration

To further assist with the deployment of the Item Sync collaboration, you can download sample ICS objects from the following Web site:

`http://www-1.ibm.com/support/entdocview.wss?uid=swg24001766`

Download the package into a temporary directory and run the installation program UCCNet_NT.exe. Next, unzip the file UCCnetSamples.zip into the installation directory of the InterChange Server.

Import the repository file called UCCnetSamples.in into the ICS repository via the System Manager. If you see any warnings, you can ignore them at this time.

4.4 Implementation of scenario 1

In this section, we describe the implementation of a UCCnet collaboration that interacts with UCCnet via TPI and the TPI connector. In later sections, this setup is changed to use the MQ connector and, later, iSoft’s P2PAgent.

4.4.1 Scenario overview

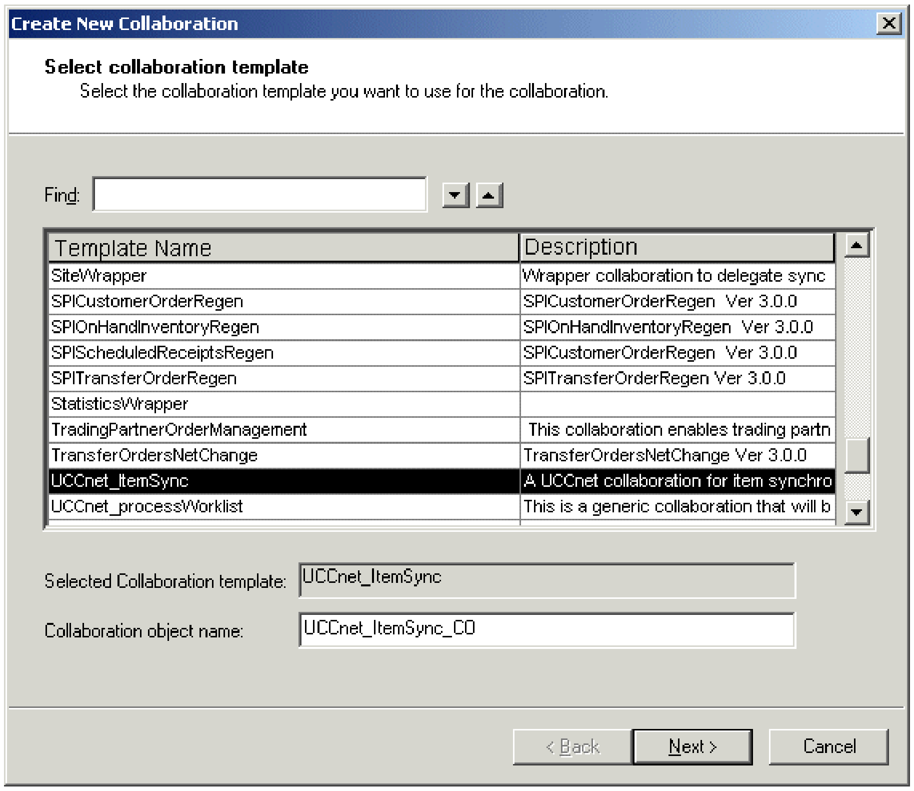
Figure 4-2 on page 171 shows the overall flow that we are going to discuss in this section. The ERP system in our case is SAP. However, we are using the SAP connector only in a simulation mode. Instead of using a real SAP system, we are using the Test connector based on an SAP connector profile. We provide an SAP-based business object to the collaboration that is going to map the data to an XML document. This XML document is written in the directory that is monitored by the TPI connector. When the TPI connector detects this new XML document, it will pass it on to the TPI Server for transmission to UCCnet over the Internet.

The implementation of this scenario consists of:

- The definition and customization of the collaboration object
- The configuration of the TPI connector
- The configuration of the Port connector, which is used to bind any unused ports of the collaboration
- The configuration of the SAP connector
- The configuration of the TPI Server

4.4.2 Collaboration object definition and customization

1. In the System Manager, right-click the folder **Collaboration Objects** and select **New Collaboration Object**. The collaboration creation wizard will appear.
2. Select the collaboration template **UCCNet_ItemSync** as shown in Figure 4-4 and provide a name for the new collaboration object.


Create New Collaboration

Select collaboration template
Select the collaboration template you want to use for the collaboration.

Find:

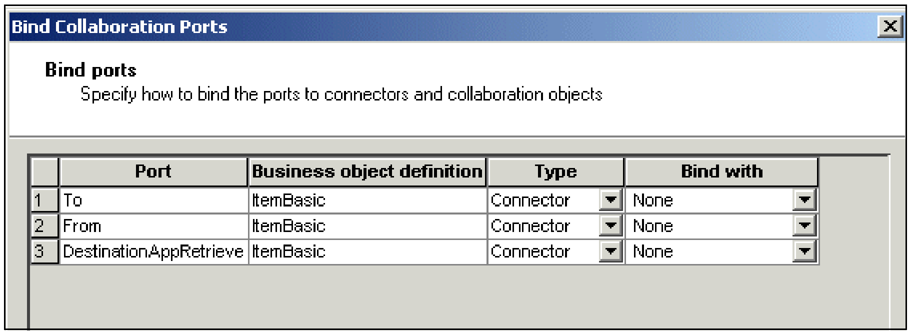
Template Name	Description
SiteWrapper	Wrapper collaboration to delegate sync
SPICustomerOrderRegen	SPICustomerOrderRegen Ver 3.0.0
SPIOnHandInventoryRegen	SPIOnHandInventoryRegen Ver 3.0.0
SPIScheduledReceiptsRegen	SPICustomerOrderRegen Ver 3.0.0
SPITransferOrderRegen	SPITransferOrderRegen Ver 3.0.0
StatisticsWrapper	
TradingPartnerOrderManagement	This collaboration enables trading partn
TransferOrdersNetChange	TransferOrdersNetChange Ver 3.0.0
UCCnet_ItemSync	A UCCnet collaboration for item synchro
UCCnet_processWorklist	This is a generic collaboration that will b

Selected Collaboration template: UCCnet_ItemSync

Collaboration object name: UCCnet_ItemSync_CO

Figure 4-4 Select the collaboration Template UCCnet_ItemSync.

3. Select **Next**, which will allow you to bind the ports of the collaboration. You may keep the default value of `None` at this time and bind the ports later after you have completed the configuration of the connectors. Click **Next**.

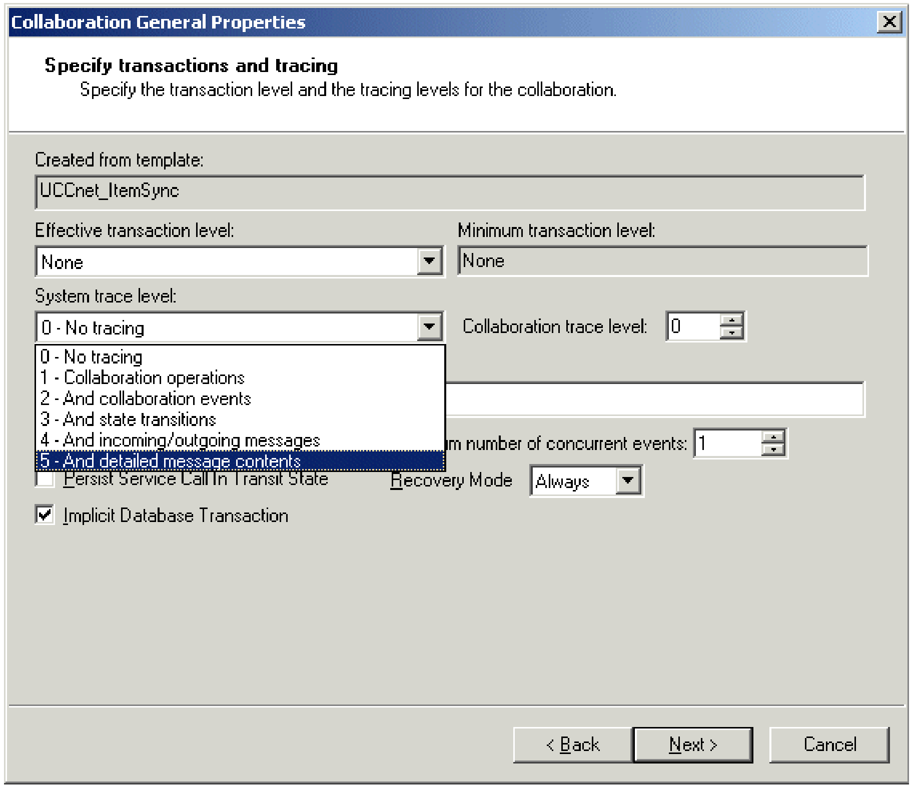

Bind Collaboration Ports

Bind ports
Specify how to bind the ports to connectors and collaboration objects

	Port	Business object definition	Type	Bind with
1	To	ItemBasic	Connector	None
2	From	ItemBasic	Connector	None
3	DestinationAppRetrieve	ItemBasic	Connector	None

Figure 4-5 Binding ports while creating new collaboration object.

4. During this step, you can set trace and transaction levels. To allow for easier debugging, set the property System Trace Level to `5`, as shown in Figure 4-6 on page 175. Click **Next** again.


Collaboration General Properties

Specify transactions and tracing

Specify the transaction level and the tracing levels for the collaboration.

Created from template: UCCnet_ItemSync

Effective transaction level: None

Minimum transaction level: None

System trace level: 0 - No tracing

Collaboration trace level: 0

- 0 - No tracing
- 1 - Collaboration operations
- 2 - And collaboration events
- 3 - And state transitions
- 4 - And incoming/outgoing messages
- 5 - And detailed message contents

m number of concurrent events: 1

Persist Service Call In Transit State

Recovery Mode: Always

Implicit Database Transaction [✓]

Figure 4-6 Setting System Trace Level

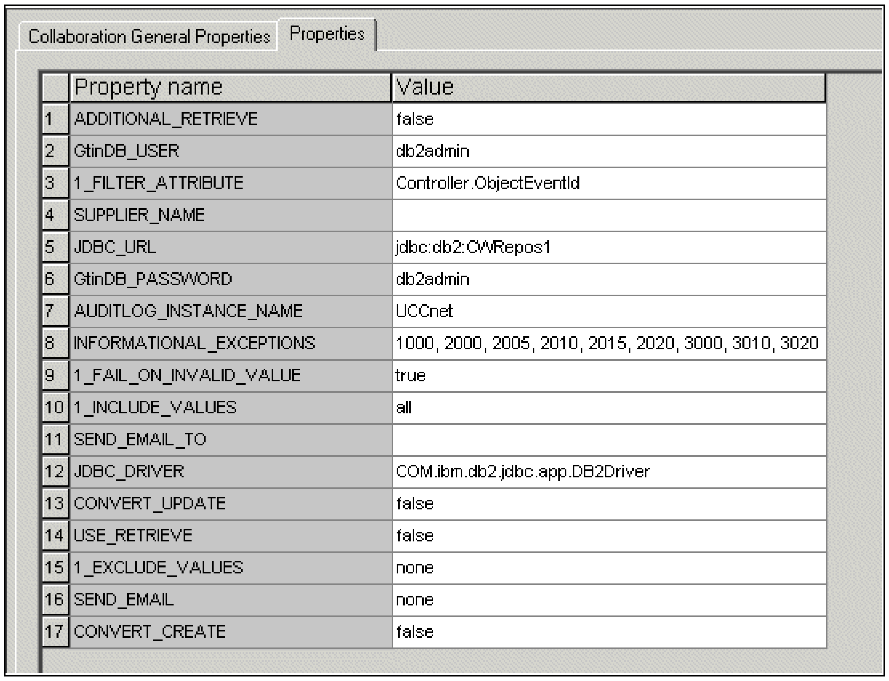
5. The final step allows you to customize the collaboration properties. Make the following changes to the properties as described in Table 4-1.

Table 4-1 Collaboration object properties

Property Name	Value
GinDB_USER	Specify the database user, for example: `db2admin`
JDBC_URL	Specify the connection URL for the database, for example: `jdbc:db2:CWREPOS1`
GinDB_PASSWORD	Specify the password of the database user
JDBC_DRIVER	The driver to connect to the database, for example: COM.ibm.db2.jdbc.app.DB2Driver

Figure 4-7 on page 176 shows other properties that can be customized. These changes can be made after the collaboration object is created by right-clicking the collaboration object and selecting the option **Properties** from the context menu.

6. Select **Finish** to complete the definition of the collaboration object.

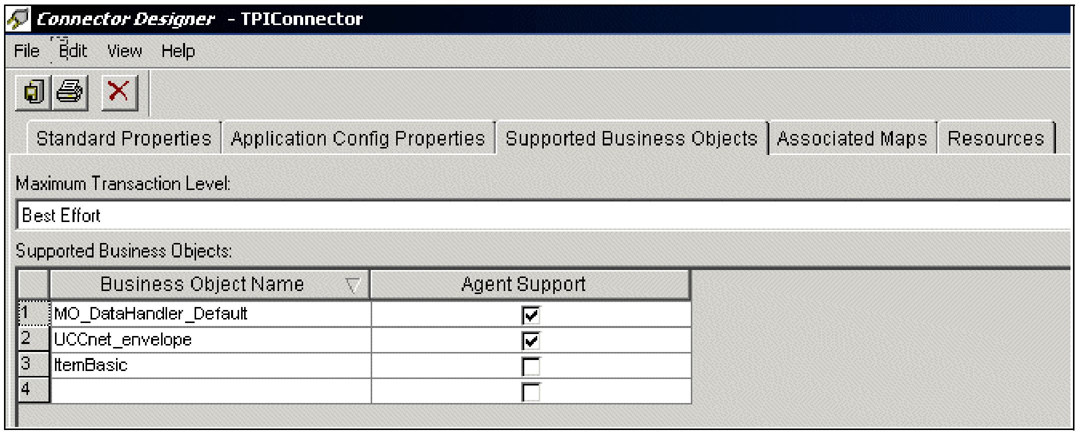

Collaboration General Properties | Properties

	Property name	Value
1	ADDITIONAL_RETRIEVE	false
2	GtinDB_USER	db2admin
3	1_FILTER_ATTRIBUTE	Controller.ObjectEventId
4	SUPPLIER_NAME	
5	JDBC_URL	jdbc:db2:CWRepos1
6	GtinDB_PASSWORD	db2admin
7	AUDITLOG_INSTANCE_NAME	UCCnet
8	INFORMATIONAL_EXCEPTIONS	1000, 2000, 2005, 2010, 2015, 2020, 3000, 3010, 3020
9	1_FAIL_ON_INVALID_VALUE	true
10	1_INCLUDE_VALUES	all
11	SEND_EMAIL_TO	
12	JDBC_DRIVER	COM.ibm.db2.jdbc.app.DB2Driver
13	CONVERT_UPDATE	false
14	USE_RETRIEVE	false
15	1_EXCLUDE_VALUES	none
16	SEND_EMAIL	none
17	CONVERT_CREATE	false

Figure 4-7 Collaboration object properties

4.4.3 TPI connector configuration

1. Select the **TPIConnector** object in the System Manager and open it in the Connector Designer. Select the tab **Supported Business Objects** and add the following business objects:
   - ItemBasic
   - MO_DataHandler_Default
   - UCCnet_envelope
2. Select the check box for **Agent Supported** for all Business Object Names except ItemBasic, as shown in Figure 4-6 on page 175.


Connector Designer - TPIConnector
File Edit View Help
Standard Properties
Application Config Properties
Supported Business Objects
Associated Maps
Resources

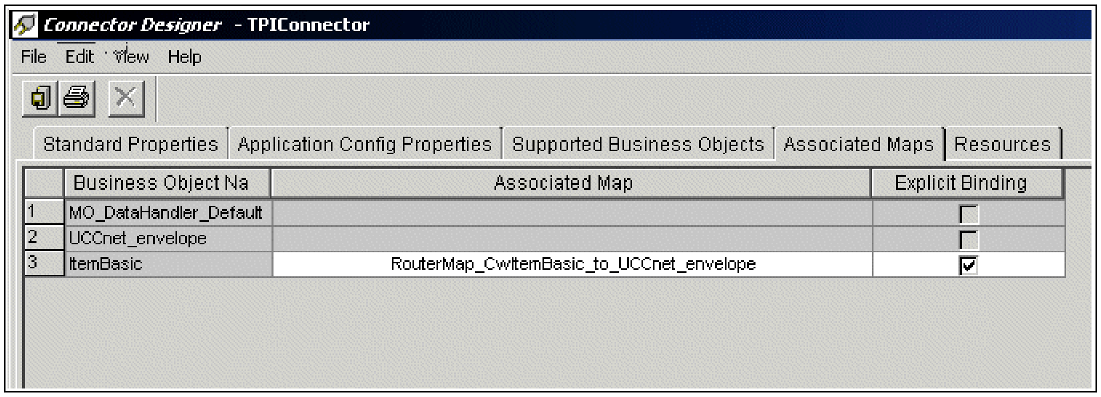
Maximum Transaction Level:
Best Effort

Supported Business Objects:

	Business Object Name	Agent Support
1	MO_DataHandler_Default	✓
2	UCCnet_envelope	✓
3	ItemBasic	☐
4		☐

Figure 4-8 Setting business object support for SAP Connector

3. Now select the tab **Associated Maps** and make sure that the correct maps and business objects are present, as shown in Figure 4-9 on page 177.


Connector Designer - TPIConnector
File Edit View Help
Standard Properties
Application Config Properties
Supported Business Objects
Associated Maps
Resources

	Business Object Na	Associated Map	Explicit Binding
1	MO_DataHandler_Default		☐
2	UCCnet_envelope		☐
3	ItemBasic	RouterMap_CwItemBasic_to_UCCnet_envelope	☑

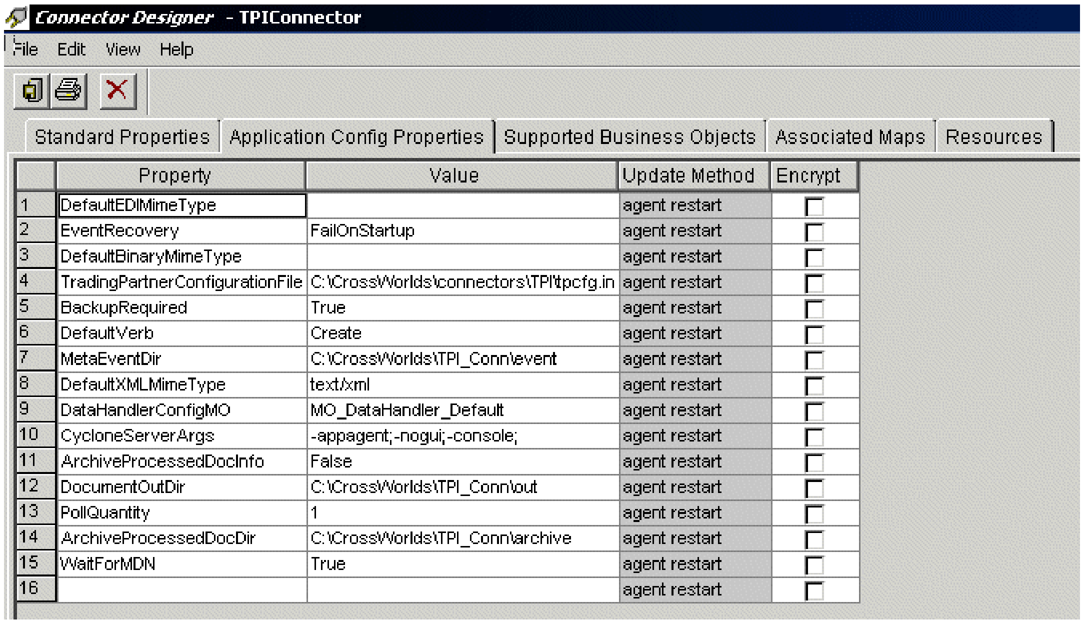
Figure 4-9 Associating maps with business objects for TPIConnector

4. Select the tab **Application Config Properties** and change the value of the properties listed in Table 4-2. You also need to create the three directories that are mentioned in Table 4-2.

Table 4-2 TPI Connector App Config Properties

TradingPartnerConfiguartionFile	Path of the configuration file, for example C:\CrossWorlds\connectors\TPI\tpicfg.in
MetaEventDir	Name of the event directory, for example C:\CrossWorlds\TPI_Conn\event
DefaultXMLMimeType	text/xml
DocumentOutDir	Name of the document directory, for example C:\CrossWorlds\TPI_Conn\out
ArchiveProcessedDocDir	Name of the archive directory, for example C:\CrossWorlds\TPI_Conn\archive

Figure 4-10 shows the updated properties in the Connector Designer.


Connector Designer - TPIConnector
File Edit View Help
Standard Properties
Application Config Properties
Supported Business Objects
Associated Maps
Resources

	Property	Value	Update Method	Encrypt
1	DefaultEDIMimeType		agent restart	☐
2	EventRecovery	FailOnStartup	agent restart	☐
3	DefaultBinaryMimeType		agent restart	☐
4	TradingPartnerConfigurationFile	C:\CrossWorlds\connectors\TPI\tpcfg.in	agent restart	☐
5	BackupRequired	True	agent restart	☐
6	DefaultVerb	Create	agent restart	☐
7	MetaEventDir	C:\CrossWorlds\TPI_Conn\event	agent restart	☐
8	DefaultXMLMimeType	text/xml	agent restart	☐
9	DataHandlerConfigMO	MO_DataHandler_Default	agent restart	☐
10	CycloneServerArgs	-appagent;-nogui;-console;	agent restart	☐
11	ArchiveProcessedDocInfo	False	agent restart	☐
12	DocumentOutDir	C:\CrossWorlds\TPI_Conn\out	agent restart	☐
13	PollQuantity	1	agent restart	☐
14	ArchiveProcessedDocDir	C:\CrossWorlds\TPI_Conn\archive	agent restart	☐
15	WaitForMDN	True	agent restart	☐
16			agent restart	☐

Figure 4-10 TPI Connector Application Config Properties

5. Save the changes to the TPIConnector object and close the Connector Designer.
6. The connector configuration refers to a configuration file tpicfg.in, which we need to create. This file holds information about patterns and MIME types. A sample file can be found in the directory connectors\TPI\samples within the ICS installation directory. Copy the sample file to the location that you specified in the connector properties (see Figure 4-10 on page 177).
7. Open the copied file in a text editor and add the line for the UCCnet trading partner and MIME type text/xml.

   ```
   <Suppliername> text/xml
   ```

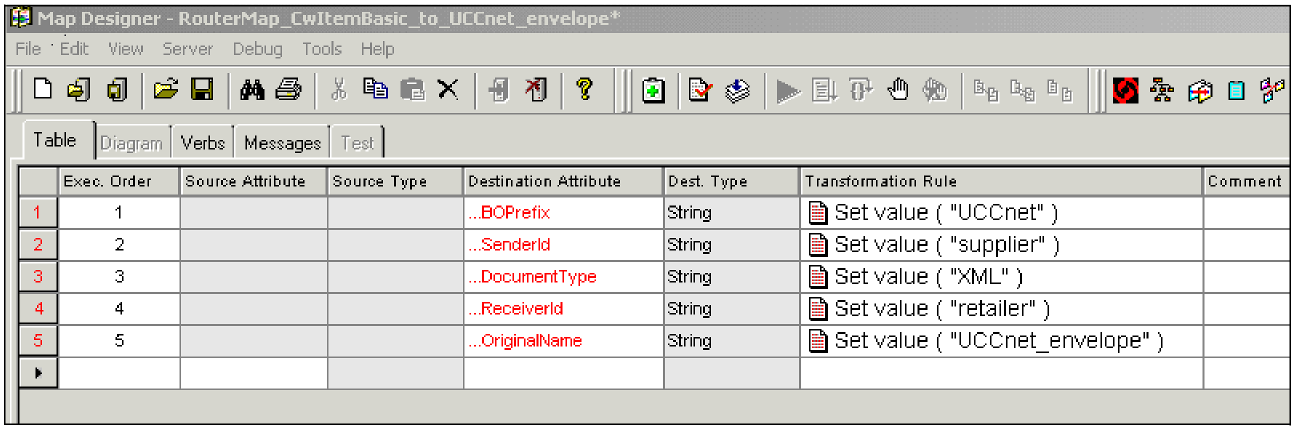
   You can comment out the other data in the file by prefixing each line with a pound sign (#).
8. The TPIConnector is started via a batch file called start_TPI.bat. We need to update this file to specify the installation directory for the TPI Server. The file start_TPI.bat is located in connectors\TPI within the ICS installation directory. Open it in a text editor and make the following changes:
   – Locate the statement `SET CYCLONEHOMEDIR=` and update this environment variable to have the TPI installation directory as its value.

     ```
     set CYCLONEHOMEDIR=<TPI server installation path>
     ```

   – If you have not done so yet, create the folders that were specified in the TPIConnector configuration (see Figure 4-10 on page 177). For example:

     ```
     C:\CrossWorlds\TPI_Conn\archive
     C:\CrossWorlds\TPI_Conn\event
     C:\CrossWorlds\TPI_Conn\logs
     C:\CrossWorlds\TPI_Conn\out
     ```

9. Finally, open the map RouterMap_CwItemBasic_to_UCCnet_envelope in the Map Designer. Select the tab **Table**, locate the attributes SenderID and ReceiverID and update the mapping statements. These properties should contain the TPI profile names for the sender and receiver trading partners.


Map Designer - RouterMap_CwItemBasic_to_UCCnet_envelope*
File Edit View Server Debug Tools Help
Table Diagram Verbs Messages Test

	Exec. Order	Source Attribute	Source Type	Destination Attribute	Dest. Type	Transformation Rule	Comment
1	1			...BOPrefix	String	Set value ("UCCnet")	
2	2			...SenderId	String	Set value ("supplier")	
3	3			...DocumentType	String	Set value ("XML")	
4	4			...ReceiverId	String	Set value ("retailer")	
5	5			...OriginalName	String	Set value ("UCCnet_envelope")	
▸							

Figure 4-11 Configuring MAP

10. Shut down the Interchange server and the TPIConnectorgracefully . Restart the ICS and the TPIConnector to pick up the changes that were made.

4.4.4 Port connector configuration

Usually, the Port connector is configured during the base installation and configuration of the ICS. Start the MQ Explorer program and verify that the following queues exist:

- IC/CWREPOS1/PORTCONNECTOR
- AP/PORTCONNECTOR/CWREPOS1
- APE/PORTCONNECTOR/CWREPOS1
- ICA/CWREPOS1/PORTCONNECTOR
- APA/PORTCONNECTOR/CWREPOS1

Replace CWREPOS1 with the name of your ICS. If those queues do not exist, define them using MQ Explorer.

Open the PortConnector object in the Connector Designer and add the business object ItemBasic to the list of supported business objects. You can now save the PortConnector in the System Manager.

4.4.5 SAP connector configuration

There is more to the configuration of an SAP connector than what we describe in this section. However, full integration with SAP and step-by-step configuration of the SAP connector is beyond the scope of this redbook. For more documentation and scenarios that include the SAP connector, refer to “Related publications” on page 221.

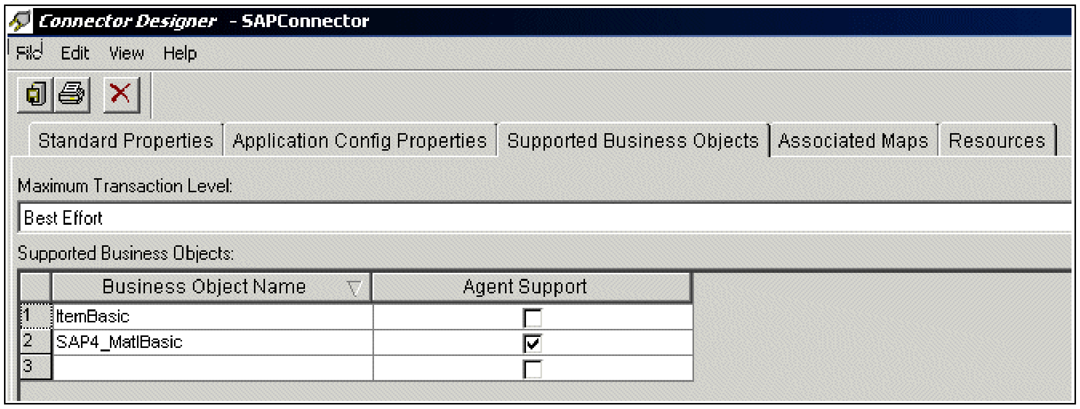
1. Open the SAP Connector.

2. Click the **Supported Business Objects** tab and add the business objects listed in Table 4-3.

Table 4-3 Supported business objects for the SAP connector

Business Object Name	Agent Supported
ItemBasic	
SAP4_MATBasic	yes

Figure 4-12 shows the completed tab Supported Business Objects for the SAP connector.


Connector Designer - SAPConnector
File Edit View Help
Standard Properties
Application Config Properties
Supported Business Objects
Associated Maps
Resources

Maximum Transaction Level:
Best Effort

Supported Business Objects:

	Business Object Name	Agent Support
1	ItemBasic	☐
2	SAP4_MatlBasic	✓
3		☐

Figure 4-12 Setting business object support for SAP Connector

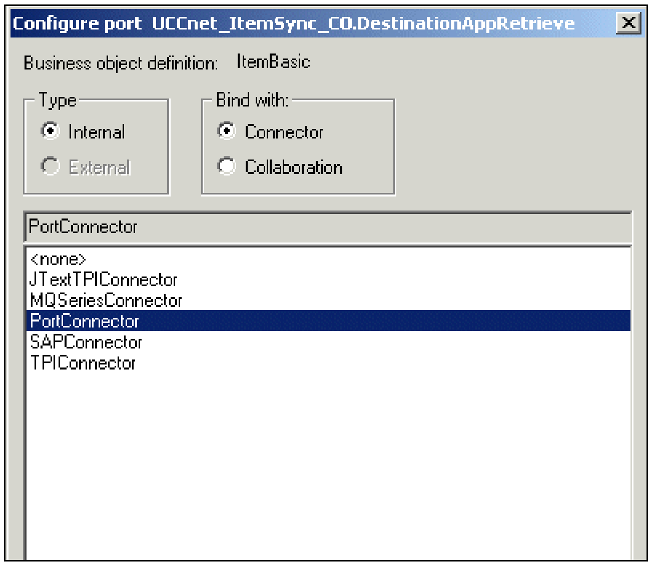
3. Save and close.

4.4.6 Binding the ports

Now that we have configured all the connectors that we want to use, we can go back to the collaboration object and bind the ports to the connectors.

In the System Manager, double-click the collaboration object that we created previously (see 4.4.2, “Collaboration object definition and customization” on page 173). This will bring up the graphical view of the collaboration. Right-click the **From** port and select **Bind Port**. A window similar to Figure 4-13 will appear. For the From port, select the **SAPConnector**. Repeat this process for the To port and the DestinationAppRetrieve port. The To port should be bound to the TPIConnector and the DestinationAppRetrieve port should be bound to the PortConnector. Binding to the PortConnector basically means that this port is not going to be used.

If the required connector does not appear in the list of possible connectors, you should review the list of supported business objects for the missing connector and make sure that that list is complete.


Configure port UCCnet_ItemSync_CO.DestinationAppRetrieve

Business object definition: ItemBasic

Type: (● Internal) ○ External

Bind with: (● Connector) ○ Collaboration

PortConnector

<none>
JTextTPIConnector
MQSeriesConnector
PortConnector
SAPConnector
TPIConnector

Figure 4-13 Binding the ports

The completed collaboration object is shown in Figure 4-14.

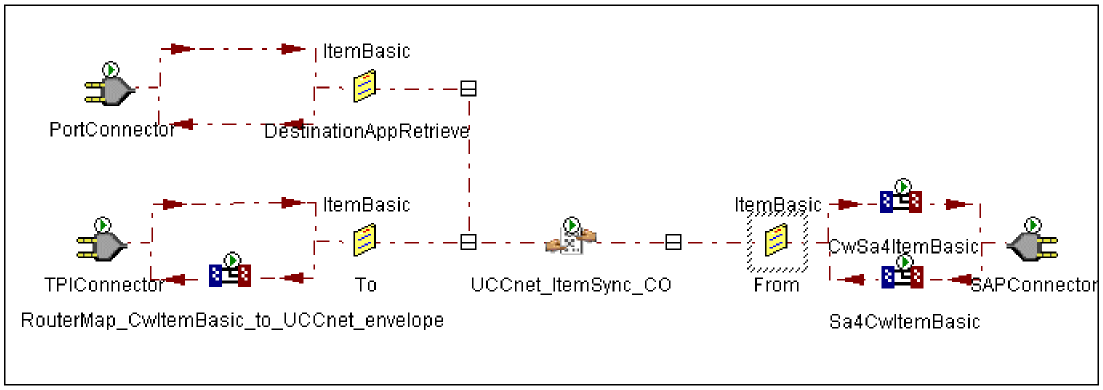
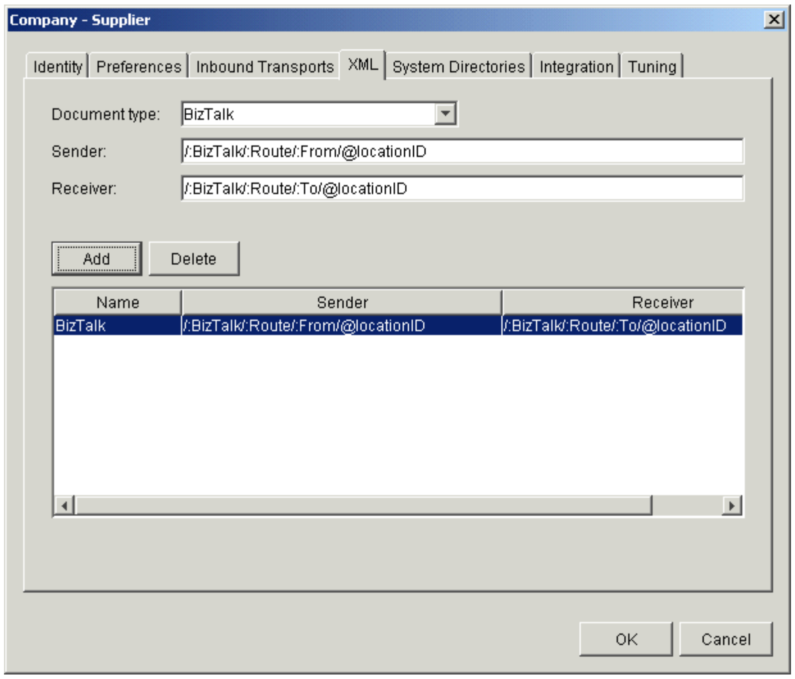


Figure 4-14 Collaboration object diagram

4.4.7 TPI Server configuration

Assuming that the TPI Server is configured in a way similar to the configurations described in Chapter 3, “Implementing multi-product AS/2 communication with trading partners” on page 101, the configuration of the company profile needs to be updated to support the XML documents for UCCnet. Start the TPI Administrator and select the view **Company Profiles**. Double-click your company profile and select the tab **XML** (see Figure 4-15). Select **BizTalk** in the drop-down box labeled Document type and click **Add**. Click **OK** to close the company profile. This change is applied dynamically, in case the TPI Server is running.



Document type: BizTalk
Sender: /:BizTalk/:Route/:From/@locationID
Receiver: /:BizTalk/:Route/:To/@locationID

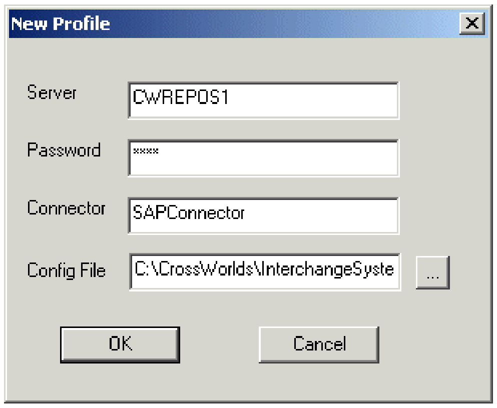
Name	Sender	Receiver
BizTalk	/:BizTalk/:Route/:From/@locationID	/:BizTalk/:Route/:To/@locationID

Figure 4-15 Configuring TPI company profile for UCCnet

4.4.8 Running the test scenario

To test the collaboration, the connectors and the TPI Server setup, we use the Test Connector to send a SAP business object through the collaboration to the TPI Server.

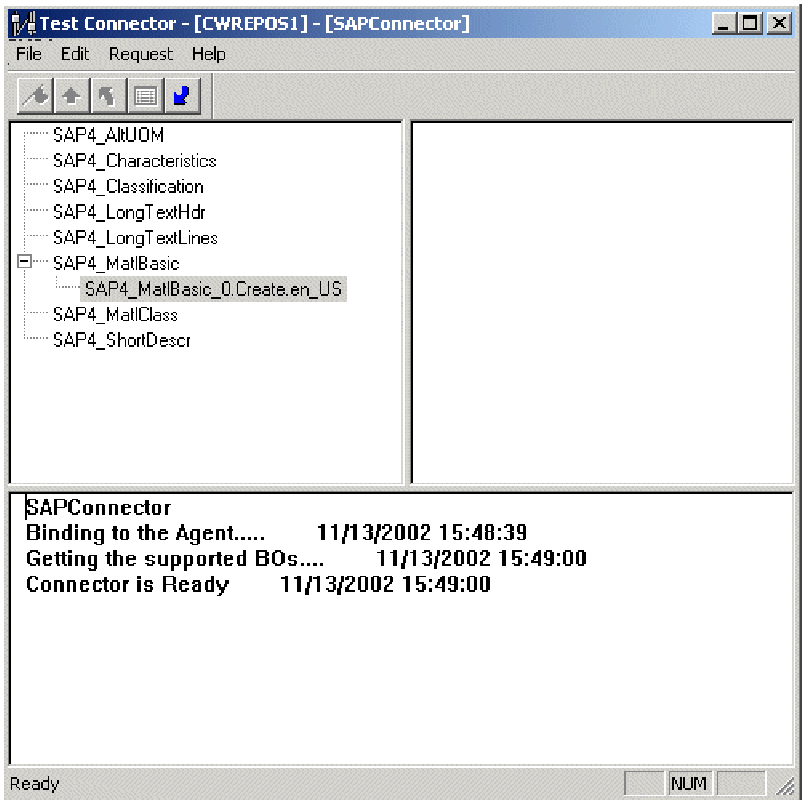
1. Start the Test Connector and select **File -> New Profile.** Click **Add** to create a new profile for the testing of the SAPConnector.
2. Provide a server name, a password, the connector name and the location of the InterchangeSystem.cfg file, as shown in Figure 4-16 on page 182.



Server	CWREPOS1
Password	****
Connector	SAPConnector
Config File	C:\CrossWorlds\InterchangeSyste

Figure 4-16 Creating new profile in Test Connector

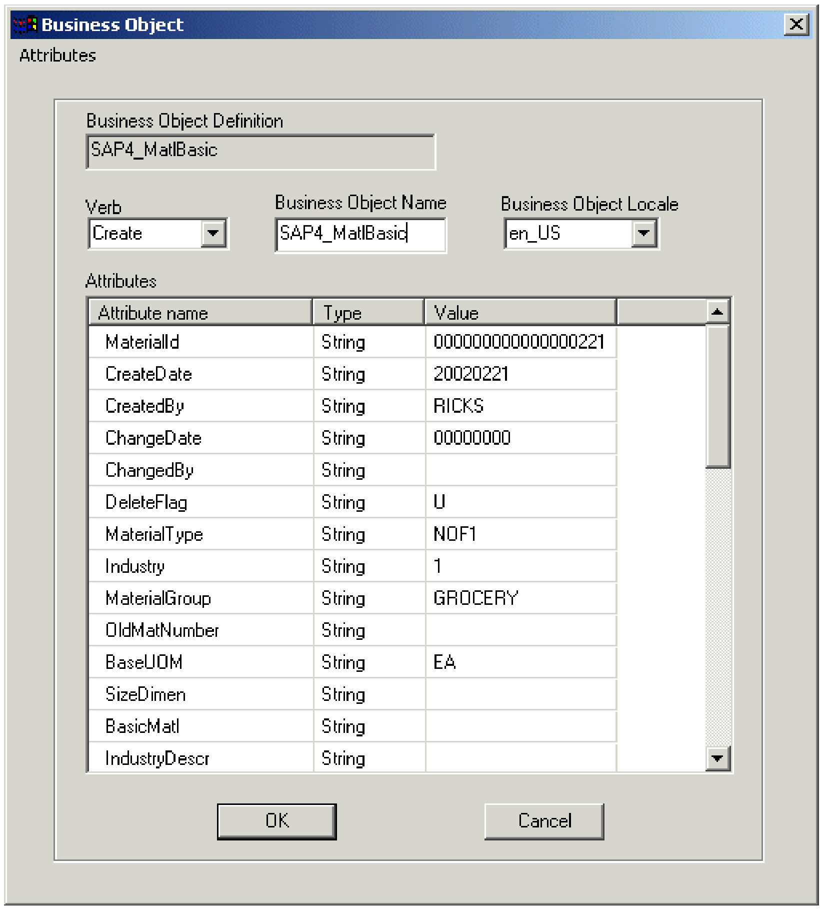
3. Click **OK** to close the profile definition window and select **File -> Connect Agent** to bind the agent.
4. When the Test connector is connected to the server, a list of supported business objects will appear. Select the **SAP4_MatBasic** business object.
5. Select **Edit -> Load BO** if you have an existing saved business object. If you do not have a saved business object, you can create one by right-clicking the **SAP4_MatBasic** business object and selecting **New** from the context menu. You can then name the new object and populate the required fields.



```
SAPConnector
Binding to the Agent.....      11/13/2002 15:48:39
Getting the supported BOs....      11/13/2002 15:49:00
Connector is Ready      11/13/2002 15:49:00
```

Figure 4-17 Running Test Connector for SAP Agent

Figure 4-18 on page 183 shows an SAP-based business object with applicable values for different attributes of this business object.



Business Object

Attributes

Business Object Definition: SAP4_MatlBasic

Verb: Create

Business Object Name: SAP4_MatlBasic

Business Object Locale: en_US

Attributes

Attribute name	Type	Value
MaterialId	String	000000000000000221
CreateDate	String	20020221
CreatedBy	String	RICKS
ChangeDate	String	00000000
ChangedBy	String	
DeleteFlag	String	U
MaterialType	String	NOF1
Industry	String	1
MaterialGroup	String	GROCERY
OldMatNumber	String	
BaseUOM	String	EA
SizeDimen	String	
BasicMatl	String	
IndustryDescr	String	

Figure 4-18 Sample BO for SAP4_MatlBasic with Test Connector

6. Select **Request -> Send** to send the business object to the collaboration.
7. Review the ICS log and the TPIConnector log to inspect the execution.
8. If all goes well, the TPI Server will pick up the XML document and send it off.

Figure 4-19 on page 184 and Figure 4-20 on page 185 show the UCCnet request message that is being built by the ICS and sent by the TPI Server.

```
<?xml version="1.0" ?>
  <!DOCTYPE envelope (View Source for full doctype...)>
- <envelope communicationVersion="2.0">
- <messageHeader>
- <messageIdentifier>
  <value>MSGID1036612411921</value>
  </messageIdentifier>
  <userId>testUser</userId>
- <representingParty>
  <gln>0000000000001</gln>
  </representingParty>
  </messageHeader>
- <body>
- <transaction>
- <entityIdentification>
  <uniqueCreatorIdentification>MSGID1036612411921</uniqueCreatorIdentification>
- <globalLocationNumber>
  <gln>0000000000001</gln>
  </globalLocationNumber>
  </entityIdentification>
- <command>
- <documentCommand>
- <documentCommandHeader type="ADD">
- <entityIdentification>
  <uniqueCreatorIdentification>UID21036612411921</uniqueCreatorIdentification>
- <globalLocationNumber>
  <gln>0000000000001</gln>
  </globalLocationNumber>
  </entityIdentification>
  </documentCommandHeader>
- <documentCommandOperand>
- <item>
- <documentInformation documentStructureVersion="2.0" status="ORIGINAL">
  <creationDate>2002-11-06</creationDate>
  <lastUpdateDate>2002-11-06</lastUpdateDate>
  </documentInformation>
- <itemInformation>
- <globalTradeItemNumber>
  <gtin>02050000000454</gtin>
  </globalTradeItemNumber>
  <itemEffectiveDate>2002-11-06</itemEffectiveDate>
  <versionStatus value="F" />
- <globalLocationNumber>
  <gln>0000000000001</gln>
  </globalLocationNumber>
  <itemBrandName>Natural Beauty Soap</itemBrandName>
  <productTypeName>EA</productTypeName>
- <categoryList>
  <categoryCode>UDEX.05.0139.0334</categoryCode>
  </categoryList>
```

Figure 4-19 XML data sent to UCCnet for ItemAdd

```
- <itemDimensions>
  <size>5.0</size>
  <sizeUnits>OZ</sizeUnits>
  <height>1.0</height>
  <width>1.0</width>
  <length>1.0</length>
  <linearUnits>IN</linearUnits>
  <netWeight>5.0</netWeight>
  <grossWeight>5.0</grossWeight>
  <weightUnits>OZ</weightUnits>
  <volume>1.0</volume>
  <volumeUnits>CI</volumeUnits>
  <ti>10</ti>
  <hi>10</hi>
  <pack>1</pack>
  </itemDimensions>
- <itemDescription>
  <itemName>NOF1</itemName>
  <publicOrPrivate value="PUBLIC" />
  <upcType>EN</upcType>
  <upc>2050000000454</upc>
  </itemDescription>
- <itemMiscInfo>
  <consumerUnit value="TRUE" />
  <orderable value="TRUE" />
  </itemMiscInfo>
- <itemDates>
  <firstOrderDate>2002-11-06</firstOrderDate>
  </itemDates>
  </itemInformation>
  </item>
  </documentCommandOperand>
  </documentCommand>
  </command>
  </transaction>
  </body>
  </envelope>
```

Figure 4-20 XML data sent to UCCnet for ItemAdd - continued

After sending the business object to the UCCnet, you will get an MDN and a response message file from UCCnet. Figure 4-21 on page 186 shows a sample XML response message from UCCnet.

This XML document is typically the input for the collaboration UCCnet_requestWorklist.

```
<?xml version="1.0" ?>
  <!DOCTYPE MQenvelope (View Source for full doctype...)>
- <envelope communicationVersion="2.0">
- <messageHeader>
- <messageIdentifier>
  <value>1023219322107</value>
  </messageIdentifier>
  <userId>IBMRTPERP2</userId>
- <representingParty>
  <gln>7789788000015</gln>
  </representingParty>
  </messageHeader>
- <body>
- <transaction>
- <entityIdentification>
  <uniqueCreatorIdentification>1023219322107</uniqueCreatorIdentification>
- <globalLocationNumber>
  <gln>7789788000015</gln>
  </globalLocationNumber>
  </entityIdentification>
- <command>
- <queryCommand showDetails="TRUE">
- <entityIdentification>
  <uniqueCreatorIdentification>1023219322107</uniqueCreatorIdentification>
- <globalLocationNumber>
  <gln>7789788000015</gln>
  </globalLocationNumber>
  </entityIdentification>
- <query type="NOTIFICATION">
- <where>
- <termList>
- <term>
- <field>
  <fieldName>status</fieldName>
  <value>ALL</value>
  </field>
  </term>
  </termList>
  </where>
  </query>
  </queryCommand>
  </command>
  </transaction>
  </body>
  </envelope>
```

Figure 4-21 Sample XML response notification sent by UCCnet

The notification message is then followed by an actual response from UCCnet, as shown in Figure 4-22 on page 187 and Figure 4-23 on page 188.

```
<?xml version="1.0" ?>
  <!DOCTYPE MQenvelope (View Source for full doctype...)>
- <envelope communicationVersion="2.0">
- <messageHeader>
- <to>
- <globalLocationNumber>
  <gln>0000077897886</gln>
  </globalLocationNumber>
  </to>
- <from>
- <globalLocationNumber>
  <gln>0614141800001</gln>
  </globalLocationNumber>
  </from>
- <messageIdentifier>
  <value>2814650</value>
  </messageIdentifier>
  <userId>UCCNET_SYSTEM</userId>
- <representingParty>
  <gln>0614141800001</gln>
  </representingParty>
  </messageHeader>
-<body>
- <response>
- <acknowledge>
- <acknowledgement>
- <acknowledgementHeader type="PROCESSED" success="TRUE" duplicate="TRUE">
+ <entityIdentification>
  <uniqueCreatorIdentification>3450644332534556899001</uniqueCreatorIdentification>
- <globalLocationNumber>
  <gln>0000077897886</gln>
  </globalLocationNumber>
  </entityIdentification>
- <messageIdentifier>
  <value>3450644332534556899001</value>
  </messageIdentifier>
  </acknowledgementHeader>
- <subdocumentValid success="TRUE">
- <documentIdentifier>
- <typedEntityIdentification type="TRANSACTION">
- <entityIdentification>
  <uniqueCreatorIdentification>76917633</uniqueCreatorIdentification>
+ <globalLocationNumber>
  <gln>0000077897886</gln>
  </globalLocationNumber>
  </entityIdentification>
  </typedEntityIdentification>
  </documentIdentifier>
- <subdocumentValid success="TRUE">
- <documentIdentifier>
- <typedEntityIdentification type="QUERY_COMMAND">
+ <entityIdentification>
```

Figure 4-22 Sample XML response message sent from UCCnet

```
<uniqueCreatorIdentification>76917633</uniqueCreatorIdentification>
- <globalLocationNumber>
  <gln>0000077897886</gln>
  </globalLocationNumber>
  </entityIdentification>
  </typedEntityIdentification>
  </documentIdentifier>
- <resultList showDetails="TRUE">
+ <notification type="PUBLICATION_INFORMATION" topic="PUB_RELEASE_NEW_ITEM">
  + <notification type="PUBLICATION_INFORMATION" topic="PUB_RELEASE_NEW_ITEM">
   +<notification type="PUBLICATION_INFORMATION" topic="PUB_RELEASE_WITHDRAW">
  + <notification type="PUBLICATION_INFORMATION" topic="PUB_RELEASE_WITHDRAW">
  + <notification type="PUBLICATION_INFORMATION" topic="PUB_RELEASE_WITHDRAW">
  + <notification type="PUBLICATION_INFORMATION" topic="PUB_RELEASE_WITHDRAW">
   +<notification type="PUBLICATION_INFORMATION" topic="PUB_RELEASE_WITHDRAW">
  + <notification type="PUBLICATION_INFORMATION" topic="PUB_RELEASE_NEW_ITEM">
  + <notification type="PUBLICATION_INFORMATION" topic="PUB_RELEASE_NEW_ITEM">
  </resultList>
    </subdocumentValid>
  </subdocumentValid>
  </acknowledgement>
  </acknowledge>
  </response>
  </body>
  </envelope>
```

Figure 4-23 Sample XML response message sent from UCCnet

4.5 Implementation of scenario 2

This scenario is actually only a step up to the next scenario. The third scenario will use the MQ connector and iSoft, instead of the TPI connector and TPI Server. In this intermediate step, we are replacing the TPI connector with the MQ connector. Note that not all editions of TPI Server support the MQ interface. Some editions support the JMS interface and not the MQ interface.

4.5.1 Updating the business object

Open the System Manager, right-click the business **UCCnet_envelope** and select **Copy**. Right-click the folder Business Object and select **Paste**. Rename the new business object as UCCnet_MQenvelope. Then open the new object in the Business Object Designer. Delete the element TPIRouteInfo and save the business object.

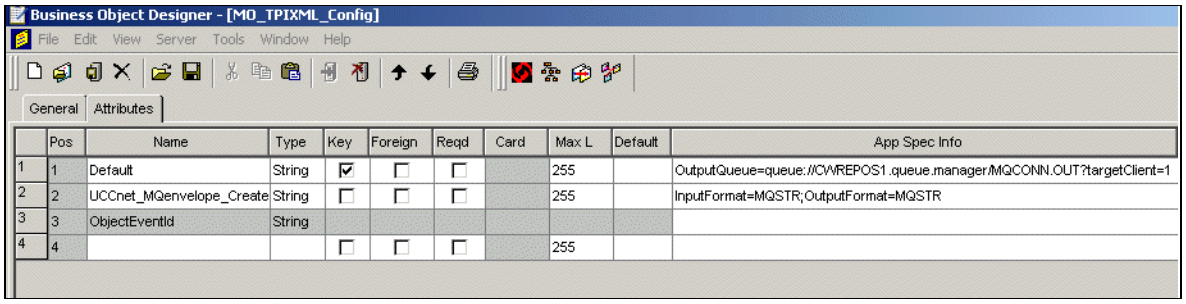
Create a new business object called MO_TPIXML_Config. Select the tab **Attributes** for this business object and add the attributes and values that are listed in Table 4-4 on page 189.

Replace the string <qmgr name> with the actual name of the queue manager. The queue name MQCONN.OUT will be the outbound document queue for the TPI company profile. The name of the attribute UCCnet_MQenvolepe_Create is the name of the business object created previously, suffixed with the verb. This name is case sensitive.

Table 4-4 Properties and values for meta-object MO_TPIXML_Config

Name	Type	Key	App Spec Info
UCCnet_MQenvelope_Create	String		InputFormat=MQSTR;OutputFormat=MQSTR
Default	String	Yes	OutputQueue=queue://<qmgr name>/MQCONN.OUT?targetClient=1

The completed business object is shown in Figure 4-24. Save the business object and close the Business Object Designer.


Business Object Designer - [MO_TPIXML_Config]
File Edit View Server Tools Window Help
General Attributes

	Pos	Name	Type	Key	Foreign	Reqd	Card	Max L	Default	App Spec Info
1	1	Default	String	☑	☐	☐		255		OutputQueue=queue://CWREPOS1.queue.manager/MQCONN.OUT?targetClient=1
2	2	UCCnet_MQenvelope_Create	String	☐	☐	☐		255		InputFormat=MQSTR;OutputFormat=MQSTR
3	3	ObjectEventId	String							
4	4			☐	☐	☐		255		

Figure 4-24 Meta-object for XML documents

4.5.2 Configuring the MQ connector

To support the use of the MQ connector, we need to create a number of queues. Open MQ Explorer and define the following queues:

- MQCONN.REPLY
- MQCONN.UNSUBSCRIBED
- MQCONN.IN_PROGRESS
- MQCONN.ARCHIVE
- MQCONN.ERROR
- MQCONN.IN
- MQCONN.OUT

The naming of these queues is in line with common practices for naming queues that are used by connectors. However, the IN and OUT queues should match the configuration of the TPI company profile.

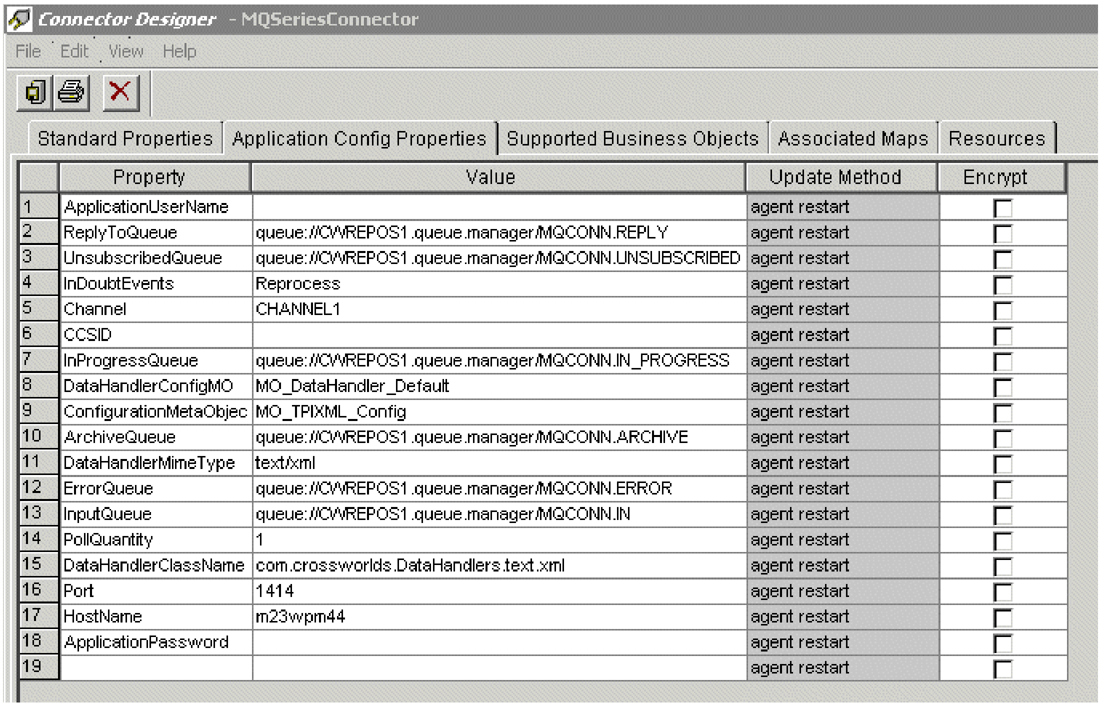
Next, open the MQSeriesConnector object and select the tab **Application Config Properties**. Update the properties as listed in Table 4-5.

Table 4-5 Configuration details for the MQ connector

ReplyToQueue	queue://<qmgr name>/MQCONN.REPLY
UnsubscribedQueue	queue://<qmgr name>/MQCONN.UNSUBSCRIBED
IndoubtEvents	Reprocess
Channel	MQ server channel name, for example CHANNEL1
InProgressQueue	queue://<qmgr name>/MQCONN.IN_PROGRESS
DataHandlerConfigMO	MO_DataHandler_Default
ConfigurationMetaObject	Name of the configuration meta-object

ReplyToQueue	queue://<qmgr name>/MQCONN.REPLY
ArchiveQueue	queue://<qmgr name>/MQCONN.ARCHIVE
DataHandlerMimeType	text/xml
ErrorQueue	queue://<qmgr name>/MQCONN.ERROR
InputQueue	queue://<qmgr name>/MQCONN.IN
DataHandlerClassName	com.crossworlds.DataHandlers.text.xml
Port	Port number on which the MQ listener accepts connections, for example 1414
Hostname	hostname of the computer that runs the queue manager

Figure 4-25 shows the complete set of properties and values for the MQ connector.


Connector Designer - MQSeriesConnector
File Edit View Help
Standard Properties
Application Config Properties
Supported Business Objects
Associated Maps
Resources

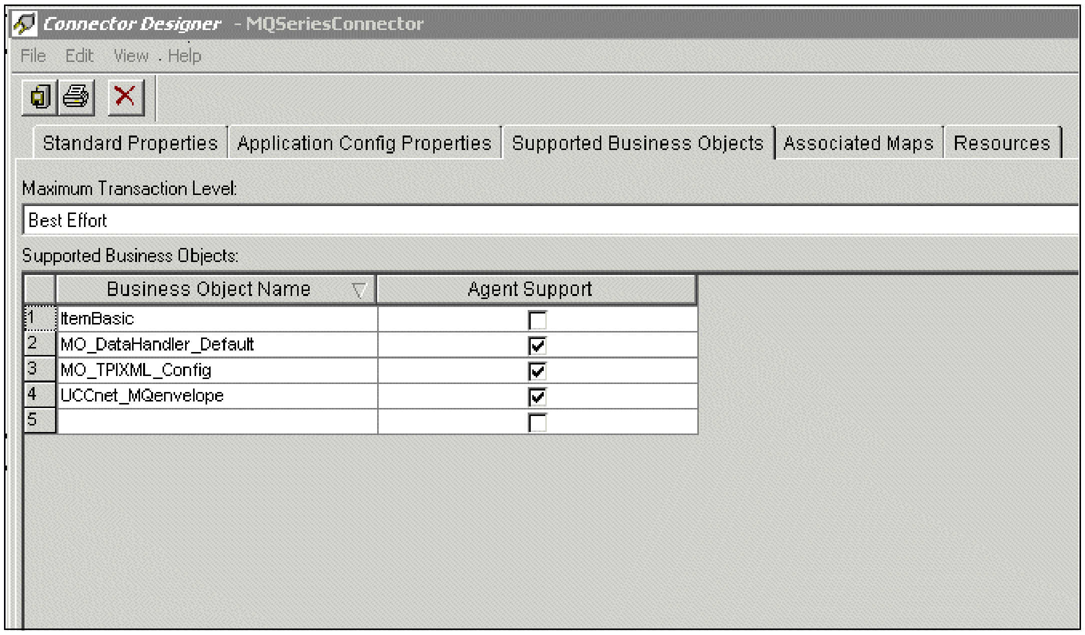
	Property	Value	Update Method	Encrypt
1	ApplicationUserName		agent restart	☐
2	ReplyToQueue	queue://CWREPOS1.queue.manager/MQCONN.REPLY	agent restart	☐
3	UnsubscribedQueue	queue://CWREPOS1.queue.manager/MQCONN.UNSUBSCRIBED	agent restart	☐
4	InDoubtEvents	Reprocess	agent restart	☐
5	Channel	CHANNEL1	agent restart	☐
6	CCSID		agent restart	☐
7	InProgressQueue	queue://CWREPOS1.queue.manager/MQCONN.IN_PROGRESS	agent restart	☐
8	DataHandlerConfigMO	MO_DataHandler_Default	agent restart	☐
9	ConfigurationMetaObjec	MO_TPIXML_Config	agent restart	☐
10	ArchiveQueue	queue://CWREPOS1.queue.manager/MQCONN.ARCHIVE	agent restart	☐
11	DataHandlerMimeType	text/xml	agent restart	☐
12	ErrorQueue	queue://CWREPOS1.queue.manager/MQCONN.ERROR	agent restart	☐
13	InputQueue	queue://CWREPOS1.queue.manager/MQCONN.IN	agent restart	☐
14	PollQuantity	1	agent restart	☐
15	DataHandlerClassName	com.crossworlds.DataHandlers.text.xml	agent restart	☐
16	Port	1414	agent restart	☐
17	HostName	m23wpm44	agent restart	☐
18	ApplicationPassword		agent restart	☐
19			agent restart	☐

Figure 4-25 Application Config Properties for the MQ connector

Switch now to the tab Supported Business Objects and add the following business objects to the list:

- ItemBasic
- MO_DataHandler_Default
- MO_TPIXML_Config
- UCCnet_MQenvelope

Set the flag for Agent Support, as shown in Figure 4-26 on page 191.


Connector Designer - MQSeriesConnector
File Edit View Help
Standard Properties
Application Config Properties
Supported Business Objects
Associated Maps
Resources

Maximum Transaction Level:
Best Effort

Supported Business Objects:

	Business Object Name	Agent Support
1	ItemBasic	☐
2	MO_DataHandler_Default	✓
3	MO_TPIXML_Config	✓
4	UCCnet_MQenvelope	✓
5		☐

Figure 4-26 Supported business objects for the MQ connector

Save and close the MQ connector.

4.5.3 Creating maps

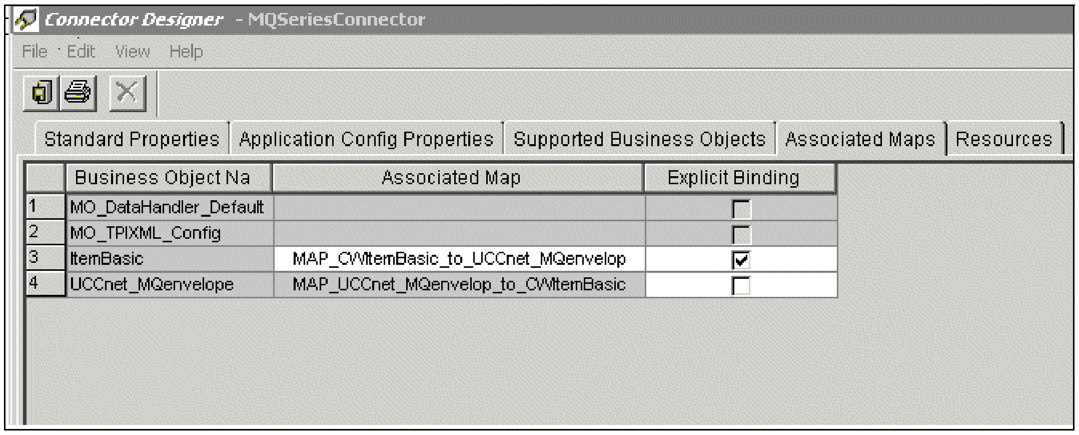
Open the System Manager, right-click the folder **Maps** and select **New**. Choose **ItemBasic** as the source business object and **UCCnet_MQenvelope** as the destination business object. Name the map MAP_CWItemBasic_to_UCCnet_MQenvelope.

Map the attributes as listed in the Table 4-6.

Table 4-6 Mapping statements for the map MAP_CWItemBasic_to_UCCnet_MQenvelope.

Source Attribute	Target Attribute	Transformation rule
ItemId	uniqueCreatorIdentification	Move
	TLO.messageHeader.from.globalLocationNumber.gln	Set value(Retailer1)
	TLO.messageHeader.to.globalLocationNumber.gln	Set value(Supplier)
	XMLDeclaration	Set value(“xml version=\”1.0””)
	DocType	Set Value(“DOCTYPE envelope SYSTEM \”http://cert.uccnet.net/xmlschema/2.0/Envelope.dtd\””)

Select the tab Verbs for this map and set the value `Create` for the column Verb. Save and compile the map. Re-open the connector MQSeriesConnector and select the tab **Associated Maps**. This time, the new map should be listed for the business object ItemBasic. If it does not show, select the check box **Explicit Binding** and choose the correct box in the drop-down menu under the heading Associated Map.


Connector Designer - MQSeriesConnector

File Edit View Help

Standard Properties | Application Config Properties | Supported Business Objects | Associated Maps | Resources

	Business Object Na	Associated Map	Explicit Binding
1	MO_DataHandler_Default		☐
2	MO_TPIXML_Config		☐
3	ItemBasic	MAP_CWItemBasic_to_UCCnet_MQenvelop	☑
4	UCCnet_MQenvelope	MAP_UCCnet_MQenvelop_to_CWItemBasic	☐

Figure 4-27 Associated maps for the MQ connector

4.5.4 Updating the collaboration object

You can either update the collaboration that we created in 4.4.2, “Collaboration object definition and customization” on page 173 or create a new object based on the same template. to create a new collaboration object:

1. In the System Manager, right-click the folder **Collaboration Objects** and select **New Collaboration Object**. The collaboration creation wizard will appear.
2. Select the collaboration template **UCCNet_ItemSync** and name the new collaboration object UCCnet_ItemSync_MQ_CO.
3. Bind the ports in the same way as was done for the first scenario. However, this time bind the To port to the MQ connector.

Your collaboration diagram should look as shown in Figure 4-28.

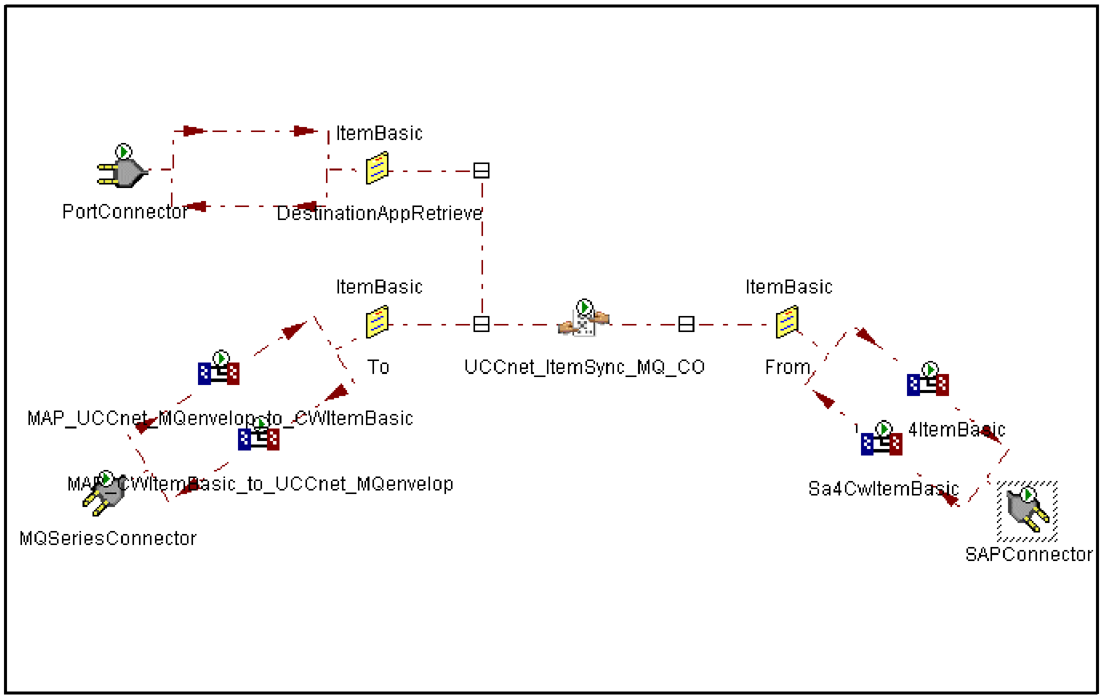


Figure 4-28 Collaboration object diagram with MQ connector

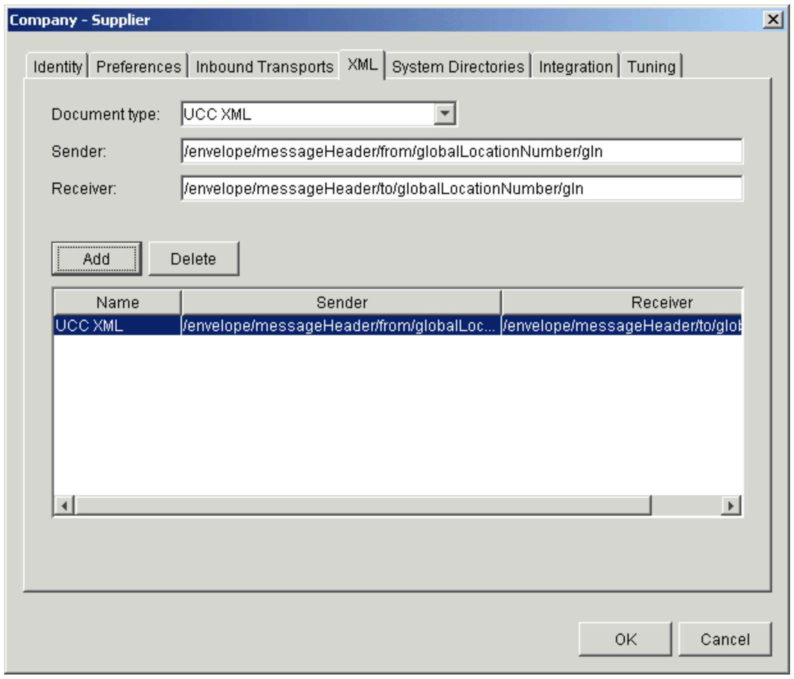
4.5.5 Updating the TPI Server configuration

1. Start the TPI Administrator tool and open the company profile. Select the tab **XML** and set `UCC XML` as the document type. Modify the fields Sender and Receiver as shown below.

   – **Sender:** /envelope/messageHeader/from/globalLocationNumber/gln

   – **Receiver:** /envelope/messageHeader/to/globalLocationNumber/gln

2. Click **Add** (see Figure 4-29).

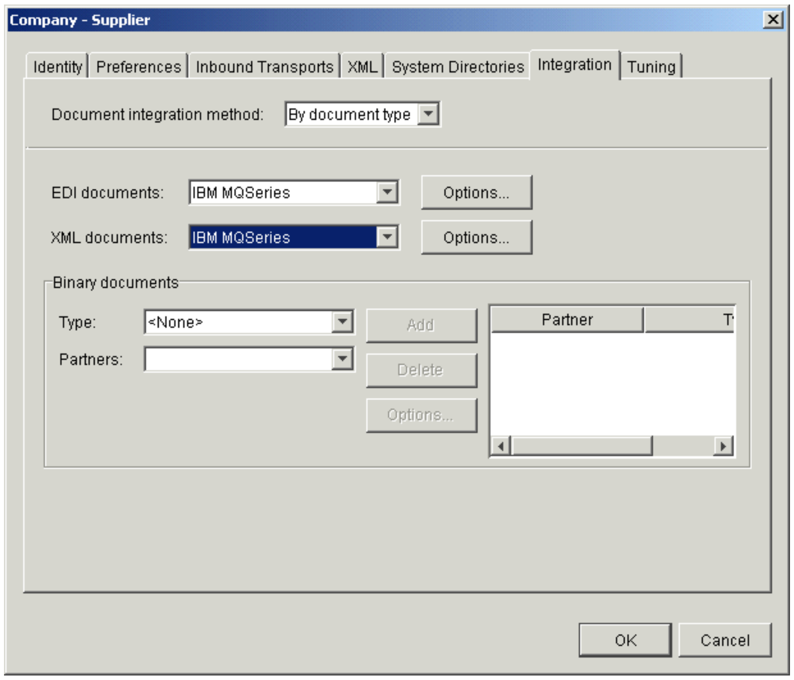


Document type: UCC XML
Sender: /envelope/messageHeader/from/globalLocationNumber/gln
Receiver: /envelope/messageHeader/to/globalLocationNumber/gln

Name	Sender	Receiver
UCC XML	/envelope/messageHeader/from/globalLoc...	/envelope/messageHeader/to/glol

Figure 4-29 Configuring XML support in TPI

3. Select the tab **Integration** and choose **By document type** as the Document integration method. Select **IBM MQSeries** for the field XML documents (see Figure 4-30 on page 194).


Company - Supplier

Identity | Preferences | Inbound Transports | XML | System Directories | Integration | Tuning

Document integration method: By document type

EDI documents: IBM MQSeries

XML documents: IBM MQSeries

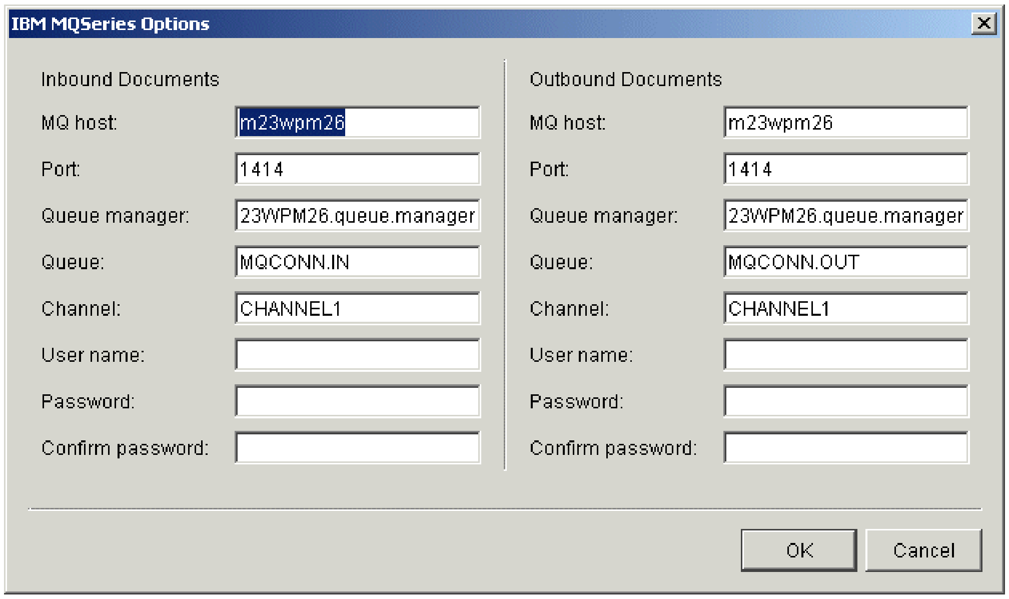
Binary documents

Type: <None>

Partners:

Figure 4-30 Add WebSphere MQ as an integration option for XML

4. Click **Options** for XML documents to bring up the window that allows you to configure the integration with WebSphere MQ.
5. As shown in Figure 4-31, provide the required parameters to interact with WebSphere MQ for inbound and outbound documents. Note that the communication is through MQ client channels.


IBM MQSeries Options

Inbound Documents		Outbound Documents	
MQ host:	m23wpm26	MQ host:	m23wpm26
Port:	1414	Port:	1414
Queue manager:	23WPM26.queue.manager	Queue manager:	23WPM26.queue.manager
Queue:	MQCONN.IN	Queue:	MQCONN.OUT
Channel:	CHANNEL1	Channel:	CHANNEL1
User name:		User name:	
Password:		Password:	
Confirm password:		Confirm password:	

Figure 4-31 Setting integration options for WebSphere MQ

6. Save the profile by clicking **OK**.

4.5.6 Running the test scenario

Run a test using the Test connector as described previously to validate the setup of the MQ connector. This time, the XML documents that are prepared by the ICS should pass to the TPI Server via the MQ connector. However, the actual XML documents should look as was shown earlier in Figure 4-19 on page 184 and in the following figures.

4.6 Implementation of scenario 3

In this section, we describe the use of iSoft's P2PAgent as the AS/2 provider. Typically, a customer will standardize on a single As/2 provider. While the ItemSync collaboration was developed for use with TPI, it can be configured to work with iSoft's P2Pagent as well.

The overall flow is shown in Figure 4-32. We will continue to use the SAP connector and test the setup via the Test connector. The collaboration itself is not different from the collaboration that was developed earlier in this chapter. The MQ connector in the diagram is the connector that was configured in the previous section. We are now going to configure iSoft's P2PAgent to pick up the XML documents from the same queue that was used by the TPI connector. The P2PAgent program will then be able to send the XML document to UCCnet, receive any responses from UCCnet and store those responses in an MQ queue.

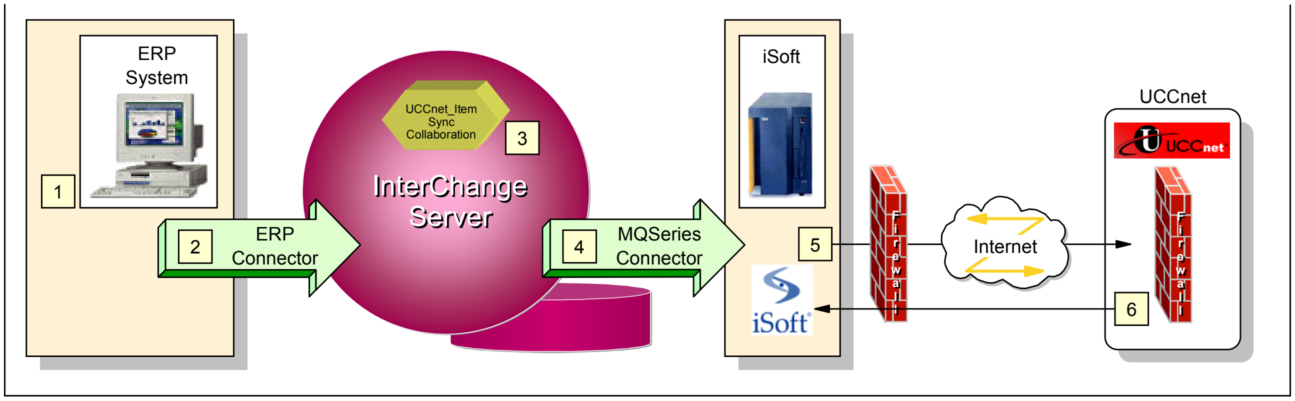


Figure 4-32 UCCnet solution with iSoft's P2PAgent

4.6.1 Configuration of iSoft's P2PAgent

The configuration file p2pagent.cfg that is used to configure iSoft's P2PAgent needs to be updated to reflect the use of queues. Also, you should be careful to use those queues that are used by the MQ connector. The **addpair** command in Example 4-1 that ties UCCnet to the P2PAgent of this company refers to the queue MQCONN.IN that is also used by the MQ connector.

For the other destination parameters, you can use files or queues, as shown in Example 4-1.

Example 4-1 p2pagent.cfg with mq queues

```
<xml>

#  configuration settings
<command>set -eperror -ef</command>
<command>set -lplog -lf</command>
<command>set -npmq://ISOFT.M23WPK60.QMANAGER/NOTICES    -nf      </command>
<command>set -opmq://ISOFT.M23WPK60.QMANAGER/WORKORDERS -of -oswo </command>
<command>set -pppki</command>
```

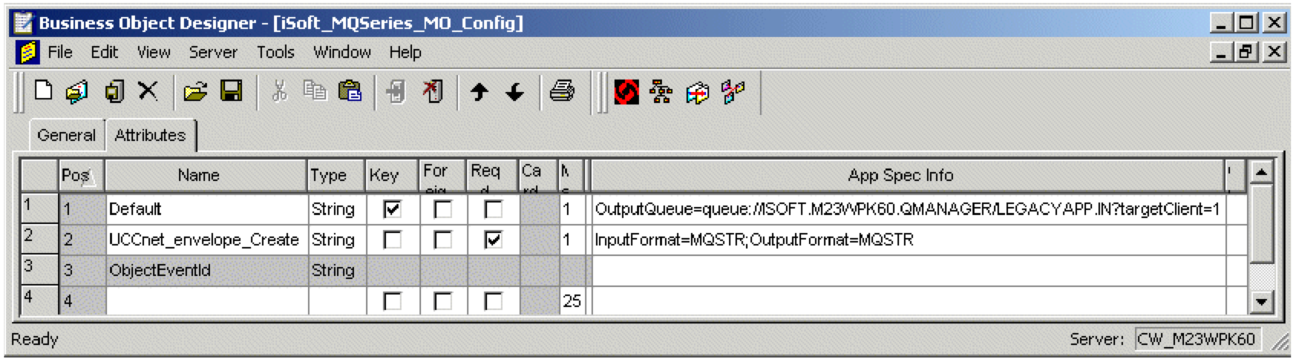
```
<command>set -rpmq://ISOFT.M23WPK60.QMANAGER/RECEIPTS             </command>
<command>set -tr300s</command>
<command>set -bhmq://ISOFT.M23WPK60.QMANAGER                      </command>


#  iSoft Testing Relationship
<command>addpair Supplier1 UCCnet http://9.24.105.161:4080/ http://9.24.104.115:4080/ Supplier1 inbox
</command>
<command>addpair UCCnet Supplier1 http://9.24.104.115:4080/   *    UCCnet
mq://ISOFT.M23WPK60.QMANAGER/MQCONN.IN</command>
 </command>

#  iSoft certificates and keys
<command>importkey Supplier1 UCCnet E -fCpki\Supplier1.cer -fKpki\Supplier1.prv</command>
<command>importkey Supplier1 UCCnet J -fCpki\UCCnet.cer</command>
<command>importkey UCCnet Supplier1 E -fCpki\Supplier1.cer -fKpki\Supplier1.prv</command>
<command>importkey UCCnet Supplier1 J -fCpki\UCCnet.cer</command>


#
#  start services
<command>start http://9.24.104.115:4080/</command>
<command>send http iSoft UCCnet -de -dsMAILBOXID=MQCONN.OUT -tC30s -tE20031231000000 -e -sC -r1
-cX</command>
</xml>
```

While it is not strictly required, we preferred to create a separate MQ configuration meta-object, called iSoft_MQSeries_MO_config. Create this business object using the Business Object Designer and add two attributes to it. Those attributes and their values are listed in Table 4-7.

Table 4-7 Attributes and values for the configuration meta-object

Name	Type	Key	App Spec Info
Default	String	yes	OutputQueue=queue://ISOFT.M23WPK60.QMANAGER/MQCONN.OUT?targetClient=1
UCCnet_envelope_Create	String		InputFormat=MQSTR;OutputFormat=MQSTR

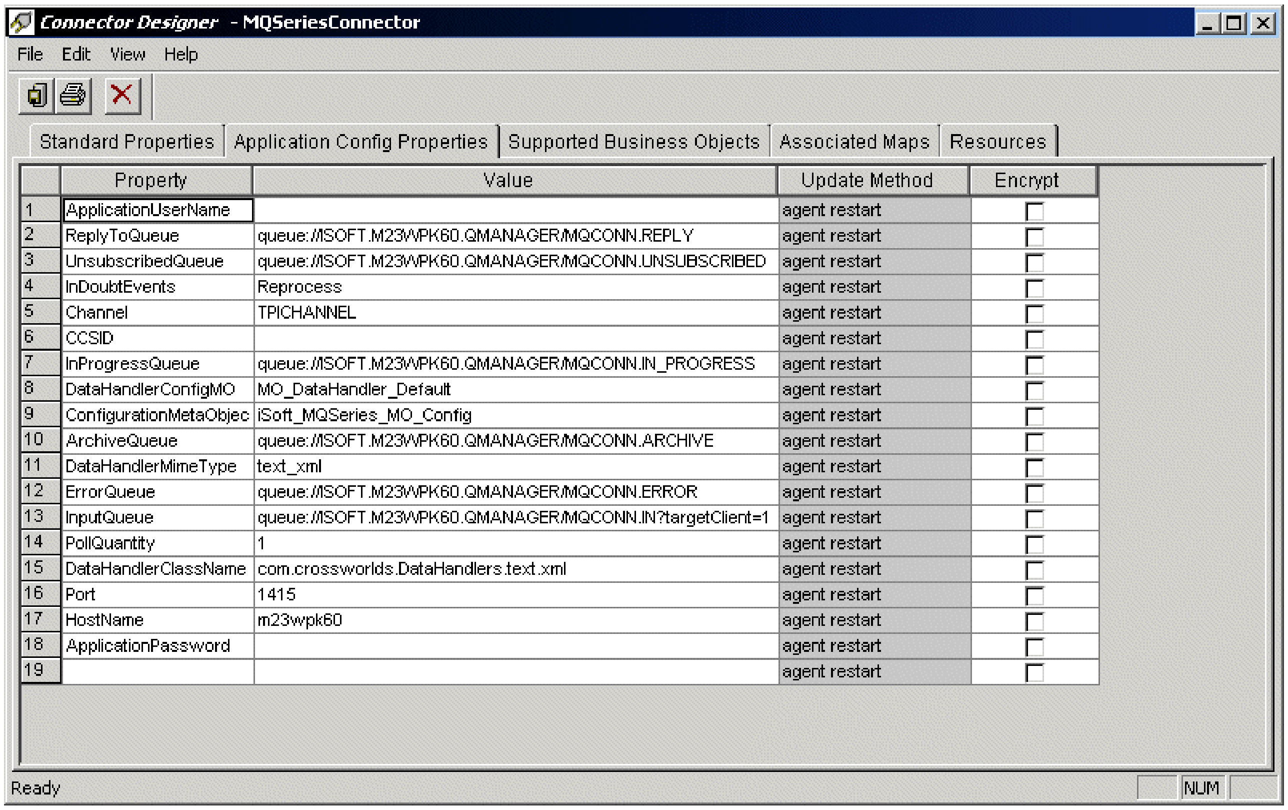
The attribute Default refers to the name of the queue manager and the name of the queue; these values should match the values of the **addpair** command in the P2PAgent's configuration file. Figure 4-33 shows the completed business object.


Business Object Designer - [iSoft_MQSeries_MO_Config]
File Edit View Server Tools Window Help
General Attributes
Ready
Server: CW_M23WPK60

	Pos	Name	Type	Key	For eig	Req d	Ca rd	M s	App Spec Info
1	1	Default	String	✓				1	OutputQueue=queue://ISOFT.M23WPK60.QMANAGER/LEGACYAPP.IN?targetClient=1
2	2	UCCnet_envelope_Create	String			✓		1	InputFormat=MQSTR;OutputFormat=MQSTR
3	3	ObjectEventId	String						
4	4							25	

Figure 4-33 Meta-object for use by the MQ connector

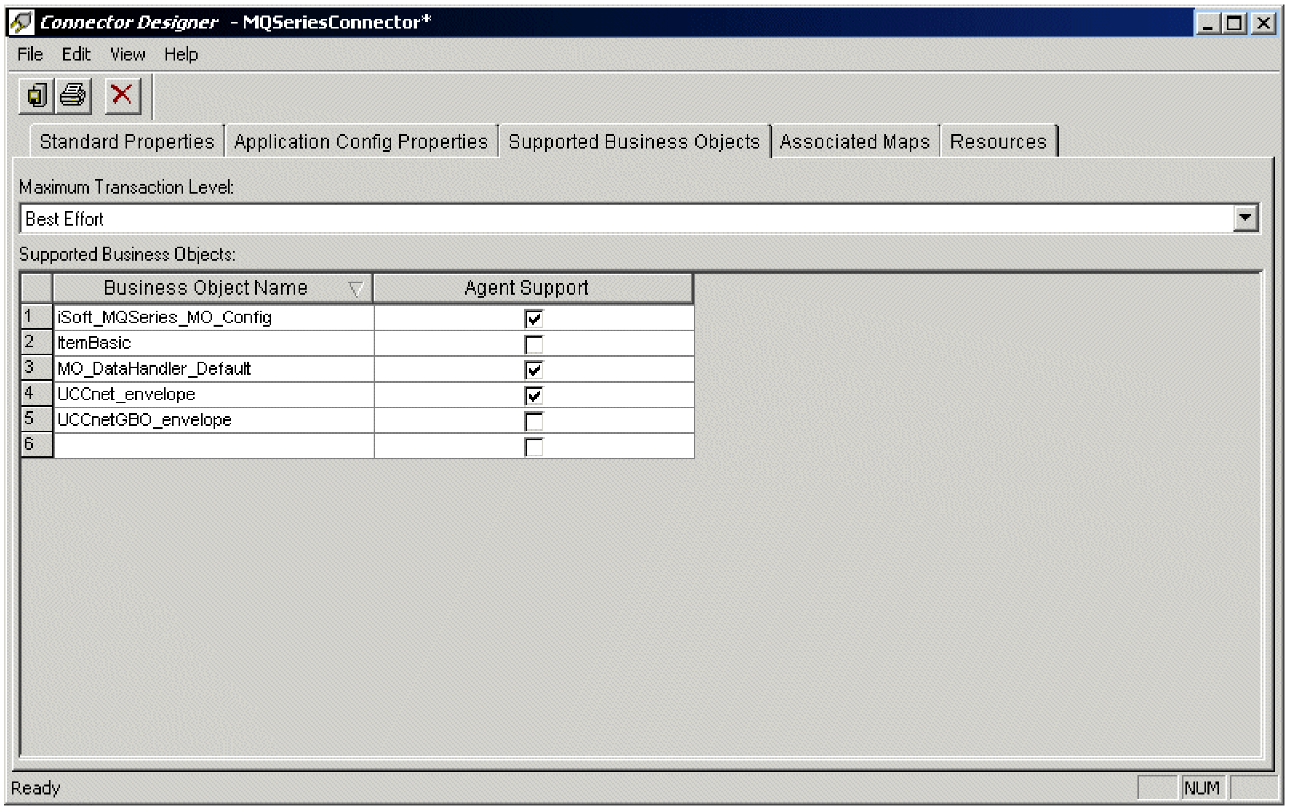
Assuming that the MQ connector uses different MQ objects to interact with iSoft's P2PAgent, you need to update the MQ connector configuration. Figure 4-34 shows the tab Application Config Properties of the MQ connector where the queue names match the settings in the P2PAgent's configuration file. It also refers to the configuration meta-object iSoft_MQSeries_MO_Config which we created previously.


Connector Designer - MQSeriesConnector
File Edit View Help
Standard Properties
Application Config Properties
Supported Business Objects
Associated Maps
Resources
Ready
NUM

	Property	Value	Update Method	Encrypt
1	ApplicationUserName		agent restart	☐
2	ReplyToQueue	queue://ISOFT.M23WPK60.QMANAGER/MQCONN.REPLY	agent restart	☐
3	UnsubscribedQueue	queue://ISOFT.M23WPK60.QMANAGER/MQCONN.UNSUBSCRIBED	agent restart	☐
4	InDoubtEvents	Reprocess	agent restart	☐
5	Channel	TPICHANNEL	agent restart	☐
6	CCSID		agent restart	☐
7	InProgressQueue	queue://ISOFT.M23WPK60.QMANAGER/MQCONN.IN_PROGRESS	agent restart	☐
8	DataHandlerConfigMO	MO_DataHandler_Default	agent restart	☐
9	ConfigurationMetaObjec	iSoft_MQSeries_MO_Config	agent restart	☐
10	ArchiveQueue	queue://ISOFT.M23WPK60.QMANAGER/MQCONN.ARCHIVE	agent restart	☐
11	DataHandlerMimeType	text_xml	agent restart	☐
12	ErrorQueue	queue://ISOFT.M23WPK60.QMANAGER/MQCONN.ERROR	agent restart	☐
13	InputQueue	queue://ISOFT.M23WPK60.QMANAGER/MQCONN.IN?targetClient=1	agent restart	☐
14	PollQuantity	1	agent restart	☐
15	DataHandlerClassName	com.crossworlds.DataHandlers.text.xml	agent restart	☐
16	Port	1415	agent restart	☐
17	HostName	m23wpk60	agent restart	☐
18	ApplicationPassword		agent restart	☐
19			agent restart	☐

Figure 4-34 Properties of the MQ connector

Since we have used a different configuration meta-object, we need to make sure that this business object is added to the list of supported business objects of this MQ connector. Figure 4-35 on page 198 shows us the MQ connector's Supported Business Objects tab.


Connector Designer - MQSeriesConnector*
File Edit View Help
Standard Properties
Application Config Properties
Supported Business Objects
Associated Maps
Resources
Ready
NUM

Maximum Transaction Level: Best Effort

Supported Business Objects:

	Business Object Name	Agent Support
1	iSoft_MQSeries_MO_Config	✓
2	ItemBasic	
3	MO_DataHandler_Default	✓
4	UCCnet_envelope	✓
5	UCCnetGBO_envelope	
6		

Figure 4-35 Supported business objects for the MQ connector

This concludes the setup of the MQ connector. After restarting the MQ connector, we can execute the same test process as shown earlier in 4.4.8, “Running the test scenario” on page 181.

4.7 Conclusion

In Chapter 3, “Implementing multi-product AS/2 communication with trading partners” on page 101, we described how trading partners with different AS/2 products can be connected to each other. The choice of one business partner for a given product does not imply that all its partners need to use the same product. We described what settings were required to make iSoft’s P2PAgent interoperable with TPI.

In this chapter, we described how iSoft’s P2PAgent can be used as well as TPI for integration with the InterChange Server and any back-office application systems. By configuring an MQ connector and binding this MQ connector to the collaboration, we can set up AS/2 communication with iSoft’s P2PAgent. This proves that an integration solution built for one AS/2 provider can work equally well for another AS/2 provider. As such, it shows the interoperability of iSoft’s P2PAgent and TPI both within the company and between trading partners.

5

Implementing a back-up solution using IBM Expedite

Using Internet technology for the transfer of EDI documents is cost-effective, but it also means that a company is relying on technology for which there is no (or at least less) guaranteed reliability. Therefore, it might be required for an organization to define a back-up strategy for those situations where the Internet is not offering a reliable connection. In this chapter, we discuss the use of IBM Expedite and the IBM VAN and how it can be combined with iSoft to create a solution with full reliability.

5.1 Introduction

Usage of a back-up communication method for iSoft is strongly recommended and allowed for the iSoft product. One such method is using Expedite Base for Windows along with Information Exchange. Using a personal computer, a modem, Expedite Base for Windows and Information Exchange, you can have a complete back-up solution for your data transmissions to your trading partners.

Expedite Base for Windows and Information Exchange provide for an excellent low cost back-up solution for iSoft. When iSoft fails to connect to the receiving trading partner via the Internet, it will write out the file to a given directory or to a message queue. Using this directory, or message queue, we are able to gather the unsent files and send them directly using Expedite Base for Windows and Information Exchange. These dual paths are separate from one another so you can maintain your connectivity even during Internet outages.

In order to use Expedite Base for Windows, you must have an Information Exchange mailbox account, a user ID and a password. Expedite Base can only receive and send messages from the Information Exchange system.

5.2 Expedite Base for Windows installation

Expedite Base for Windows is available over the Web at no charge from IBM. There is a registration which you must complete, but that is all. To install Expedite Base, perform the following steps:

1. Download the *Expedite Base for Windows Programming Guide* from:

   `http://ieas.services.ibm.com/expedite/publications.shtml`

   While this is not required, it will come in handy later for reference and error codes.

2. Download the Expedite Base for Windows 4.6.2 product code from:

   `https://www6.software.ibm.com/dl/expv2/expv2-p`

   You must complete the registration form if you have not done so before. Notice we chose Expedite Base for Windows 4.6.2. This is due to the fact that we want to automate this process later. Selecting Expedite for Windows will not allow you to automate the process.

3. Install the software according to the instructions. Make sure you choose the TCP/IP option.

4. Edit the file win.ini found in your Windows directory. Change the AutoMode value from `N` to `Y`. This will allow Expedite Base to run fully automatically. Otherwise, it will pause and wait for you to select **File -> Start**. An example of the win.ini file is shown in Example 5-1.

Example 5-1 Sample file win.ini

```
[Expedite Base]
AutoMode=Y
MainWindow=Show
WindowSize=201,27,849,523
DialDelay=3
FileNameFormat=0
```

5. Add the install directory for Expedite to the Windows PATH environment variable.

   a. Click **Start -> Settings -> Control Panel**.

   b. In the Control Panel window, click **System**.

   c. In the System window, click **Advanced**.

   d. In the Advanced window, click **Environment Variables**.
   e. In System Variables, select the **PATH** variable and then click **Edit**.
   f. Add the Expedite directory to the PATH variable. For example, `C:\Expedite;`
   g. Click **OK**.

6. Two files must be copied from the C:\Expedite\Samples directory to the C:\Expedite directory.
   – C:\Expedite\Samples\BASEMSG.IN **->** C:\Expedite\BASEIN.MSG
   – C:\Expedite\Samples\TCPDSAMP.PRO **->** C:\Expedite\BASEIN.PRO

   Note: The files must also be renamed and edited to match your communication method.

7. Edit the file C:\Expedite\BASEIN.PRO. This will contain your Information Exchange account, user ID and password. Edit the IDENTIFY section of the file so it is correct. The TRANSMIT section will set your communication method. The default is a TCP/IP leased line. For this redbook, we will be using TCP/IP dial. Therefore, we need to change `COMMTYPE(T)` to `COMMTYPE(C)`. A sample file is shown in Example 5-2.

   For more information on the profile commands, see the "Using Expedite Base for Windows Profile commands" chapter in the *Expedite Base for Windows Programming Guide*.

Example 5-2 Sample file BASEIN.PRO

```
IDENTIFY                    # provide information for logon
     IEACCOUNT(IEACCT)  # replace "ieacct" with your IE account
     IEUSERID(IEUSER01)      # replace "ieuser01" with your IE user ID
     IEPASSWORD(IEPASSS)     # replace "iepass" with your IE password
     ;

TRANSMIT
     COMMTYPE(C)            # Connect using TCP/IP dialed line connection.
     AUTOSTART(Y)
     AUTOEND(Y)
     RECONNECT(5)
     RECOVERY(C)
     ;

TCPCOMM
     DIALPROFILE(IEACCT)
     ;
```

8. Edit C:\Expedite\BASEIN.MSG. This is where the actual commands go. As an example, it could look like:

   ```
   SENDEDI FILEID(C:\EDIDataNotSent\edidatatobesent.txt);
   ```

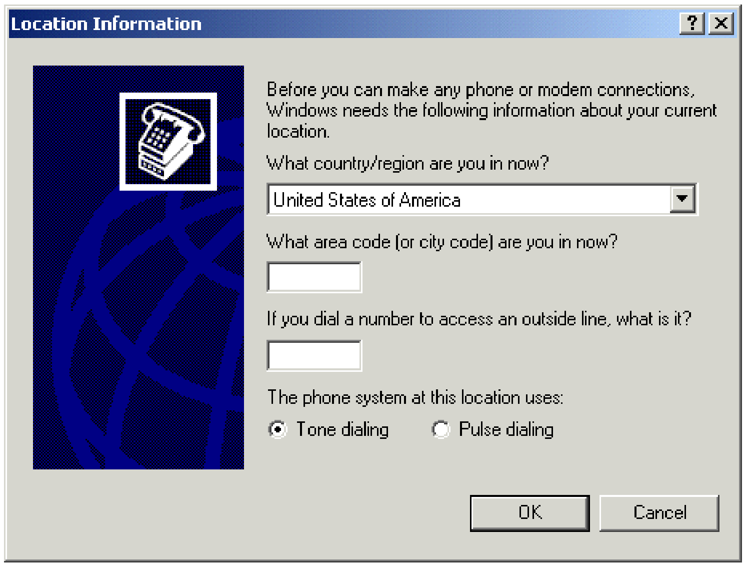
   For all the commands and syntax, see the "Using Expedite Base for Windows message commands" chapter in the *Expedite Base for Windows Programming Guide*.

5.3 AT&T Global Network Dialer installation

When Expedite Base for Windows is installed, AT&T Global Network Dialer will also be installed since the TCP/IP option was chosen during the installation of Expedite itself. We recommend that you set up the AT&T Global Network Dialer first, then let Expedite Base for

Windows use it. To do this, start the AT&T Global Network Dialer by selecting **Start -> Programs -> AT&T Global Network -> AT&T Global Network Dialer**.

1. When starting the dialer for the very first time, it will prompt you to perform a number of setup tasks. In the first window, shown in Figure 5-1, select your country/region. Fill in the area code from which you will be dialing into Information Exchange and if a number is needed to reach an outside line, complete the appropriate field. Also, you may select **Tone dialing** or **Pulse dialing**. Then click **OK**.



Before you can make any phone or modem connections, Windows needs the following information about your current location.

What country/region are you in now?

United States of America

What area code (or city code) are you in now?

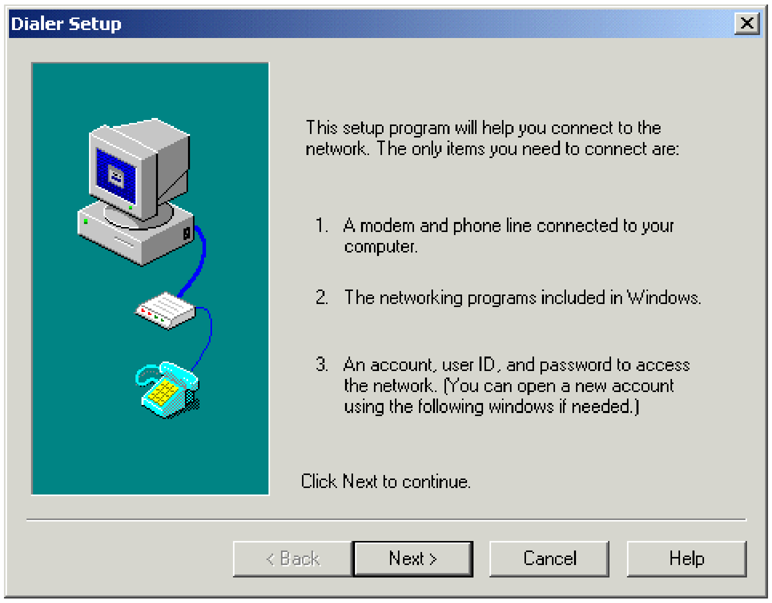
If you dial a number to access an outside line, what is it?

The phone system at this location uses:

[✓ Tone dialing] Pulse dialing

Figure 5-1 AT&T Global Network Dialer install program

2. Figure 5-2 shows the requirements for AT&T Global Network Dialer. Item 3 is the account, user ID and password on the Information Exchange system. Review this and make sure you meet these requirements, then click **Next**.



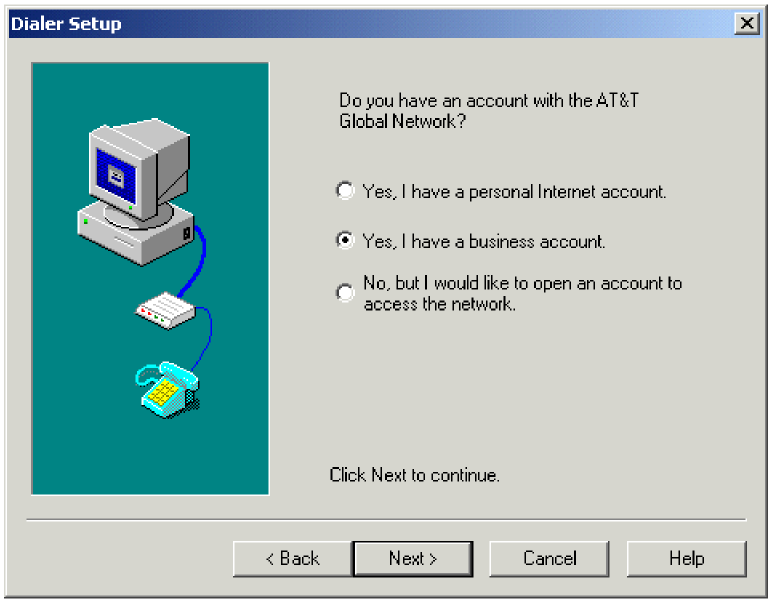
This setup program will help you connect to the network. The only items you need to connect are:

1. A modem and phone line connected to your computer.
2. The networking programs included in Windows.
3. An account, user ID, and password to access the network. (You can open a new account using the following windows if needed.)

Click Next to continue.

Figure 5-2 Requirements window for AT&T Global Network Dialer

3. As shown in Figure 5-3, you want to be sure and select the option **Yes, I have a business account**. Then click **Next**.


Dialer Setup

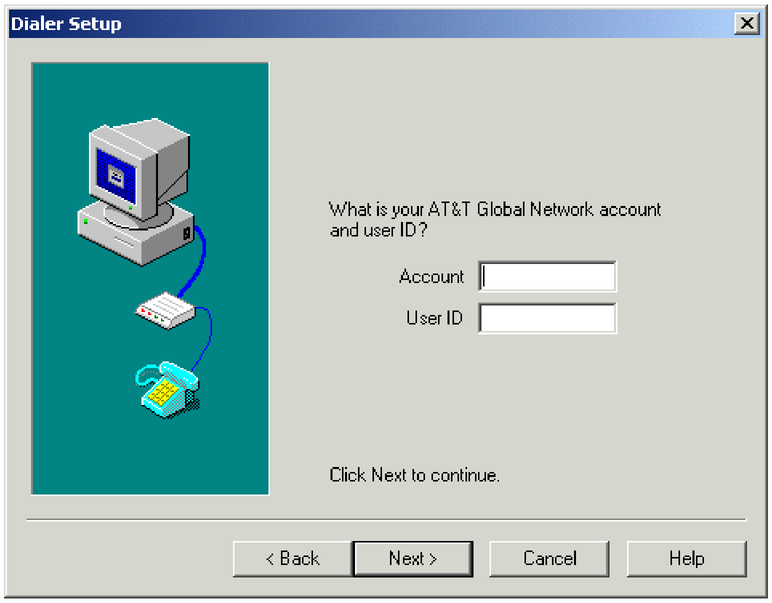
Do you have an account with the AT&T Global Network?

Yes, I have a personal Internet account. [✓ Yes, I have a business account.] No, but I would like to open an account to access the network.

Click Next to continue.

Figure 5-3 AT&T Global Network Dialer account type window

4. In the account and user ID window, shown in Figure 5-4, enter your Information Exchange Account and user ID. If you do not have those, then click **Cancel** and run the install when you do have the proper information. After completing this, click **Next**.


Dialer Setup

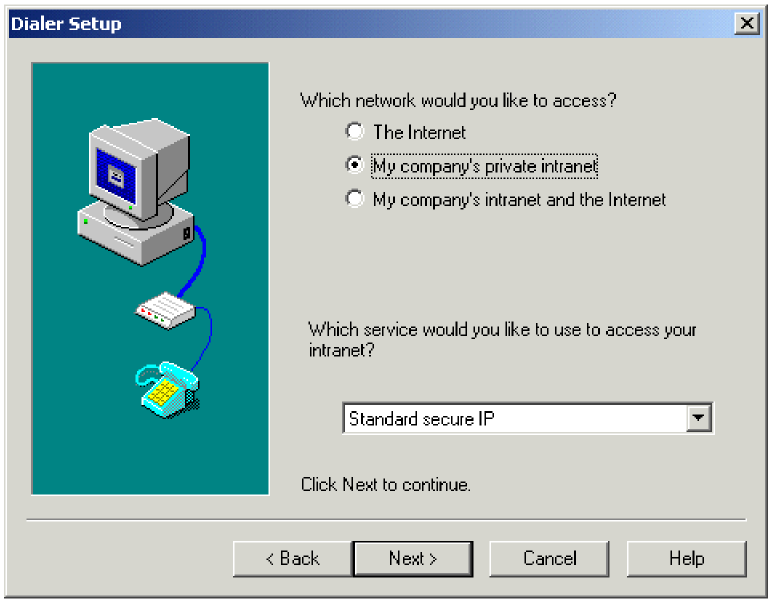
What is your AT&T Global Network account and user ID?

Account:
User ID:

Click Next to continue.

Figure 5-4 Provide your account and user ID from the Information Exchange system

5. As in Figure 5-5 on page 204, select **My company's private intranet** for the question `Which network would you like to access?` While this account and ID may not be for your company's private intranet, it is for IBM. Based upon your account and user ID, AT&T will make sure that you get connected to the right system. The default Standard secure IP is correct for the services option. Click **Next**.



Which network would you like to access?

( ) The Internet
(●) My company's private intranet
( ) My company's intranet and the Internet

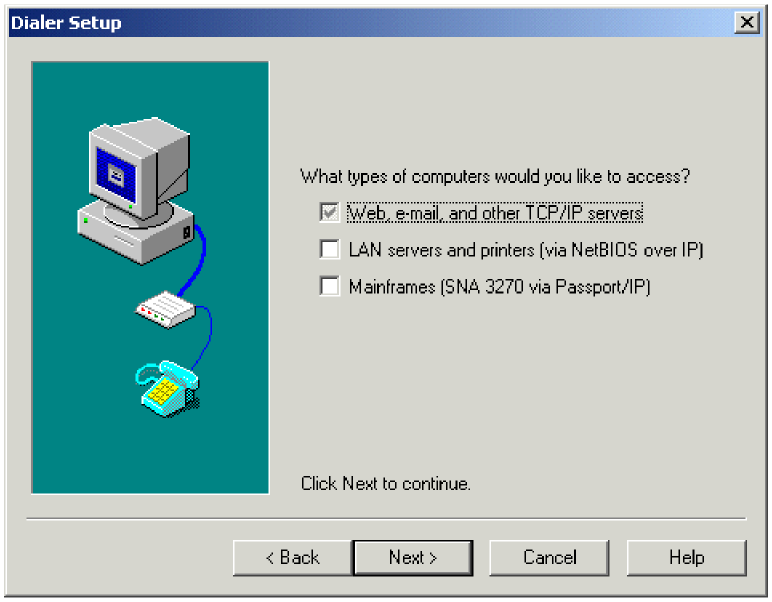
Which service would you like to use to access your intranet?

Standard secure IP

Click Next to continue.

Figure 5-5 AT&T Global Network Dialer access type window

6. Select **Web, e-mail, and other TCP/IP servers** as the type of computers you would like to access, as shown in Figure 5-6. In this redbook, we are using TCP/IP to reach Information Exchange. Click **Next**.



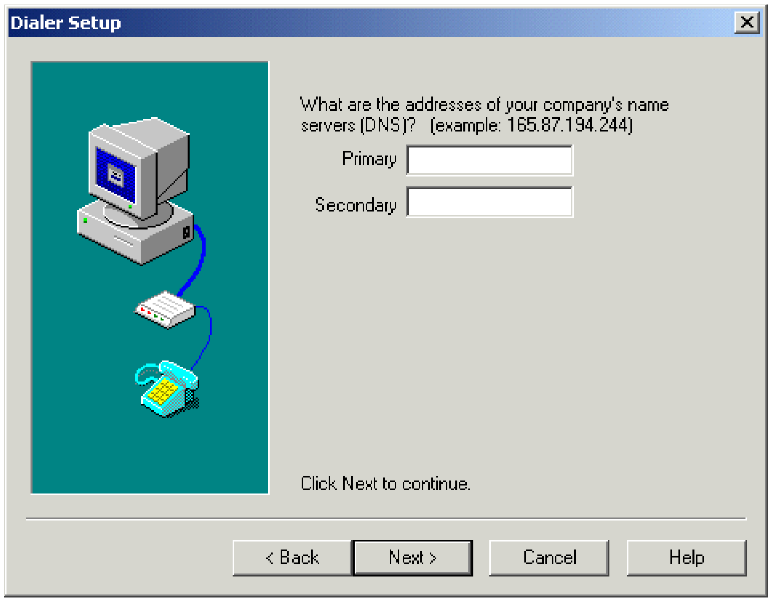
What types of computers would you like to access?

[✓] Web, e-mail, and other TCP/IP servers
[ ] LAN servers and printers (via NetBIOS over IP)
[ ] Mainframes (SNA 3270 via Passport/IP)

Click Next to continue.

Figure 5-6 AT&T Global Network Dialer system types you need access to

7. Figure 5-7 on page 205 shows the window where you can set information about domain name servers (DNS). It is optional to fill this out and you should only do so if you know the information. If you do not know the DNS for your company, then leave the fields blank and click **Next**.



Dialer Setup

What are the addresses of your company's name servers (DNS)? (example: 165.87.194.244)

Primary

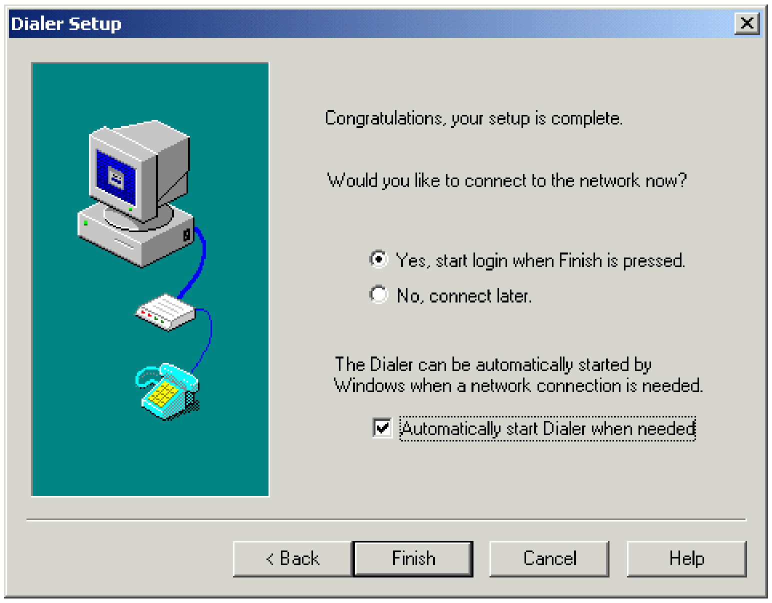
Secondary

Click Next to continue.

Figure 5-7 AT&T Global Network Dialer optional name server window

8. As shown in Figure 5-8, the setup is completed. Make sure you select **Yes, start login when Finish is pressed** and **Automatically start Dialer when needed**. Then click **Finish**.

> **Note:** The first time you connect, many downloads and updates will occur. This is normal for the first time. This could take 20 minutes or more to complete. Also, as you sign on, select the box **Save Password**.



Dialer Setup

Congratulations, your setup is complete.

Would you like to connect to the network now?

[● Yes, start login when Finish is pressed.]

○ No, connect later.

The Dialer can be automatically started by Windows when a network connection is needed.

[✓ Automatically start Dialer when needed]

Figure 5-8 AT&T global network dialer completion window

5.4 Integrating iSoft and Expedite

The configuration of iSoft needs an additional command in order to produce a usable file which Expedite Base for Windows can send to Information Exchange. The additional command is:

```
<command> set -yprecycle -yf</command>
```

where `recycle` is the name of a directory.

This command can be placed in the usual configuration file. The P2PAgent also generates an error file (or message), but that error file contains a dump of the HTML buffer. It contains HTML specific information and the actual data can be in an encrypted format.

If you would like to use WebSphere MQ to move the data, you will need to add this command instead:

```
<command> set -ypmq://queue_manager_name/queue_name -yf</command>
```

For example:

```
<command> set -ypmq://WDI/recycle_for_expedite -yf</command>
```

Using this messaging queue, you can send it directly into Information Exchange using the MQSeries Services/Information Exchange Bridge. The document *IBM MQSeries Services Administration and Application Development Guide,* in the chapter “MQSeries Services/Information Exchange Bridge”, covers this in detail. Since this is outside the scope of this redbook, it will not be covered.

5.5 Case study

Company Supplier is sending and receiving EDI data with several retailers. One retailer has requested the usage of the Internet and EDI-INT AS2 technology instead of the traditional Value Add Network (VAN). Because the Internet may not always be available, it is required that Information Exchange be used if that is the case. Also, if the company Supplier is not reachable from a retailer, then that retailer may use the VAN to send the data. Since company Supplier currently does not use Information Exchange, what does this mean to them?

The first question is: how much data could be sent out using this back-up method? If it is more than 3.6 MB in 20 minutes of time, then this solution will not work. The 3.6 MB number was obtained by taking 20 minutes times 60 seconds times 3 KB per second. This works out to be 3.6 MB. The 20 minutes of time value was chosen as the time interval between sending runs. The 3 KB per second value is the upload speed expected from a 56 Kb modem. How much is 3.6 MB of data? If an average segment is 60 characters long, then the file would be 60,000 lines long.

Assuming that we can live with a 3.6 megabyte limit per 20 minutes, let us lay out the solution. A personal computer (running Windows 95, 98, NT 4.0 or Windows 2000), a modem, an analog phone line and an account on Information Exchange would be required. The modem has to be supported by Windows and support a 56 Kb connection. It may be an internal or external modem.

Since company Supplier does not have an Information Exchange account, it will need to call IBM and request an account/user ID on the Information Exchange system. The phone number is (800) 655-8865 and the e-mail address is edihelp@us.ibm.com®.

5.5.1 Sending data from the supplier to the customer

If iSoft's P2PAgent fails to connect to the receiving party, it will place the original file in the recycle directory and move on to the next file. In this solution, the file will stay there until the next run of a Windows command file that tries to send the failed file using Exchange. In the worst case, it would be there in 19 minutes 59 seconds. The best case would be one second. The company Supplier decided it could live with a 20 minutes wait. Here is a list of steps we need to take in order to automate this.

1. Install and configure iSoft as described in Chapter 2, "Implementing iSoft P2PAgent" on page 49.
2. Decide how you will exchange information with iSoft. Are you are going to run Expedite Base for Windows on the same machine that has iSoft on it? Then you can use directories to access the files. If this is another machine then will you be using Microsoft Networking to exchange the information? In that case, you can make the recycle directory a shared directory and connect to the shared directory on the Expedite Base for Windows machine. Will you be using WebSphere MQ to move the data around? If so, you will need to install WebSphere MQ on the machine and write a program to read it off the queue. For this example scenario, we will assume that iSoft and Expedite Base for Windows are running on the same machine.
3. Download the Expedite Manual and Software onto the machine that will be running it. See 5.3, "AT&T Global Network Dialer installation" on page 201 for more details. Also complete section 5.3, "AT&T Global Network Dialer installation" on page 201.
4. Review and include the changes described in 5.4, "Integrating iSoft and Expedite" on page 206.
5. At this point, we need to make some assumptions. The recycle directory for iSoft is going to be C:\Recycle. Expedite Base for Windows was installed in the default directory (C:\Expedite). A directory named C:\EDIDataNotSent has been created.
6. Edit C:\Expedite\Basein.msg. It should contain:

   ```
   SENDEDI FILEID(C:\EDIDataNotSent\edidatatobesent.txt);
   ```

7. Expedite Base for Windows will handle sending out the file only if it is one file and has the given name of a file. It cannot be C:\Recycle\*.file.in or C:\RECycle\*.*. iSoft will write out one file per transaction. If several files cannot be sent, you will have several files in the directory. This requires a Windows command file that will combine the files together and then send them out. An example of such a command file is shown in Example 5-3.

Example 5-3 Sample command file to prepare a single file for transmission via VAN

```
Dir C:\recycle\* | find "0 File" > nul
if errorlevel 0 if not errorlevel 1 GOTO END
Type C:\recycle\*.file.in > C:\EDIDataNotSent\edidatatobesent.txt
Del C:\recycle\*.file.in
For /f "tokens=2,3,4 delims=/- " %%x in ("%date%") do set d=%%x%%y%%z
For /f "tokens=1,2,3 delims=:. " %%x in ("%time%") do set t=%%x%%y%%z
If exist C:\EDIDataNotSent\edidatatobesent.txt CD C:\Expedite
If exist C:\EDIDataNotSent\edidatatobesent.txt C:\Expedite\IEBase.exe
If exist C:\EDIDataNotSent\edidatatobesent.txt find "SESSIONEND(00000)"
c:\Expedite\Baseout.msg > nul
If ErrorLevel 1 GOTO BACKUPDATA
GOTO CLEANUP

:BACKUPDATA
Rename C:\EDIDataNotSent\edidatatobesent.txt EDIDATANOTSENT%d%-%t%.txt
GOTO END
```

```
:CLEANUP
Del C:\EDIDataNotSent\edidatatobesent.txt
GOTO END

:END
```

8. The first two lines of this command file:

   ```
   Dir C:\recycle\* | find "0 File" > nul
   if errorlevel 0 if not errorlevel 1 GOTO END
   ```

   perform a check to see whether there are any files in the directory to be sent out. If the directory is empty then the command file ends without sending anything.

   The next two lines:

   ```
   Type C:\recycle\*.file.in > C:\EDIDataNotSent\edidatatobesent.txt
   Del C:\recycle\*.file.in
   ```

   are executed if there are files in the directory. With the **type** command, we can combine them into one large file (C:\EDIDataNotSent\edidatatobesent.txt). With the **DEL** command, we can remove the ones we have just combined; this way, iSoft can continue processing transactions. Next, we create temporary variables that will be used if there is an error from Expedite Base for Windows when the file is sent. Those lines are:

   ```
   For /f "tokens=2,3,4 delims=/- " %%x in ("%date%") do set d=%%x%%y%%z
   For /f "tokens=1,2,3 delims=:. " %%x in ("%time%") do set t=%%x%%y%%z
   ```

   Next, we perform the actual sending to Information Exchange. Those lines are:

   ```
   If exist C:\EDIDataNotSent\edidatatobesent.txt CD C:\Expedite
   If exist C:\EDIDataNotSent\edidatatobesent.txt C:\Expedite\IEBase.exe
   ```

   Then we review the return code sent back from Expedite Base for Windows. That line is:

   ```
   If exist C:\EDIDataNotSent\edidatatobesent.txt find "SESSIONEND(00000)"
   c:\Expedite\Baseout.msg > nul
   ```

   Note: This is one line. If the text inside C:\Expedite\Baseout.msg is anything other than SESSIONEND(00000) then there is an error and the file may not have been sent out. The rest of the command file is for error handling. It is:

   ```
   If ErrorLevel 1 GOTO BACKUPDATA
   GOTO CLEANUP

   :BACKUPDATA
   Rename C:\EDIDataNotSent\edidatatobesent.txt EDIDATANOTSENT%d%-%t%.txt
   GOTO END

   :CLEANUP
   Del C:\EDIDataNotSent\edidatatobesent.txt
   GOTO END
   ```

   If a problem occurred, then go to BACKUPDATA and create a file named EDIDATANOTSENT%DATE%-%TIME%.txt., where %DATE% is today's date in YYYYMMDD format. %TIME% is the system time in HHMMSS format. This file is placed in the C:\EDIDataNotSent directory. If the file was sent out correctly, then the temporary file is deleted and you can end the command file.

   In summary, the command file checks to see if there is anything to be sent. If there are files, it gathers them together into a temporary file, then deletes the original files. Expedite Base for Windows is called to send them out. Upon completion of Expedite Base for Windows, the script examines the return code to make sure Expedite Base for Windows was successful. If Expedite Base for Windows was not successful, it then creates a file in C:\EDIDataNotSent with a date and time stamp in the name so that it will not be overwritten. If Expedite Base for Windows was successful, it deletes the temporary file and ends.

5.5.2 Creating a Windows task

In order to run this command file every 20 minutes, we need to create an item in the Windows Task Manager.

1. Click **Start -> Programs -> Accessories -> System Tools -> Scheduled Task**.

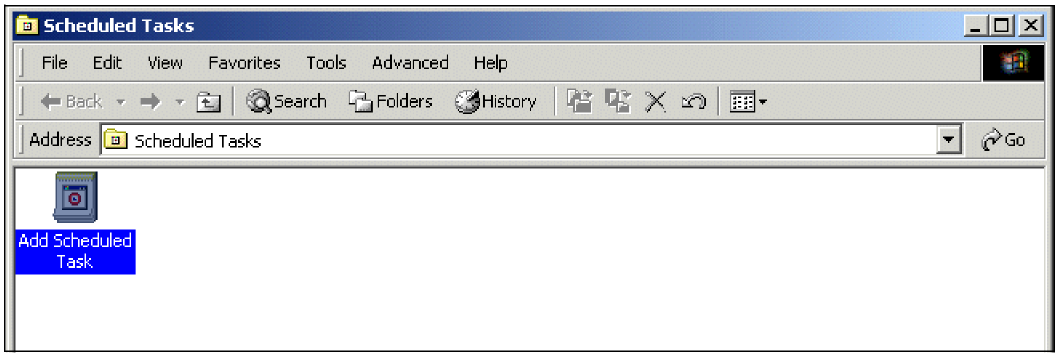
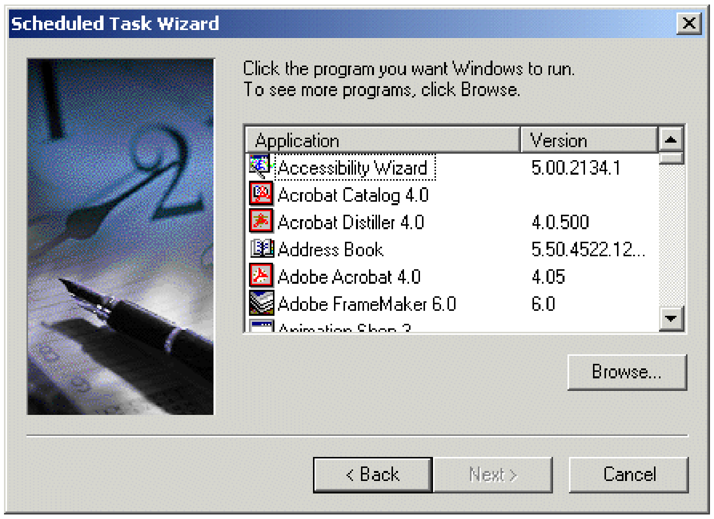

Scheduled Tasks
File Edit View Favorites Tools Advanced Help
Back Search Folders History
Address Scheduled Tasks Go
Add Scheduled Task

Figure 5-9 Windows scheduled tasks

2. Click the **Add Scheduled Task** icon as shown in Figure 5-9. A window will open up announcing that it is running a wizard to schedule a task. Click **Next**. In the next window (Figure 5-10), choose a program from the list presented by Windows or click **Browse** and find the command file.



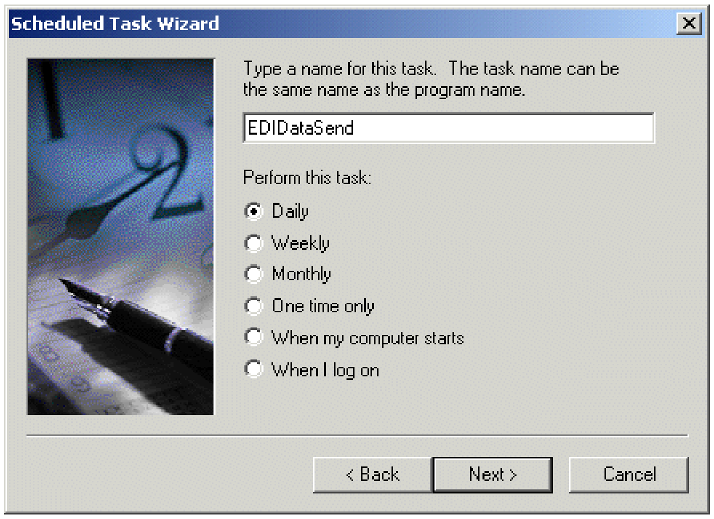
Scheduled Task Wizard

Click the program you want Windows to run.
To see more programs, click Browse.

Application	Version
Accessibility Wizard	5.00.2134.1
Acrobat Catalog 4.0	
Acrobat Distiller 4.0	4.0.500
Address Book	5.50.4522.12...
Adobe Acrobat 4.0	4.05
Adobe FrameMaker 6.0	6.0

Figure 5-10 Choose a task

3. After locating the command file (name EDIDATASEND.bat), click **Open**. Next, name the task for Windows Task Manager. You can accept the default or change it. Select **Daily** for the Perform this Task option. Click **Next**. Fill in your user ID and password in the appropriate fields in the next window.


Scheduled Task Wizard

Type a name for this task. The task name can be the same name as the program name.

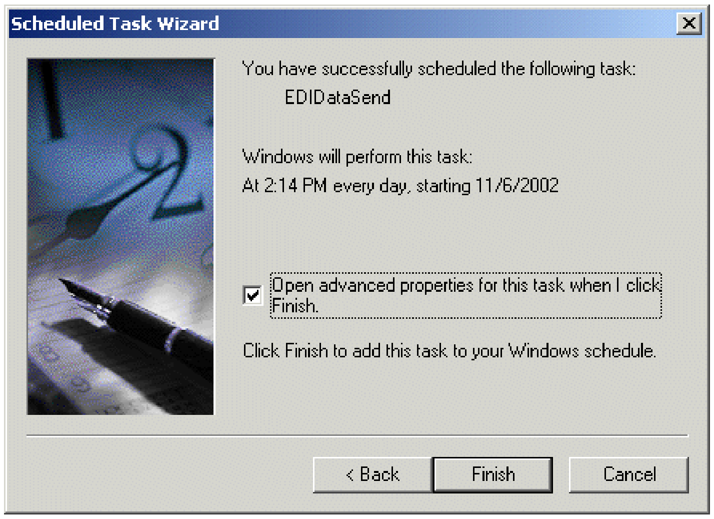
EDIDataSend

Perform this task:

(● Daily) ○ Weekly ○ Monthly ○ One time only ○ When my computer starts ○ When I log on

Figure 5-11 Choose a schedule

4. In the final window, shown in Figure 5-12, make sure to select the check box **Open advanced properties for this task when I click Finish**. Then click **Finish.**


Scheduled Task Wizard

You have successfully scheduled the following task:

EDIDataSend

Windows will perform this task:

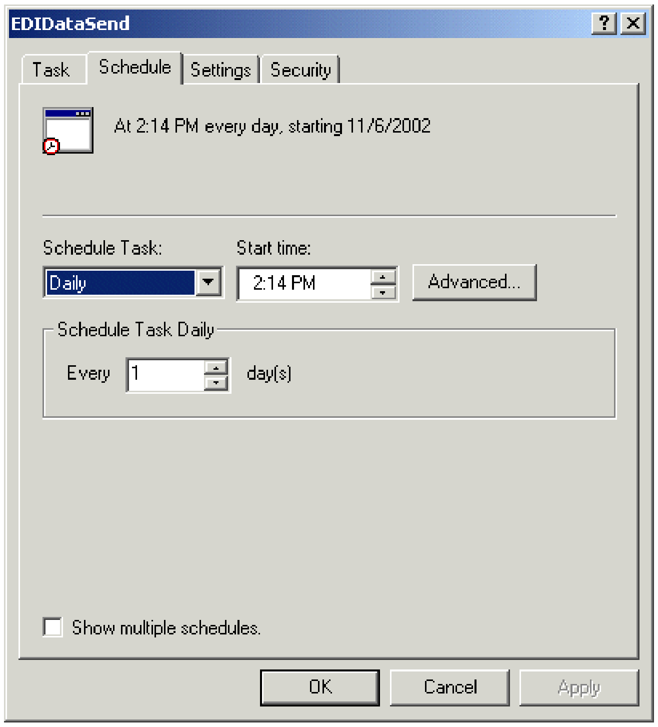
At 2:14 PM every day, starting 11/6/2002

[✓] Open advanced properties for this task when I click Finish.

Click Finish to add this task to your Windows schedule.

Figure 5-12 Successful scheduling

5. In the Advanced Properties window, click the tab **Schedule** at the top. Then click **Advanced**.


EDIDataSend

At 2:14 PM every day, starting 11/6/2002

Schedule Task: Daily

Start time: 2:14 PM

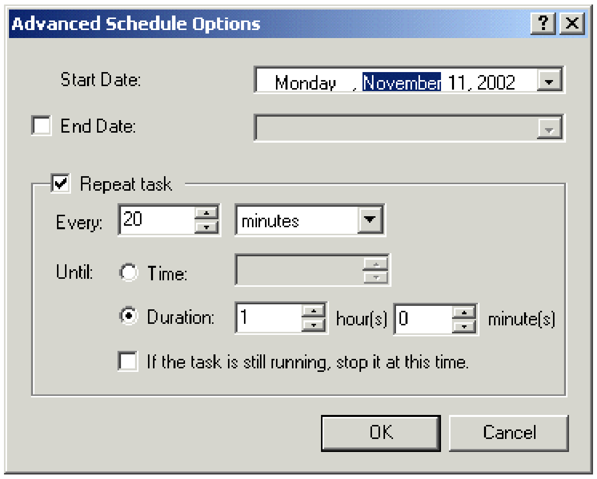
Schedule Task Daily

Every 1 day(s)

☐ Show multiple schedules.

Figure 5-13 Advanced properties for a scheduled task

6. In this window, shown in Figure 5-14, you can select **Repeat Task** and change the repeat interval to 20 minutes. Then click **OK**. Click **Apply**, then click **OK** again. The Task item should be closed.


Advanced Schedule Options

Start Date: Monday , November 11, 2002

☐ End Date:

☑ Repeat task

Every: 20 minutes

Until: ○ Time:

● Duration: 1 hour(s) 0 minute(s)

☐ If the task is still running, stop it at this time.

Figure 5-14 Set advanced schedule options

5.5.3 Receiving data from the retailer to the supplier

On the receiving side, our choices are limited. We can check the Information Exchange box periodically or use Expedite Notification Manager for Windows. Usage of Expedite Notification Manager for Windows involves the use of a modem with auto-answer turned on. For this sample scenario, we will check the mailbox periodically.

The company Supplier has decided that the mailbox should be checked twice a day at 7:45 a.m. and 8:15 p.m. The reason why 7:45 a.m. and 8:15 p.m. where chosen is the cost of receiving data in prime time. It is less expensive to receive data in non-prime time. Prime time is from 8:00 a.m. to 8:00 p.m. each weekday. By receiving just before and then again just after prime time, the company can save money if there are transactions in the mailbox. If there are no transactions, there is still a TCP/IP dial connect time charge.

In order to use Expedite Base for Windows for a receive process, a separate directory is needed. For the example, C:\RECVExpedite is created. Two files from C:\Expedite have to be copied over to this directory. It is assumed that the send process is working. They are:

- BASEIN.PRO
- BASEIN.MSG

Edit C:\RECVExpedite\BASEIN.PRO to include the following:

```
SESSION IEPATH(C:\EXPEDITE);
```

This tells Expedite Base for Windows where to find the files it needs. Edit C:\RECVExpedite\BASEIN.MSG. Tell it where to place the file it may receive. An example is:

```
RECEIVEEDI FILEID(C:\InboundEDIData\Inbound_EDI_Data.txt);
```

This command receive any EDI data from the mailbox and places it into the file Inbound_EDI_Data.txt in the directory InboundEDIData. The directory must exist before running Expedite Base for Windows. Expedite Base for Windows will overwrite this file by default. So, if data came in at 7:45 a.m. and nothing is done with it, then the 8:15 p.m. run will overwrite the file. If there is nothing to receive, the file will be empty after the 8:15 p.m. run. You may wish to change this by adding:

```
SESSION IEPATH(C:\EXPEDITE) OVERWRITE(N);
```

This will cause Expedite Base for Windows to append to the existing file instead of overwriting it. We recommend this option be added.

We will need to automate the process, then set up a task in Windows to perform it. The Windows command file could look something like Example 5-4.

Example 5-4 The command file RecvEDIData.bat

```
CD C:\RECVExpedite
C:\Expedite\IEBase.exe
```

In this command file, we are changing the directory to C:\RECVExpedite. This is a requirement so Expedite Base for Windows will use the correct BASEIN.PRO and BASEIN.MSG files. The last line calls Expedite Base for Windows. There is no error checking. There is no other processing.

To schedule these tasks, two new tasks must be created in the Task Manager. See 5.5.2, “Creating a Windows task” on page 209 for more details on how to create a task. The first one should be run at 7:45 a.m. and be named RecvEDIDataAM. The second task will be the same as the first but will run at 8:15 p.m. and be named RecvEDIDataPM.

5.5.4 Sending and receiving data at the same time

Expedite Base for Windows can connect to Information Exchange and do a send then a receive without disconnecting. This can save you in TCP/IP connect charges but you can also miss data if this is the only method you use. For example, company Supplier could not reach its customer over the Internet, so it is using the VAN to send the data. After sending the data, a receive is done and no data is found. Five minutes later, a retailer tries to send some data to the company Supplier. It, too, finds that the Internet is down and send the transactions over the VAN. Two hours later, when the Supplier tries to send more data, it finds the Internet to be up again and sends the data that way. Meanwhile, the VAN transactions sent by the customer are in the mailbox until the next time the Internet fails.

To add this dual capability, edit the BASEIN.MSG file. Add the receive line from 5.5.3, “Receiving data from the retailer to the supplier” on page 211. The file should look like the following:

```
SENDEDI FILEID(C:\EDIDataNotSent\edidatatobesent.txt);
RECEIVEEDI FILEID(C:\RECVExpedite\Inbound_EDI_Data.txt);
```

5.5.5 Problem determination

Most problems can be classified into one of three areas.

1. Expedite Base for Windows: use the manual you downloaded. Use the Search feature found in Adobe Acrobat Reader to find error codes and suggested fixes. If this does not help, call the Expedite help desk at (877) ECOM-IBM then select **Option 2** for Expedite Base for Windows support.
2. AT&T Global Network Dialer: the best way to troubleshoot this is to run it manually. Start up the product by clicking **Start -> Programs -> ATT Global Network -> AT&T Global Network Dialer**. Make sure it can connect. After it is connected, make sure you can download updates by clicking **Services -> Check for Updated Software.** If you are unable to log in, you may need to call the AT&T help desk at (800) 727-2222 then select **Option 4** for AT&T Global Network Dialer support. If you can log in, you should see a box as shown in Figure 5-15.

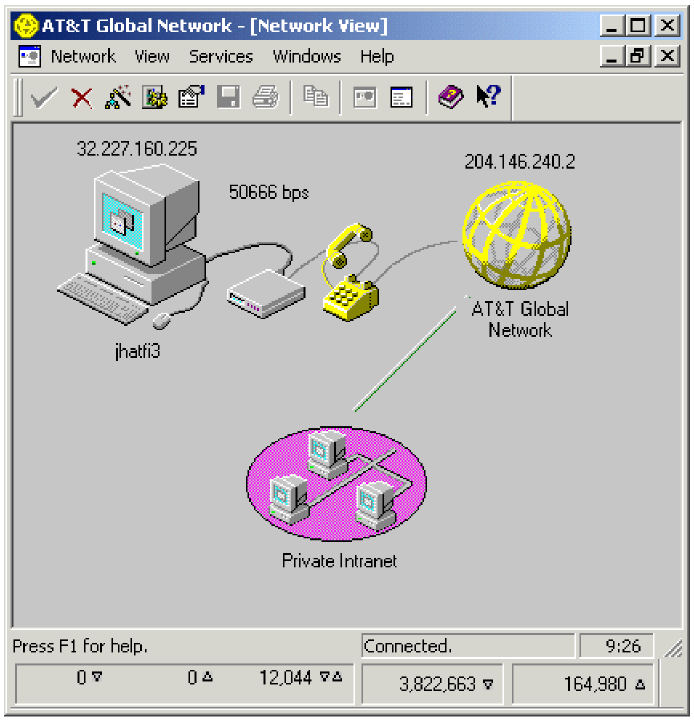


Figure 5-15 AT&T Global Network after successful dial-up

3. Windows command file: have someone else review your typing. Look at the logic. Do it in pieces instead of all at once.

Some problems are not easy to solve. If Expedite Base for Windows starts but does not run automatically, you need to edit the WIN.INI file. If you are getting a 19011 error from Expedite Base for Windows, you will need to call the Expedite Base for Windows help desk because your Information Exchange account was not created with TCP/IP access. If the AT&T Global Network Dialer is started but does not automatically connect, make sure you have saved your password. Do this by typing in your password then clicking the **Save Password** box.

If you need to see what is going on with Information Exchange (sent and received), consult this Web site:

```
http://ieas.services.ibm.com/servlet/RBNDNavigateRequestServlet?signon=Y&iesys=USA&
language=US&pagekey=index.html
```

By signing on with your Information Exchange account and password, you can see what is being sent and received from any Web browser. Note that only data processed during the last 30 days is kept.

5.5.6 Things to watch out for

Often, changes are made concerning the AT&T Global Network Dialer. New phone numbers are added, phone numbers are dropped, area codes change; those are just a few of the reasons why there can be changes with the dialer. By default, the AT&T Dialer will look to see if there are any updates and download them. This can cause you problems since you are just signing on and sending data. In order to change this default behavior, start the AT&T Global Network Dialer and sign on to the network. Once the dialer is connected, it looks something like Figure 5-15 on page 213; select **Network -> Connection Properties**.

Go to the Updates tab and deselect the option **Automatically check for updates after connecting**. This will stop the dialer from checking for phone number updates. Click **OK**.

If you would like to receive updates for the AT&T Global Network Dialer, you still can do this manually. Select **Service -> Check for Updated Software** while connected.

Whenever you perform the updates, make sure you restart the dialer manually after the update. The AT&T Global Network Dialer may force you to choose a new phone number based upon the update, your area code and your exchange. It will not run until this selection is made, which could impact automatic start-ups of the dialer and, as a consequence, Expedite.

A

Hardware and software configuration

This appendix provides details about the hardware and software configuration of the computers that were used to build the B2B solution.

Hardware configuration

The configuration consisted of three machines connected to each other via a 100Mb Ethernet connection. Each machine had the same hardware configuration.

Machine details

- IBM NetVista™ PC, Model 6792-MHU
- Pentium® 4 processor running at 1800 MHz
- 1 GB memory
- 40 GB hard disk

Software configuration

All machines had the same software configuration:

- Operating system:

  Windows 2000 Server with ServicePac® 3 installed.

- Software:

  – DB2 Universal Database™ Enterprise Edition V7.2 + FixPak 6
  – WebSphere MQ for Windows V5.3.0.1
  – InterChange Server V4.1.1
  – WebSphere Data Interchange for Multiplatforms V3.2 with CSD1
  – iSoft Peer-to-Peer Agent V3.1.2
  – Trading Partner Interchange V4.1.2.6

  Some machines had the Enterprise edition while others had the Advanced edition. Since the difference between those two editions is only a license issue, it only has an impact on how many connections you are allowed to create and it has, as such, no impact on functionality or performance.

Additional fixes for WebSphere MQ can be found at:

`ftp://ftp.software.ibm.com/software/mqseries/fixes/`

but were not required for our sample configuration.

DB2 FixPaks can be downloaded from:

`ftp://ftp.software.ibm.com/ps/products/db2/fixes/`

WebSphere Data Interchange fixes can be found at:

`http://www-3.ibm.com/software/integration/appconn/wdi/downloads/csdv32.html`

B

Additional material

This redbook refers to additional material that can be downloaded from the Internet as described below.

Locating the Web material

The Web material associated with this redbook is available in softcopy on the Internet from the IBM Redbooks Web server. Point your Web browser to:

```
ftp://www.redbooks.ibm.com/redbooks/SG246906
```

Alternatively, you can go to the IBM Redbooks Web site at:

```
ibm.com/redbooks
```

Select the **Additional materials** and open the directory that corresponds with the redbook form number, SG246906.

Using the Web material

The additional Web material that accompanies this redbook includes the following files:

File name	*Description*
SG246906.zip	Sample files for iSoft, WebSphere Data Interchange and InterChange Server in separate directories

Directory phase1

This directory contains two subdirectories named Supplier1 and Retailer1. Each directory contains a sample configuration for that specific partner and work order files to add and export keys. These files were used when the P2PAgent program was used for communication between two partners only.

Directory phase2
This directory contains three subdirectories, with files for the partners Retailer2, Retailer3 and for SUP2, which is the second identity of the Supplier. For each partner, there are again work order files and a P2PAgent configuration file.

Directory expedite
Here you will find two sample command files that were used in Chapter 5, “Implementing a back-up solution using IBM Expedite” on page 199.

Directory wdi
This directory contains two DTD files for use with WebSphere Data Interchange and CrossWorlds. A sample XML document and an EDI 850 document are available too, for use as input to the translation maps.

Directory ICS
This directory contains the import file for the collaboration template CollaborationFoundation, which was used in Chapter 2, “Implementing iSoft P2PAgent” on page 49.

System requirements for downloading the Web material

The following system configuration is recommended:

Hard disk space:	1 MB
Operating System:	Windows 2000

To recreate an environment as described in the redbook, you will need to have a machine similar to the one described in Appendix A, “Hardware and software configuration” on page 215.

How to use the Web material

Create a subdirectory (folder) on your workstation, and unzip the contents of the Web material zip file into this folder.

Abbreviations and acronyms

ANSI	American National Standards Institute
API	Application Programming Interface
ASC	Accredited Standards Committee
BO	Business Object
BPM	Business Process Management
CCITT	Comite Consultatif International Telegraphique et Telephonique
DER	Distinguished Encoding Rules
DES	Data Encryption Standard
DNS	Domain Name Server
DTD	Document Type Definition
EAI	Enterprise Application Integration
EDI	Electronic Document Interchange
EDIFACT	Electronic Data Interchange For Administration, Commerce & Transport
EDI-INT	EDI over the Internet
EJB	Enterprise JavaBean
ERP	Enterprise Resource Planning
FTP	File Transfer Protocol
GLN	Global Location Number
GTIN	Global Trade Item Number
GUI	Graphical User Interface
HIPAA	Healthcare Information Portability and Accountability Act
HTML	Hypertext Markup Language
HTTP	Hypertext Transfer Protocol
HTTPS	Extension of HTTP running under the Secure Socket Layer (SSL)
IBM	International Business Machines Corporation
ICS	InterChange Server
IDoc	Intermediate Document
IETF	Internet Engineering Task Force
IP	Internet Protocol
ISO	International Standards Organization
ITSO	International Technical Support Organization
JAR	Java Archive
JMS	Java Messaging Service
MDN	Message Disposition Notification
MIME	Multipurpose Internet Mail Extensions (RFC 1521)
ODA	Object Discovery Agent
ODETTE	Organization for Data Exchange through Teletransmission in Europe
PKI	Public Key Infrastructure
RFH	Rules and Formats Header
RSA	Rivest-Shamir-Adleman algorithm
SMTP	Simple Mail Transfer Protocol
SQL	Structured Query Language
TCP	Transmission Control Protocol
TPI	Trading Partner Interchange
UCC	Uniform Code Council
UML	Unified Modeling Language
URI	Universal Resource Identifier
URL	Universal Resource Locator
VAN	Value-added network
VICS	Voluntary Inter-industry Communications Standards

Related publications

The publications listed in this section are considered particularly suitable for a more detailed discussion of the topics covered in this redbook.

IBM Redbooks

For information on ordering these publications, see “How to get IBM Redbooks” on page 221. Note that some of the documents referenced here may be available in softcopy only.

- *WebSphere Data Interchange Installation and Configuration,* REDP-3600
- *Implementation of iSoft and integration with an EAI solution*, REDP-3625
- *Interoperability of iSoft P2PAgent and TPI*, REDP3650
- *An EAI Solution using WebSphere Business Integration (V4.1)*, SG24-6849
- *A B2B Solution using WebSphere Business Integration V4.1 and WebSphere Business Connection V1.1*, SG24-6916
- *B2B Solutions using WebSphere Business Connection*, SG24-6197

Online resources

These Web sites are also relevant as further information sources:

- The Drummond Group

  `http://www.drummondgroup.com`
- WebSphere Data Interchange download site for EDI standards

  `http://www-3.ibm.com/software/integration/appconn/wdi/downloads`
- DB2 fixes

  `ftp://ftp.software.ibm.com/ps/products/db2/fixes`
- WebSphere MQ fixes

  `ftp://ftp.software.ibm.com/software/mqseries/fixes`

How to get IBM Redbooks

You can search for, view, or download Redbooks, Redpapers, Hints and Tips, draft publications and Additional materials, as well as order hardcopy Redbooks or CD-ROMs, at this Web site:

ibm.com/redbooks

Help from IBM

- IBM Support and downloads

  ibm.com/support

- IBM Global Services

  ibm.com/services

Index

IBM Redbooks

Implementing EDI Solutions

(0.2”spine)
0.17”<->0.473”
90<->249 pages

Implementing EDI Solutions

Introduction to EDI technologies and products

Multi-partner and multi-product implementation scenarios

Integration options with WebSphere Data Interchange and InterChange Server

This IBM Redbook introduces the reader to the world of EDI. In addition to general terms about EDI, it also introduces a number of products in this area. WebSphere Data Interchange is discussed as the translation engine to map EDI documents to and from documents in other formats. This redbook also introduces two communication products that use Internet technologies: iSoft's P2PAgent and Trading Partner Interchange.

In addition to product introductions, the redbook describes several implementation scenarios in a multi-partner and multi-product environment. Besides a network where trading partners only use iSoft's P2PAgent, we also look at a setup where trading partners use a combination of the two products.

For each communication product, we investigate several integration options with internal applications and other middleware. We discuss the integration options with the translation product WebSphere Data Interchange and with the process integration product WebSphere BI Interchange Server. The integration technique can be file-based or messaging-based.

Finally, we take a look at options to combine the flexibility of the Internet with the reliability of value-added networks. When Internet connectivity is temporarily not available, a trading partner has the ability to use Expedite to dial into IBM's network and send or receive EDI documents. By exploiting the recycle mechanics in iSoft's P2PAgent, we can implement a solution that provides a highly available connection between trading partners.

SG24-6906-00 ISBN 0738453366