WebSphere Data Interchange for Multiplatforms

IBM

User's Guide

> **Note!**
>
> Before using this information and the product it supports, be sure to read the general information under Appendix E, "Notices" on page 385.

First Edition (October 2002)

This edition applies to the following products:
- IBM WebSphere Data Interchange for Multiplatforms Version 3.2
- IBM WebSphere Data Interchange for z.OS Version 3.2

and to any subsequent releases and modifications until otherwise indicated in new editions.

Contents

Figures

Tables

About this book

This book describes the setup and use of the WebSphere® Data Interchange Client V3.2 interface in creating and maintaining WebSphere Data Interchange profiles, data formats, maps, and EDI standards.

Who should read this book

This book is intended for three types of readers:
- WebSphere Data Interchange system administrators
- Translation analysts
- Business managers responsible for trading relationships

How this book is organized

This book has the following five parts:
- Part 1. Introduction

  This part contains information on installing and configuring WebSphere Data Interchange Client. It also includes information on how to use the WebSphere Data Interchange Client interface and its export and import functions. This part is intended for all readers.

- Part 2. Setup

  This part contains information on how to set up all your operational profiles. Most of the operational profiles are used for fine-tuning operations. The only required operational profiles are the mailbox profile and the network profile. This part is intended for WebSphere Data Interchange administrators who are responsible for system maintenance.

- Part 3. Trading partners

  This part contains information on setting up and maintaining trading partner profiles, which contain key business and technical information on the company with which you do business. This part is intended to help business administrators and others who are responsible for maintaining trading partner relationships.

- Part 4. Mapping

  This part covers the following items required for mapping; data formats, Extensible Markup Language (XML), and EDI Standards. You will also find information about creating maps, and specific instructions for creating data transformation maps, send maps, and receive maps.

- Part 5. Administration

  This part contains information on queries, reports, and the transaction store database, which can contain images of all your WebSphere Data Interchange translation activity. Queries control how WebSphere Data Interchange Client displays information in its windows and allow you to create reports and extract data from the transaction store. This part is intended for all readers.

About this book

Part 1. Introduction

Chapter 1. Introducing WebSphere Data Interchange

The accurate and timely flow of information in today's business environment presents a monumental challenge. As business processes evolve, your company's ongoing success depends on the flow of information. To improve these processes, you can exchange information electronically from one computer system to another. WebSphere Data Interchange, an e-*business solution,* is the bridge that can make it possible.

Why e-business?

Becoming an e-business can streamline your enterprise and give you a competitive advantage in the marketplace by providing:

- Improved ability to target new opportunities
- Automated closed-loop processes with end-to-end application integration
- Reduced costs through:
  - Increased productivity
  - Flexibility in exchanging information with your business partners
  - Just-in-time inventory management
- Greater control throughout the business cycle
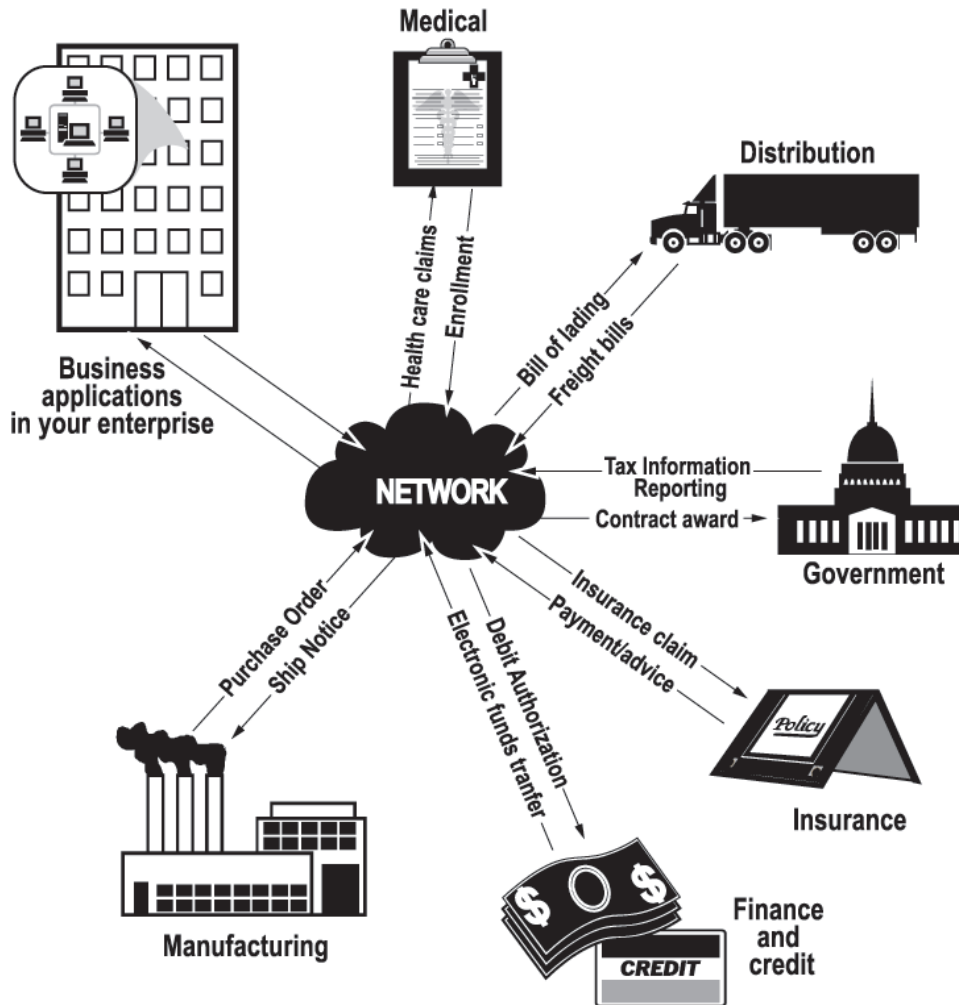
Figure 1. Extending your enterprise

WebSphere Data Interchange is an any-to-any solution

WebSphere Data Interchange is a key part of your e-business implementation to automate your business processes and extend your enterprise directly to your trading partners. WebSphere Data Interchange is an application that reformats data for electronic transmission, and includes the following features:

Flexible setup and administration
- Client interface to manage profiles, EDI standards, data formats, maps, and XML DTDs (Extensible Markup Language document type definitions)
- Export/import available to move user data between systems
- Client XML DTD import function
- Mapping designed to support:
  - Any-to-any mapping
  - XML DTD import
  - Literals/constants
  - Accumulators, arithmetic and logical operations

- Qualified loop and element mapping
- Hierarchical loop mapping
- Envelope field mapping
- User exits at the field level
- User-defined translation and validation tables
- Boolean logic

- Maps can be used by one or more trading partners

Superior translation capability

- Syntax checking
- Test and production support
- Carbon copy support
- Ability to translate and envelope separately
- Flexible command language interface
- Interactive, batch, event-driven, and real-time processing
- Automatic generation of functional acknowledgments

Versatile communications

- Support for managed networks and direct connections to trading partners
- Ability to resend individual transactions or entire envelopes
- Support for WebSphere MQ Queues as a means of exchanging data between trading partners

Extensive reporting and auditing

- Reporting of trading partner relationships, including what transaction sets are being used, and when the last communication was with a trading partner, and others
- Reporting of envelope and transaction status for both online and batch processing
- Reporting of exception information
- Setting acceptable error levels for the trading partner/map combination
- Reporting of SAP status for online and batch processing
- Optional audit log with archive recovery capability

EDI standards support

- Multiple EDI standards, including EDIFACT, X12, UCS, VICS, and RAIL
- Multiple versions and releases of EDI standards (Downloads are available from the WebSphere Data Interchange Web site http://www.ibm.com/websphere/datainterchange.)
- EDI standards distribution to speed delivery of new EDI standards
- Ability to migrate a map from one version/release of an EDI standard to another, or from one transaction to another
- Client creation and customization of EDI standards
- Optional audit log with archive recovery capability

XML support

- Direct DTD import to client
- Optional validation levels
- XML dictionary support

Additional features

- Designed for high throughput and performance
- Support for concurrent users and applications
- Support for a shared Trading Partners profile (Minimal Trading Partners)
- Ability to process in multiple environments
- DB2® implementation
- Multiplatform implementation
- Support for encryption and authentication
- Application program interface (API) to integrate directly with your application
- Utilizes the Security Access Facility (SAF) to establish system security down to the record level
- Support for WebSphere MQ Queues

What's new in this release

The major enhancements in WebSphere Data Interchange Version 3.2 are:

- Support for optimized ″any-to-any″ transformation of EDI, XML, and record oriented data formats.

  The any-to-any mapping function lets you map directly from any supported source document format (EDI, XML, or data format) to any supported target format (EDI, XML, or data format). This function, implemented through an improved graphical representation of XML data in the client, lets you create many new type mapping combinations such as EDI to EDI, XML to EDI, and EDI to XML.

- Advanced conditional and special mapping functions that you can exploit in mapping scenarios to address the most complex needs of standards translation.

  You can use new mapping commands in data transformation maps to hierarchically call translation maps, conditionally switch between translation maps, sequentially chain translation maps, and support Hierarchical Looping over data content structures. This allows you to handle translation of the complex dynamic hierarchical X12 formats used for the exchange of HL7 documents in X12 transactions, such as Advance Ship Notice and Medical Claims.

  BIN and BDS segments are now supported to enable the transmission of technical specifications and for embedding messages within messages.

  You can also create target-based maps, to help in the creation of some maps and to allow support for more complex structures, for example HLs in target documents.

- Advanced data validation and standards compliance function to provide industry leading support for both HIPAA, X12 embedded HL7, and other industry formats.

  You can now create a validation map and specify this on the mapping rule to request validation prior to base validation. You can create the validation map to include the ability to pass back information in the acknowledgement message.

  You can also create functional acknowledgement maps, and add these to the mapping rule to be executed after validation is completed. This will transform the results of the validation into a functional acknowledgement. You can create your own functional acknowledgement maps, or update the samples supplied.

- Direct import of industry standard or user defined XML DTDs for mapping and translation.

  You can import a DTD from a file, and use it to create a map in the same way that you can import and map EDI standard transactions. Support for user defined, as well as industry standard, DTDs provides unlimited choices.

- WebSphere MQ message queuing support.

WebSphere Data Interchange can now be driven by MQ messages to allow additional integration with WebSphere MQ Integrator by allowing WebSphere MQ Integrator to control the operation of WebSphere Data Interchange and cooperate in transformation of EDI messages. This facility introduces support provided by a WebSphere MQ Queue profile definition to associate logical destination names with real WebSphere MQ queues.

You can also use WebSphere MQ to interoperate with a wide range of Enterprise applications, Business Process engines such as the IBM Crossworlds InterChange Server, Information Brokers such as WebSphere MQ Integrator, and ERP systems such as SAP R3. Further, integration with the Exchange for WebSphere MQ offering from IBM e-business hosting enables WebSphere MQ connectivity direct to trading partners.

- Communications through VANs and Internet B2B gateways.

You can easily configure communication with trading partners using either Value Added Networks (VANs) or Internet B2B gateways. The WebSphere Business Connection offerings that provide AS1 and AS2 support and the IBM e-business hosting Expedite VAN gateway are two examples of supported gateways from IBM.

Sending and receiving data

Before you can use WebSphere Data Interchange to translate data, or to send or receive transactions, messages, or files, you must define certain information. This information describes how your system sends and receives data, how data is formatted in your application files and mapped to a standard, to whom you send data and from whom you receive data, and other pertinent information.

WebSphere Data Interchange also features *any-to-any* translation. Using any-to-any translation, you can translate data from any supported source document type to any supported target document type. Supported document types include data formats, EDI standards, and XML data.

Any-to-any translation uses a new type of *map* called a *data transformation map*. These maps use a different command syntax from the send and receive maps. You use a different PERFORM command, PERFORM TRANSFORM, to translate data.

For detailed information on sending and receiving data, see Chapter 5, "Sending and receiving data" on page 55

WebSphere Data Interchange information available on the Internet

The WebSphere Data Interchange product Web site is at:

    http://www.ibm.com/websphere/datainterchange/

By following links from this Web site you can:

- Obtain latest information about the WebSphere Data Interchange products.
- Access the WebSphere Data Interchange books in PDF format.

WebSphere Data Interchange on the Internet

Chapter 2. Client installation and setup

WebSphere Data Interchange Client is a Microsoft® Windows-based interface for WebSphere Data Interchange servers. This interface allows you to use a PC to create and maintain WebSphere Data Interchange profiles, EDI standards, data formats, and maps. It also can be used to load XML DTDs into WebSphere Data Interchange.

Note that WebSphere Data Interchange Client does not translate data. All translation functions are performed by the WebSphere Data Interchange server. WebSphere Data Interchange Client provides the ease of use inherent in Windows® to make your setup, mapping, and administration faster and easier.

WebSphere Data Interchange servers are supported on the following operating systems:
- AIX® Version 4.3.3
- Windows 2000
- OS/390® Version 2.8 or later (5647–A01)
- z/OS™ Version 1 (ESA/390 mode) or later (5694–A01)

If you have a WebSphere Data Interchange server on AIX or Windows, see Chapter 3, "Server installation and setup" on page 29. For installation and configuration information for z/OS servers, see the WebSphere Data Interchange Version 3.2 Program Directory and the *WebSphere Data Interchange for z/OS Installation Guide*.

Hardware requirements

WebSphere Data Interchange Client is a 32-bit application designed to run on hardware that meets the following requirements:
- Intel Pentium® PC
- CD-ROM drive for installing the distributed material
- 128 MB RAM

Make sure you have enough space on the hard drive on which you are installing WebSphere Data Interchange Client for the application, EDI standards, and data files. Minimum recommended space is 40 MB.

Software requirements

WebSphere Data Interchange Client runs on the following operating systems:
- Microsoft Windows 2000

WebSphere Data Interchange Client is designed to support any database with Open Data Base Connectivity (ODBC) Version 2 drivers. The following are examples of database software products and database connectivity products that support ODBC. These were chosen for demonstrative purposes to describe various WebSphere Data Interchange Client configuration alternatives. This is not a complete list.
- Microsoft Access 97
- Microsoft SQL Server 6.x
- DB2 Server Version 7.2
- DB2 for z/OS
- Connectivity to DB2 for z/OS through DB2 Connect™ Version 7.2

Installing WebSphere Data Interchange Client

WebSphere Data Interchange Client has an Install Wizard that guides you through installation. As with any installation, you should begin by closing any applications that you have running.

1. Insert the WebSphere Data Interchange Client Installation CD ROM. The installation process should start automatically. If it does not:

    a. On the menu, click **Start —> Run**.
        The Run dialog box opens with the cursor in the Open field.

    b. In the Open field, type `x:\client\WDIClient32.exe` (*x* indicates your CD ROM drive), and click **OK**.

    The Welcome Screen displays as the Install Wizard prepares to install WebSphere Data Interchange Client. Click **Next**.

2. A second Welcome screen displays with messages warning you to exit all Windows programs if you have not done so already. This screen also displays the license agreement and copyright. Click **Next**.

3. Read the copyright notice that displays. Click **Yes** to indicate you have read the notice and agree to its terms. Click **Next**.

    If this is the first time you are installing WebSphere Data Interchange Client, the Choose Destination Location dialog box opens. Select your installation location by clicking **Browse** and choosing the appropriate drive and directory. The drive can be selected from the drop-down list at the bottom of the dialog box. The default destination directory is `C:\Program Files\IBM\WDI Client V3.2`.

    Note: If you have installed previous versions of WebSphere Data Interchange Client or DataInterchange Client, you must install this version in a different directory.

    Click **Next**.

4. The Setup Type dialog box displays. Choose the type of setup you want by clicking the appropriate option button from the following choices:

    *   **Typical**

        Select this option the first time you install WebSphere Data Interchange Client 3.2. This option installs all the common options and creates the software's databases.

        This is the default setup type.

        Attention: If you are reinstalling WebSphere Data Interchange Client and you select this option, you receive a warning that Install will overwrite your databases. Install will only overwrite the default 3.2 database files installed through a previous 3.2 install. This option will not overwrite databases installed on a user's database system, such as DB2.

    *   **Custom**

        Select this option when you want to choose the options to install. This option is recommended for advanced users.

        If you are reinstalling WebSphere Data Interchange Client, you should use the Custom setup to avoid overwriting your databases and drivers with the defaults.

    Click **Next**.

5. If you chose **Typical**, go to the next step.

If you chose **Custom**, the Select Components dialog box displays. You can choose to install one or more of the following:
- Program files
- Crystal Report files
- Database files

Select a component by clicking the check box next to it. Click **Next**.

6. The Installer dialog box displays a listing of the selected features and the total size required to install them. Click **Next**.

7. The Install Wizard begins copying program files. To stop this process at any point, click **Cancel**.

   The screen displays the successful installation message.Click **Next**.

8. The readme file is displayed. Click **Finish**.

Select **WDI Client** from the program folder specified during installation to start WebSphere Data Interchange Client.

Note: If you are upgrading WebSphere Data Interchange Client from a previous release, see "Migrating data between versions and releases" on page 24 for more information.

Setting up connections to distributed server databases

If you have installed the WebSphere Data Interchange Server on AIX or Windows 2000, complete the instructions shown here to establish connections to the server databases.

If you have installed the WebSphere Data Interchange Server on z/OS, follow the instructions in "Accessing the WebSphere Data Interchange database on z/OS" on page 13.

The following instructions assume that the RunTime database for WebSphere Data Interchange Server has been installed successfully and that you are authorized to access those databases. Do the following tasks in the order they are given.

1. Install DB2 Connect Personal Edition V7.2

   a. Insert the DB2 Connect Personal Edition CD that is included in the package.

      The setup program should autostart; however, if it does not start, invoke setup.exe from the DB2 Connect Personal Edition CD.

   b. On the Installation dialog box, click **Install** and follow the prompts.

      1) On the Select Products dialog box, select **DB2 Connect Personal Edition**. The other products are not needed.

      2) On the Select Installation Type dialog box, select **Typical**.

      3) On the Choose Destination Location dialog box, accept the default directory or choose an alternate installation destination.

      4) On the Configure NetBIOS dialog box, accept the default.

      5) On the Enter Username and Password for Control Center Server dialog box, type your username and password. If an error occurs due to authority, you can usually ignore it and continue the installation.

      6) On the Start Copying Files dialog box, click **Next**.

         At this time, DB2 Connect begins to install on your system.

         When the installation is complete, the Setup Complete dialog box opens.

      7) Click **Finish**.

Accessing server databases

> The Steps dialog box opens.

   c. Click **Exit**.

   d. Reboot your system.

2. Configure your database connections. Complete the following instructions for your RunTime database.

   a. Open the Client Configuration Assistant in the IBM® DB2 program folder.

   b. If the Welcome dialog box opens, click **Add Database** . If the Welcome dialog box does not open, on the Client Configuration Assistant dialog box, click **Add**.

> The Client Configuration Assistant dialog box opens.

   c. On the Add Database Wizard dialog box, select **Manually configure a connection** on the Source tab page.

     1) Click **Next**.

> The Protocol tab page opens.

     2) Select **TCP/IP**.

> The TCP/IP tab page opens.

     3) Specify the name of the system or the IP address of the system that contains the databases, and specify the port number of the database.

> This information must be obtained from your system administrator.

     4) On the Database tab, specify the name of the database.

> The name is obtained from your system administrator. The default value for the database is EDIEC32E.

     5) Click **Finish**.

> A Confirmation dialog box opens. Use this dialog to test your connection. If you are not able to connect successfully and can not resolve the issue, contact your system administrator.

3. Identify your system to WebSphere Data Interchange Client

   a. Start WebSphere Data Interchange Client.

   b. From the menu, click **View > EDI Systems**.

   c. Create a new EDI System that utilizes your databases.

     1) From the menu, click **File > New**, or click **Create New Document**.

> An EDI System dialog box opens.

       a) Specify a name for the new system.

       b) Select the Data Source Name for the database.

       c) Specify the database qualifier for the database.

> The qualifier is sometimes referred to as the schema. The default value for the database is EDIENU32. Confirm this value with your system administrator.

       d) Identify the platform that the WebSphere Data Interchange Server runs on.

       e) Select a color to be associated with this EDI System.

> Associating unique colors with each EDI System helps the user to easily identify which system they are working with in a multi-system environment.

       f) Click **OK**.

> The new system is saved.

   d. Exit and restart WebSphere Data Interchange Client.

e. Click **Setup**.

f. Enter the user ID and password for the RunTime database, if requested.

If the Setup Functional Area window opens, the connection to your databases is complete. If there is an error, ensure your EDI System is set up correctly. An incorrect database qualifier is a common problem.

Accessing the WebSphere Data Interchange database on z/OS

If you install the WebSphere Data Interchange server on z/OS, it requires a z/OS-based database at run-time; DB2 is supported. Various factors will determine, or limit, how you reflect changes from the client onto the host. It may be by choice, or you may be limited by your network or database installation. You can use two methods: client-server mode or stand-alone mode.

Client-server mode

In client-server mode, WebSphere Data Interchange Client accesses the WebSphere Data Interchange database on z/OS directly using an ODBC link. ODBC is an industry standard for making connections between a variety of software products and databases on different hardware platforms. Access to the host via ODBC is controlled by middleware, as illustrated in Figure 2.

See "Middleware" on page 14 for more information about options available. If you are using DB2 on z/OS, consider client-server mode which gives you real-time updating on the host from WebSphere Data Interchange Client.
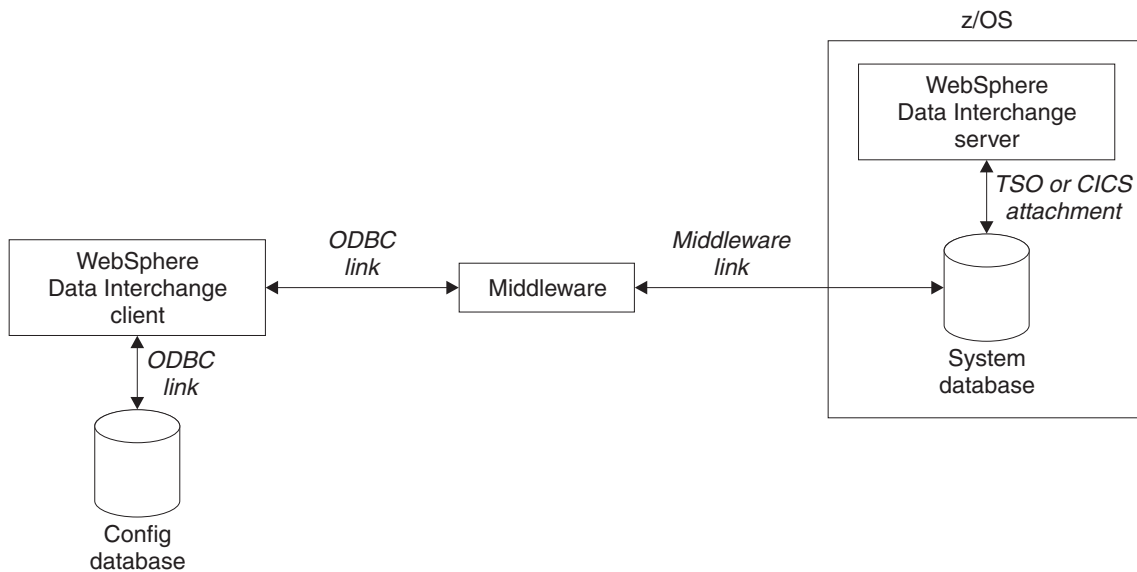
Figure 2. Client-server configurations

Stand-alone mode

In stand-alone mode, you make the connection with the host database through WebSphere Data Interchange Client's Export/Import functions, which are described in Chapter 6, "Export and Import" on page 67. Use this method if you cannot establish a client-server connection to the host.

When in stand-alone mode, you transfer a file to the host from the client by exporting the data into an export or import (EI) file. Then utilize file transfer

Stand-alone mode

software, which you must obtain separately, to upload the file to the host using the ASCII and CRLF options. On the host, you import the EI file into the WebSphere Data Interchange Host database using the host's export/import functions, as illustrated in Figure 3.
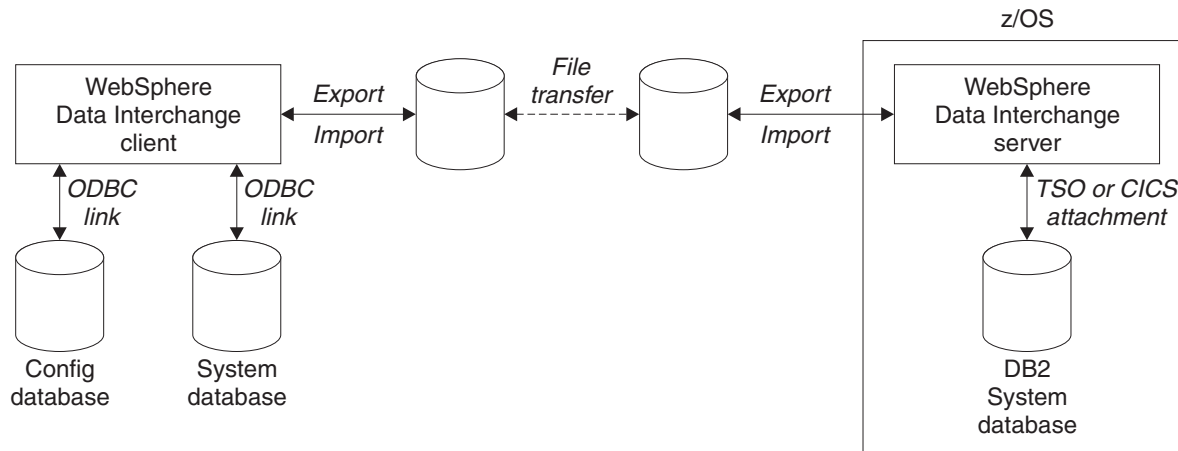
Figure 3. Stand-alone configuration

Middleware

The term middleware is used to describe many things. In this context, it enables access to DB2 on z/OS from a Windows PC. There are many solutions by various software manufacturers that support ODBC connectivity to DB2 on z/OS; either direct access from the client PC or through a gateway. There are numerous protocols and configuration options. Your middleware determines what paths are allowed between the PC and z/OS. Some products allow direct connection between the PC and the host. Other products work through Windows NT®, UNIX®, AIX, or OS/2® gateways that use the SNA or TCP/IP protocols to talk to z/OS, while TCP/IP, NetBEUI, or IPX is used to talk to the PCs.

As examples, the IBM Personal Edition PC product, DB2 Connect Version 7.2, connects a client workstation directly with DB2 for z/OS over SNA or TCP/IP networks, without additional host software. DB2 Connect V7.2 Enterprise Edition is an indirect (gateway) solution.

WebSphere Data Interchange Client databases

WebSphere Data Interchange Client is designed to support any database with ODBC Version 2 drivers. Many database packages developed by various vendors provide this level, or above, of ODBC support: each has its own database management system (DBMS). In WebSphere Data Interchange, a database is classified as either single-user or multi-user. A single-user DBMS is one that allows only one user access at a time. These usually reside on each user's PC. A multi-user DBMS, on the other hand, typically resides on a host or server, and provides multiple users shared access to a common database. DB2 for z/OS is an example of a host-based multi-user database. Microsoft's SQL Server, and DB2 for AIX are examples of LAN server databases with ODBC support.

WebSphere Data Interchange Client partitions its data into two databases. ODBC is used to access both:
- Config
- System

As distributed, WebSphere Data Interchange Client provides both databases as Microsoft Access 97 single-user databases, supporting stand-alone operation. Since ODBC is a standard, you are not limited to using the provided Microsoft Access 97 databases. And because each WebSphere Data Interchange Client database is independent, each can, theoretically, reside in a different database management system. "Configuration alternatives" on page 16 for variations and examples.

Config database

Config is a local configuration database that stores installation-wide data, such as the list of installed systems, queries, report definitions, and user preferences like the columns that appear in list windows. There can be only one config database defined to the PC running WebSphere Data Interchange Client. By default, it is named WDIClient32Cfg.

Note: Having access rights to the config database does not permit a user to connect to any particular system. Access to system databases is controlled separately. This reduces the need to control access to this PC database.

System database

The system database contains all the tables that hold customization time data and runtime data. The ODBC Data Source Name can be any name of your choice, and any number of systems can be defined. This allows you to have multiple system databases on the same PC, along with multiple system database connections to remote PCs, servers, or hosts. Since ODBC is used to access the system database, the database is not limited to being defined in DB2 on z/OS. Even a single-user database, such as the distributed Microsoft Access 97 database, can be used for the system database; however, multiple clients cannot concurrently share the same data in this scenario (which means that in this configuration they cannot readily share data developed on one another's PCs).

If you have multiple WebSphere Data Interchange Client users and require concurrent access, define the system database in a multi-user database management system that supports ODBC. DB2 on z/OS can be used for this, but it is not the only option. Any multi-user database supporting ODBC Version 2 or later can be used. Of course, if you choose not to use the distributed single-user Microsoft Access 97 database, you will need to define new tables. The data definition language (DDL) used to create databases, tables, and indexes is not the same for every DBMS. Sample DDL is distributed in subdirectory DDL when choosing Sample DDL Files during custom installation.

Note: You might have, or find, vendor software to aid in converting DDL from one DBMS to another.

Customization time data
Customization time data includes tables related to the customization of EDI standards, data formats, and maps.

Runtime data
Runtime data includes data that is required by the WebSphere Data Interchange server translator, including control strings, profiles, send and receive usages, map rules, translation tables, code lists, the transaction store, and event logs.

Configuration alternatives

You can configure your database using one of several alternatives, as explained below.

Single user, stand alone

Both databases, Config and System, reside on the client PC, as distributed, using Microsoft Access 97 databases.

- Easiest to set up, lowest cost.
- Updates to the host databases are not updated in real time. Must use export/import to exchange data with WebSphere Data Interchange server.
- Must use separate file transfer utility to upload and download export/import files between z/OS and the PC.
- Must use export/import to exchange data with other WebSphere Data Interchange Clients.

This is a good alternative if there is only one user of WebSphere Data Interchange Client, or for initial testing.

Setup example

1. Install WebSphere Data Interchange Client (Typical) on the PC of each user.
2. Establish export and import procedures. Always use tagged export format when exchanging objects between host and client.
3. Establish file transfer procedures.

   Note: Check your file transfer options. Use ASCII and CRLF options. Allow for long, variable-length records: a variable-record format (RECFM=V or VB), a record length of 8152 (LRECL=8152), and block size of 8156 (BLKSIZE=8156).

4. If multiple WebSphere Data Interchange Clients are involved, establish external controls:
   a. Establish a naming convention to avoid overlaying one another's work.
   b. Establish procedures for sharing trading partners. For translation purposes, a trading partner who is common between WebSphere Data Interchange Client users must be shared on the host so that you do not have two different names for the same trading partner on the host.

Multi-user, stand alone

Config resides on the WebSphere Data Interchange Client PC as distributed using Microsoft Access 97 single-user database. The system database is shared on a server other than the host.

- Relatively easy to set up.
- All data is shared with other WebSphere Data Interchange Client users.
- Updates to the host database is not real-time. Must use export/import to exchange data with WebSphere Data Interchange server on z/OS.
- Must use file transfer utility, obtained separately, to upload and download export/import files between z/OS and the PC.

Setup steps

The following describes the steps for using DB2 on Windows 2000. These steps might be different for other databases, servers, or both. TCP/IP connectivity between clients and Windows 2000 server is assumed.

1. Install WebSphere Data Interchange Client (Typical) on PC of each EDI user.
2. Establish Export/Import procedures. Always use tagged export format when exchanging objects between host and client.
3. Establish file transfer procedures.

   Note: Check your file transfer options. Use ASCII and CRLF options. Allow for long, variable-length records: a variable-record format (RECFM=V or VB), a record length of 8152 (LRECL=8152), and block size of 8156 (BLKSIZE=8156).

4. Install DB2 Connect Personal Edition on each client PC.
5. Install DB2 Universal Database™ on the server.
6. Create and load the system database on the server.
7. Create an ODBC data source for the system database on each client PC. Choose a name other than DIClient32Dev as this ODBC data source name is already taken by the distributed Microsoft Access 97 database.
8. Authorize users to access all system database tables on the server. Users will require SELECT, INSERT, UPDATE, and DELETE privileges on all tables defined on the server.

Multi-user, client-server, all DB2

Config resides on the WebSphere Data Interchange Client PC as distributed using Microsoft Access 97 single-user database. The system database is the shared server DB2 database.

- Data is shared with other WebSphere Data Interchange Client and server users in real-time.
- The WebSphere Data Interchange Client rename action is limited because of DB2 restrictions.

This is a good alternative for DB2 customers who have many people working together on projects.

Setup steps

This configuration describes the steps for using DB2 Connect V7.2.

1. Install WebSphere Data Interchange Client (Typical) on the PC of each user.
2. Install IBM DB2 Connect V7.2 on gateway server. Follow the same steps as for multi-user, client-server example above.
3. Install IBM DB2 Connect for z/OS on each client PC.
4. The ODBC data source name for the z/OS DB2 subsystem is registered for you by DB2 Connect's Client Configuration assistant.
5. Grant user privileges to the z/OS DB2 tables. The ability to change the system database from the client requires SELECT, INSERT, UPDATE, and DELETE privileges on a subset of WebSphere Data Interchange z/OS DB2 tables.

Configuring systems

WebSphere Data Interchange Client allows you to manage multiple systems with a single installation. You can use this if you have more than one WebSphere Data Interchange system, for example, if you are using a test system to test new maps you create, and your production system is an active system that handles business transactions. On the server, these systems might even be separate installations of WebSphere Data Interchange.

Configuring systems

Each system has its own database tables. For example, you can have a map named MAP1 in both test and production systems. Although you can only use one system at a time if you use the WebSphere Data Interchange interface on z/OS, you can work in more than one system at a time with WebSphere Data Interchange Client. For example, you can edit maps in test and production systems, simultaneously.

Each system database must be defined as a unique ODBC Data Source Name (ODBC DSN) in order for WebSphere Data Interchange Client to access the system. The particulars of creating ODBC DSNs depend on the database management system. The Windows ODBC Manager maps Data Source Names (DSNs) to ODBC drivers. ODBC Manager is found in Settings -> Control Panel.

WebSphere Data Interchange Client is shipped with one system called Development. The ODBC DSN associated with it is WDIClient32Dev. This DSN is added automatically during Typical WebSphere Data Interchange Client installation, and it uses Microsoft Access 97 and points to the supplied Microsoft Access 97 database file WDIClient32Dev.mdb. The Development system is ready to go for single-user, stand-alone mode after a Typical installation.

Note: WebSphere Data Interchange Client distributes one other database file: WDIClient32Cfg.mdb. This has a DSN of WDIClient32Cfg and is also automatically added during Typical installation. There is only one config DSN per WebSphere Data Interchange Client installation, not one per system. Config is necessary before accessing any system.

Whether using WebSphere Data Interchange Client in stand-alone or client-server mode, there are three basic steps for creating a new system:
1. Create the system database and load the default data.
2. Create the ODBC DSN for the database.
3. Define a new system in WebSphere Data Interchange Client using the new Data Source Names.

Defining additional single-user databases

As an example, suppose you want to add another single-user, stand-alone system called Test.

Setup steps
1. Copy the file `c:\Program Files\IBM\WDI Client V3.2\WDIClient32Dev.mdb` to `WDIClient32Dev2.mdb`.

   Note: You do not need to copy WDIClient32Cfg.mdb because it contains tables common to all systems.
2. Create the ODBC data definition for the new database. Do the following:
   a. From the Windows Control Panel select the ODBC icon. On some Windows systems this icon can be found within the Administrative Tools icon. The ODBC Data Sources window displays.
   b. Select the User DSN tab, and then click **Add**.
   c. Select the Microsoft Access 97 driver, and then click **OK**.
   d. To create a new system database entry, do the following:
   e. Enter a data source name, such as Test.
   f. Enter a description, such as Test database.
   g. Click **Select**.

    h. Select the database you copied to.

3. Set up the System definition in WebSphere Data Interchange Client:

    a. Select Systems from the View menu.

       The Systems List window displays with a list of current systems.

    b. Click **New**.

       The Systems dialog box displays.

    c. Enter a name in the System Name dialog box. The name can contain embedded spaces, and be up to thirty characters long. This is a required field.

    d. Select the appropriate server platform from the server platform field. This identifies the platform the host translator will run on. If the host translator will run on both z/OS and CICS®, select z/OS.

    e. Select the data source name Test from the drop-down list in the Data Source Name field.

       This field displays in the database ODBC Information group box.

    f. Leave the Database Qualifier blank.

       The Database Qualifier is used when connecting to multi-user or host databases. For example, on DB2 for Windows NT Server, this corresponds to the database schema. On DB2 for z/OS, this is the high-level-qualifier specified on the host DB2 tables and views.

    g. To change the color of the window background for this system, click **Change**. For details on changing system color, see "Selecting the system color" on page 50.

    h. When you are finished, click **OK**.

    i. You must restart WebSphere Data Interchange Client before you can use the new system.

       The name of the new system will display in the System drop-down list.

Defining a multi-user database

The particulars of defining multi-user or host databases and creating ODBC DSNs depend on your database management system and its operating system. After you have installed the appropriate ODBC drivers on the PCs that are running WebSphere Data Interchange Client, you must configure them using the Windows ODBC data source administrator. In Windows, for example, go to Control Panel and double-click the ODBC icon. Some database products provide GUI tools to set up databases or connections and they might have an option to register the ODBC data sources for you. If not, manually use the ODBC Data Source Administrator to add your new DSN. Select your database driver from the list of available ODBC drivers. Next, name your new DSN and provide any required parameters specific to your driver. Lastly, define your system definition in WebSphere Data Interchange Client.

Recommended configuration

The following configuration combines elements of a source code control system with a rigidly managed deployment process for changes in a multiple translation server environment. This configuration uses both PC and server databases. It makes extensive use of client-server access, along with the Export to System function within WebSphere Data Interchange Client to move objects, such as maps and DTDs, from system to system.

Multi-user, multi-server with both stand alone and client-server access

Each developer has a local database that resides on their PC. In addition, three server DB2 repositories are shared by developers, testers, and administrators.

- Complex setup, higher cost.
- Sharing between developers is easy because there is a single DB2 repository for all build time objects.
- Minimum response time is ensured for developers by keeping items currently in progress in a local database.
- Minimizes potential contamination of test environment by developers by separating the test database from the development database, and allowing only testers to have access to the test database.
- Minimizes potential contamination of the production environment by testers or developers by separating the production database from the test and development databases, and not allowing testers or developers access to the production database.
- Minimizes potential contamination of complex build time objects (XML, EDI standard, and Data Format document definitions, along with maps) by not deploying them to the test and production environment. Only runtime objects (such as trading partners, usages, rules, and control strings) are deployed, so testers and administrators cannot change build time objects.
- Check-out, check-in, and manual deployment (or promotion) of objects can be done directly by an authorized user with the WebSphere Data Interchange Client Export to System function.
- Deployment can be automated with the use of the PERFORM EXPORT and PERFORM IMPORT server commands.

This is a good alternative for a medium to large scale B2B (Business to Business) shop.

The following is an illustration of the recommended configuration.
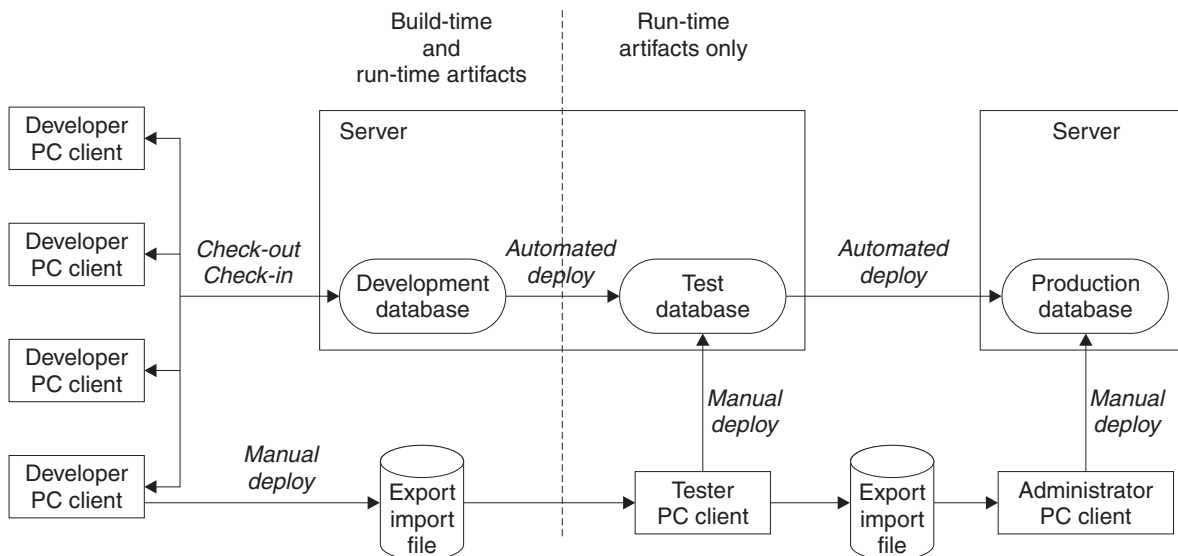
Figure 4. Recommended configuration

Setup example

Install WebSphere Data Interchange Client (Typical) on the PC of each user (developer, tester, or administrator).

- For developers, install the local database and set up their computers for ODBC access to the development database.
- For testers, do not install a local database. Set up their computers for ODBC access to the test database, and if manual deployment will be allowed via Export to System, then also set up access to the development database.
- For administrators, do not install a local database. Set up their computers for ODBC access to the production database and if manual deployment will be allowed via Export to System, then also set up access to the test database.

Establish check-in and check-out procedures for the developers. WebSphere Data Interchange does not provide a check-in, check-out mechanism, so this must be done manually.

Establish procedures for deploying (or promoting) objects. If automated deployment is to be used, write the deployment JCL jobs, or scripts.

Running WebSphere Data Interchange Client

WebSphere Data Interchange Client is designed to prevent more than one user from working on the same item at the same time. As a result, you need to log on to WebSphere Data Interchange Client databases every time you start the software, as follows.

1. Select WebSphere Data Interchange from the WebSphere Data Interchange item on the Start menu.

   Note: If you are using a multi-user or client-server setup, your User ID and Password are determined by your system administrator.

You are now ready to connect to the system that displays in the System drop-down list. To change which system will be accessed, select an alternate system name from the list.

Attention: Use the ODBC Administrator in the Windows Control panel to set up your passwords if you do not want to type your passwords every time you access a system.

You can work on items in different systems at the same time. For example, you can edit a map from the Development system while also editing one from the Test system.

Importing EDI standards

When you have configured your WebSphere Data Interchange Client system, you must import EDI standards before you can map with EDI standards. WebSphere Data Interchange Client is shipped with a number of common EDI standards on the CD. You can download the most recent EDI standards from the WebSphere Data Interchange Web site at:

`http://www.ibm.com/websphere/datainterchange;`

To import the EDI standards from the CD:

1. Insert the CD ROM containing the EDI standards file you want to install.

Importing EDI standards

2. Select Open Import File from the File menu of WebSphere Data Interchange Client.

3. Select the directory STDS on the CD ROM.

4. Choose one of the following subdirectories:
   - X12
   - VICS
   - RAIL
   - UCS
   - EDIFACT

   The available EDI standards display.

5. Select the EDI standard you want to import by clicking its plus sign or double-clicking its folder.

   The folder opens and each transaction within the EDI standard displays.

   Note: You do not have to import an entire EDI standard. If you prefer, you can import only the transactions you need.

6. Click the transactions you want to import, or click the EDI Standard to import all transactions.

   The Import button on the tool bar becomes available.

7. Click **Import**.

   An Execution Status dialog box displays showing your import progress.

Your EDI standards are now imported. Their components appear in the windows you see when you click EDI Standards on the Navigator bar. For details, see Chapter 24, "EDI standards" on page 185.

Moving to WebSphere Data Interchange Client

It is extremely important that you have proper external controls and procedures in place to prevent users from updating the same items using both the host and client interfaces, as that can result in loss of some of your updates.

If you are moving Host format maps, EDI standards, and data formats from a previous release, refer to the migration information in *WebSphere Data Interchange for z/OS Installation Guide*.

Moving setup profiles and trading partner profiles

Setup profiles and trading partner profiles can be accessed from WebSphere Data Interchange Client. Setup profiles include mailboxes, activity logs, network profiles, and so on.

Client-server mode

If you have client-server access to the WebSphere Data Interchange Host databases, you can create new profiles or update existing ones using the WebSphere Data Interchange Client windows interface. Changes that you make using WebSphere Data Interchange Client are immediately reflected in the host databases when you save your changes.

While you are making changes using WebSphere Data Interchange Client, other client users are locked out from making changes to the same profile. Likewise, if you are making changes using WebSphere Data Interchange Host, other WebSphere Data Interchange Host users cannot make changes to the same profile. However, it is possible for a client user and a host user to make changes to the

same profile at the same time. As a result a preferred interface should be chosen to avoid this type of problem. Ideally, when using Client-server mode, objects should be updated only in the Client except in cases of emergency. Emergency change management procedures should be documented

Stand-alone mode
If you want to make changes to Setup and trading partner profiles in stand-alone mode, you must first export, download, and import them to WebSphere Data Interchange Client from WebSphere Data Interchange Host. Make sure that another user is not updating the item on the Host at the same time you are updating it on the Client. If that happens, the Client user overwrites any changes made by the Host user. As a result, it is extremely important to have external controls in place to prevent this from happening.

To update a profile that currently exists on the host:

1. Export the profile from WebSphere Data Interchange Host in tagged format.
2. Using a file transfer utility, download the profile to the PC where WebSphere Data Interchange Client is running.
3. Import the profile into WebSphere Data Interchange Client
4. Update the profile using WebSphere Data Interchange Client.
5. Export the profile from WebSphere Data Interchange Client in tagged format.
6. Using a file transfer utility, upload the file to the WebSphere Data Interchange Host.
7. Import the profile into the WebSphere Data Interchange Host.

   You do not have to import all the profile fields into the WebSphere Data Interchange Host.

Control strings

The WebSphere Data Interchange Host translator does not use EDI standards, data formats, or maps directly during translation. Instead, it uses a control string, which includes information about the EDI standard, data formats, and map.

On WebSphere Data Interchange Client, EDI standards, data formats, and maps are compiled into a control string. A control string contains what is needed of these objects to perform EDI translation on the host.

Following are considerations for compiling maps into control strings in client-server and stand-alone modes:

Client-server mode
When you compile a map in client-server mode, the resulting control string is placed directly in the host database.

Stand-alone mode
In stand-alone mode, the compile option creates a control string in the PC database. You must then export the control string in tagged format, upload the file, and import the data into WebSphere Data Interchange Host.

Note: Check your file transfer options. Use ASCII and CRLF options. Allow for long, variable-length records: a variable-record format (RECFM=V or VB), a record length of 8152 (LRECL=8152), and block size of 8156 (BLKSIZE=8156).

Migrating data between versions and releases

Use the WebSphere Data Interchange Client Release Migration function to migrate data from an earlier version of DataInterchange or WebSphere Data Interchange Client to WebSphere Data Interchange Client 3.2.

If you are currently using an earlier version of DataInterchange or WebSphere Data Interchange Client, you must perform the release migration to move the data to WebSphere Data Interchange Client 3.2.

Note: Release Migration will not move DataInterchange 3.1 Host format maps, EDI standards, and data formats. You must import these in Host 3.2. Refer to the migration information in *WebSphere Data Interchange for z/OS Installation Guide*.

When planning your release migration, consider the following:

- A database administrator should perform the release migration for shared databases.
- An individual user can perform the release migration for a stand-alone database configuration.
- Migrate runtime data that is contained on the host as part of the WebSphere Data Interchange 3.2 Host installation process. Runtime data not contained on the host can be migrated using the WebSphere Data Interchange Client release migration procedure below.
- Migrate customization time data using only WebSphere Data Interchange Client.
- Migrate configuration data only if you want to save custom queries, reports, or system data. If configuration data is migrated, the system data moves from the earlier version of DataInterchange to WebSphere Data Interchange Client 3.2. The first time you use WebSphere Data Interchange Client 3.2 after migrating configuration data, you must edit the system data and adjust Data Source Names and Qualifiers as needed to reflect WebSphere Data Interchange 3.2 databases.

  Note: The WebSphere Data Interchange Client Release Migration option uses unique data structures from the regular WebSphere Data Interchange export and import formats, and should not be intermixed with that capability. To use the import capability, the data must have been exported from WebSphere Data Interchange Client using WebSphere Data Interchange Client Release Migration option.

- When you migrate data from a previous release, you might overwrite some of the default load data that comes with WebSphere Data Interchange 3.2. You can recover some of the default load data using individual import files that are installed in the **samples** directory on the WebSphere Data Interchange server. The import files include:

Table 1. Sample import files

Import file	Description
appdef,.eif	Application defaults profiles
envstd.eif	Envelope standards
famaps.eif	Functional acknowledgement maps
netprof.eif	Network profiles
sampmap.eif	Sample map for install verification and API example

Table 1. Sample import files (continued)

Import file	Description
servprof.eif	Service profiles
tradpart.eif	Trading partner profiles (ANY trading partner)

Exporting data

To perform a release migration, export the data from DataInterchange 3.1 or DataInterchange 4.1 using the following procedure.

1. Select Release Migration from the View menu.

   The Release Migration wizard displays.

   Note: All list windows within WebSphere Data Interchange Client must be closed.

2. Select Export to create a set of files with data from Systems associated with the WebSphere Data Interchange Client you are using, and then click **Next**.

3. Select the type of data you are exporting, and then click **Next**. Your choices include:

   - System Data - only system data will be processed. System data is runtime data and customization time data.
   - Configuration Data - only configuration data will be processed.
   - Both - system data and configuration data will be processed.

4. Select one or more systems to be used by the release migration process you chose, and then click **Next**. If you chose Configuration Data in step 3, skip to step 6.

5. Select the system from which data will be exported by the release migration process, and then click **Next**. Your choices include:

   - Customization Time Data - EDI standards, data formats, and maps
   - Runtime Data - trading partners, other profiles, translate tables, and control strings
   - Both - Customization time and Runtime data

6. Type the migration path the data will be exported to. An existing path, from a previous WebSphere Data Interchange Client export, can be selected from the drop-down list. Click **Next**.

7. Click **Finish** to begin the process. A status box displays during the export.

Note: After you import configuration data, you should review your systems definitions to ensure the settings are correct, especially data source names.

Importing data

1. Select Release Migration from the View menu on WebSphere Data Interchange Client.

   The Release Migration wizard displays.

   Note: All list windows within WebSphere Data Interchange Client must be closed.

2. Select Import to populate the Client you are using with data from another version or release of the product, and then click **Next**.

3. Select the type of data you are importing, and then click **Next**. Your choices include:

Migrating between versions and releases

- System Data - only system data will be processed. System data is runtime data and customization time data.
- Configuration Data - only configuration data will be processed.

4. Select the path containing the migration data, and then click **Next**.

5. Select the system that the selected migration data source will migrate into, and then click **Next**. This screen displays only if more than one system is defined in the WebSphere Data Interchange Client.

6. Select the system into which exported data will be imported. Export data will be filtered to populate the selected system. After you make your selection, click **Next**. Your choices include:

   - Customization Time Data - EDI standards, data formats, and maps
   - Runtime Data - trading partners, other profiles, translate tables, and control strings
   - Both - Customization time and Runtime data

7. Confirm the options you have chosen for the import. To change the selected options, click **Back**; otherwise, click **Finish** to initiate the import.

   You can also use Release Migration to save and restore data in a system.

Naming convention changes for WebSphere Data Interchange Client

In the WebSphere Data Interchange Client interface, some of the terminology has been changed to reflect generally accepted conventions as well as requests from customers. Note the following changes in terminology:

Table 2. Differences in terminology used in WebSphere Data Interchange Host and WebSphere Data Interchange Client

WebSphere Data Interchange Host Terminology	WebSphere Data Interchange Client Terminology
Application data formats (ADF)	Data formats
Application definition profile	Application defaults
Generate a control string	Compile a map into a control string
Multi-occurrence structure mapping	Path qualified mapping
Network operation profile	Network commands
Requestor profiles	Mailboxes
Security profile	Network security
System profile	CICS performance
Trading partner transactions (TPT)	Maps
Translation tables	Forward and reverse translation tables
Validation tables	Code Lists

Host functions not available in the WebSphere Data Interchange Client interface

The following functions must be accessed through the WebSphere Data Interchange Host interface (for z/OS users), or directly from the WebSphere Data Interchange server:

- WebSphere Data Interchange utility
- Translation and communication options of the transaction store

Limitations

Chapter 3. Server installation and setup

This chapter provides instructions for installing and configuring the WebSphere Data Interchange Server on AIX and Windows 2000. It also provides details of hardware and software prerequisites for the Server. Before you install WebSphere Data Interchange Server, ensure that you have the required hardware and software listed here.

Instructions for uninstalling the Server are also provided.

Hardware requirements

WebSphere Data Interchange Server

Your hardware must meet the following minimum standards:
- Server capable of running the AIX or Microsoft Windows 2000 operating system
- CD-ROM drive for installing the distributed material
- Required network hardware and communication connections

If you are installing on Windows 2000, the following additional requirements apply:
- An Intel Pentium III processor at 933 MHz or faster
- 1024 MB of memory
- A storage device with a minimum of 8 GB free space

Software requirements

Your software must meet the following minimum standards:
- One of the following operating systems for the Server:
  - AIX V4.3.3
  - Windows 2000
- The following server databases and tools:
  - Administrative database supplied with the product
  - DB2 Universal Database Workgroup Edition V7.2 with open database connectivity (ODBC) (provided with the product)
- Optional operating products:
  - WebSphere MQ V5.3
  - Expedite/Base for AIX V4.5 or Expedite/Base for Windows V4.6

Installing WebSphere Data Interchange Server

WebSphere Data Interchange Server has an InstallShield Wizard that guides you through the installation process for AIX and Windows 2000 installations.

AIX

To use the InstallShield Wizard, you must be logged in as the root user. If you are running from a remote terminal, you must be using X-Windows and your DISPLAY environment variable must be set to your X-Server IP address.

By default, WebSphere Data Interchange Server is installed to the `/usr/wdi/DIv32` directory. You must have at least 50 MB of free space on this file system.

1. If you are installing from a CD-ROM, insert the WebSphere Data Interchange Server CD and mount the CD-ROM drive.

   a. Change to the directory containing the wdi.aix executable.

   b. Run the wdi.aix executable to start the InstallShield Wizard.

   The Welcome screen opens as the InstallShield Wizard prepares to install WebSphere Data Interchange Server. Click **Next**.

2. The license agreement opens. Read the information and license terms on the panel.

   Click the appropriate button to accept the terms of the license agreement and to indicate that you have read the notice and agree to its terms. Click **Next**.

3. The Installer dialog box displays a listing of the installation directory and the total size requirement. Click **Next**.

4. You are asked if you have purchased sufficient processor license units for the number of processors you have on your computer.

   Processor license units are used by IBM as the basis for product ordering as detailed in the license agreement.

   If you have enough processor license units, click **Yes**. The Installing panel is displayed.

   If you do not have enough processor license units, click **No** to cancel the installation.

5. The InstallShield Wizard begins copying program files. To stop this process at any point, click **Cancel**.

   The screen displays the successful installation message. Click **Finish**.

6. The installation of the files is complete. The following directories are created within the installation directory:

   bin
   bind
   ddl
   include
   ixf
   lib
   runtime
   runtime/adf
   runtime/aex
   runtime/dicmd
   runtime/dicts
   runtime/dtds
   runtime/edi

runtime/eex
runtime/fak
runtime/prt
runtime/qry
runtime/rcv
runtime/rpt
runtime/trk
runtime/wrk
runtime/xex
runtime/xml
samples

Additional directories might be created for use by InstallShield, for example for storing uninstall information, and Java™ Virtual Machine binaries.

For information on setting the PATH and setting up a TCP/IP alias to allow remote access to AIX databases, see the WebSphere Data Interchange Version 3.2 `readme.txt` file.

Windows 2000

To use the InstallShield Wizard, you must be logged in as an administrator.

By default, WebSphere Data Interchange Server is installed to the `C:\WDIServer32` directory. You must have at least 70 MB of free space on this drive.

1. Insert the WebSphere Data Interchange Server CD into the CD-ROM drive.
   a. On the menu bar, click **Start —> Run**.
   b. Find the directory containing the wdi.exe executable.
   c. Run the wdi.exe executable to start the InstallShield Wizard.

   The Welcome screen opens as the InstallShield Wizard prepares to install WebSphere Data Interchange Server. Click **Next**.
2. The license agreement opens. Read the information and license terms on the panel.

   Click the appropriate button to accept the terms of the license agreement and to indicate you have read the notice and agree to its terms. Click **Next**.
3. The Installer dialog box displays a listing of the installation directory and the total size requirement. Click **Next**.
4. You are asked if you have purchased sufficient processor license units for the number of processors you have on your computer.

   Processor license units are used by IBM as the basis for product ordering as detailed in the license agreement.

   If you have enough processor license units, click **Yes**. The Installing panel is displayed.

   If you do not have enough processor license units, click **No** to cancel the installation.
5. The InstallShield Wizard begins copying program files. To stop this process at any point, click **Cancel**.

   The screen displays the successful installation message. Click **Finish**.
6. The installation of the files is complete. The following directories are created within the installation directory:

Installing the server

```
bin
bind
ddl
include
ixf
lib
runtime
runtime\adf
runtime\aex
runtime\dicmd
runtime\dicts
runtime\dtds
runtime\edi
runtime\eex
runtime\fak
runtime\prt
runtime\qry
runtime\rcv
runtime\rpt
runtime\trk
runtime\wrk
runtime\xex
runtime\xml
samples
```

Additional directories might be created for use by InstallShield, for example for storing uninstall information, and Java Virtual Machine binaries.

For information on setting the PATH, see the WebSphere Data Interchange Version 3.2 `readme.txt` file.

Displaying and setting processor license units

You must purchase the correct number of processor license units to support your use of WebSphere Data Interchange on this computer. When you install the server, the installation program detects the number of processors on the computer. It asks you to confirm that you have purchased the appropriate number of processor license units. If you indicate that you have, this number is stored in a file within the install directory structure.

You can display the setting for processor license units using the command **edigcap**. You can update the setting using the command **ediscap**.

When running ediscap and edigcap on AIX, it is important to have the environment variable $DIROOT to be set to the installation directory of WDI. If DIROOT is not set, then the following error messages are displayed:

```
Unable to retrieve capacity information.
  Please define your DIROOT environment variable.
```

In Windows, If the registry becomes corrupt, then the following message will be displayed:

```
Unable to retrieve capacity information.
 Please verify that DI was properly installed.
```

The syntax for edigcap is:

```
edigcap
```

The syntax for ediscap is:

```
ediscap <number of capacity units>
```

Setting up the WebSphere Data Interchange database

Perform the following steps as a user with DB2 administrator authority.

1. If you are installing on Windows, using your DB2 administrator user ID, select **Start —> Programs —> IBM DB2 —> Command Window** to open the DB2 Command window. If you are installing on AIX, log in as a user with administrator authority.

   The remaining database setup steps use this command window or login session.

2. Change to the `ddl` directory under the installation directory.

   a. Issue the following command:
      ```
      db2 create db ediec32e
      ```

      When this process has successfully completed, the database has been built.

   b. Now issue the following command:
      ```
      altrec32
      ```

      This process alters some of the default parameters related to log file size and to the number of primary and secondary logs.

3. Change to the DB2 directory that contains the bind files for the DB2 utilities:

   - If you are installing on AIX, this is typically `/u/<db2_instance>/sqllib/bnd`, where <db2_instance> is the userID of the instance owner.

   - If you are installing on Windows 2000, this is typically `C:\Program Files\SQLLIB\bnd`.

   For more information, see the section "Binding Database Utilities" in the appropriate *DB2 Quick Beginnings* book.

   Issue the following commands:
   ```
   db2 connect to ediec32e
   db2 bind @db2ubind.lst messages bind.msg grant public
   db2 bind @db2cli.lst messages clibind.msg grant public
   db2 connect reset
   ```

   If you are working on AIX, you might not have write authority in the current directory. If you do not, you must specify a different file for the messages, for example `/tmp/bind.msg`.

4. Change back to the `ddl` directory under the installation directory.

   a. Issue the following command:
      ```
      db2 -tf ediec32.ddl -l ec32.log
      ```

      This creates the WebSphere Data Interchange tables, indexes, views, and so on.

   b. Invoke the command to issue the GRANT statements that are required to grant access to the newly created tables for the WebSphere Data Interchange Client. By default this GRANTs access to public. You might want to change public to specific user IDs or a group of authorized users.
      ```
      db2 -tf grntec32.ddl -l grntec32.log
      ```

5. Change to the `ixf` directory under the installation directory.

Setting up the server database

Issue the following command:

```
loadec32
```

This process loads initial data into the DB2 tables. The loading might generate warnings, but you can safely ignore these.

6. Change to the `bind` directory under the installation directory.

   Invoke the command to bind the WebSphere Data Interchange DB2 packages and issue the GRANT statements that are required to grant access to the newly created tables for the WebSphere Data Interchange Server. The default in `bindgrnt.fil` GRANTs access to public. You might want to change public to specific user IDs or a group of authorized users.

   ```
   db2 -tf bindgrnt.fil -l bind.log
   ```

   You have now completed the set up of the WebSphere Data Interchange database.

Verifying your installation

To verify your installation of WebSphere Data Interchange Server, complete the following steps:

1. Change directory to the samples directory
2. Enter the command:

   ```
   wditest
   ```

   This command runs a batch file on Windows, and a shell script on AIX to set up environment variables and run a translation using the sample map and data provided. The environment is restored at the end of the test.

If the installation is successful, the following output is returned:

```
DI Translator Started, build date: (WDI build date)
DI Translator processed your request.
DI Translator shutdown
```

If an error or errors are written to the console or to the prtfile in the samples directory check the messages and take the appropriate corrective action.

Uninstalling WebSphere Data Interchange Server

WebSphere Data Interchange Server provides programs that help you to uninstall the product on AIX and Windows 2000.

AIX

To uninstall WebSphere Data Interchange Server on AIX:

1. Change to the `_uninst` directory under the installation directory.
2. Issue the command:

   ```
   uninstall.bin
   ```

Windows 2000

To uninstall WebSphere Data Interchange Server on Windows 2000:

1. Select **Settings—Control Panel** from the **Start** menu.
2. Select **Add/Remove Programs**.
3. Select the WebSphere Data Interchange Server product entry and click **Remove**.

Chapter 4. The WebSphere Data Interchange Client interface

WebSphere Data Interchange Client is designed to make setup, maintenance, and management of WebSphere Data Interchange easier by using the Microsoft Windows graphical environment.

Through the WebSphere Data Interchange Client interface you can create, update, and manage the following WebSphere Data Interchange components:

- Setup profiles
- Trading Partner profiles
- Maps
- Data formats
- EDI Standards

You can also import eXtensible Markup Language (XML) Document Type Definitions (DTDs) into WebSphere Data Interchange Client and create and print reports.

Using windows

WebSphere Data Interchange Client displays information in three types of windows:

- List
- Tree
- Editor

This section describes those windows and explains how to control them.

List windows

The purpose of a list window is to display a list of items of a given type, such as a list of trading partners. List windows allow you to choose items on which you want to perform such actions as editing, printing, deleting, and renaming.
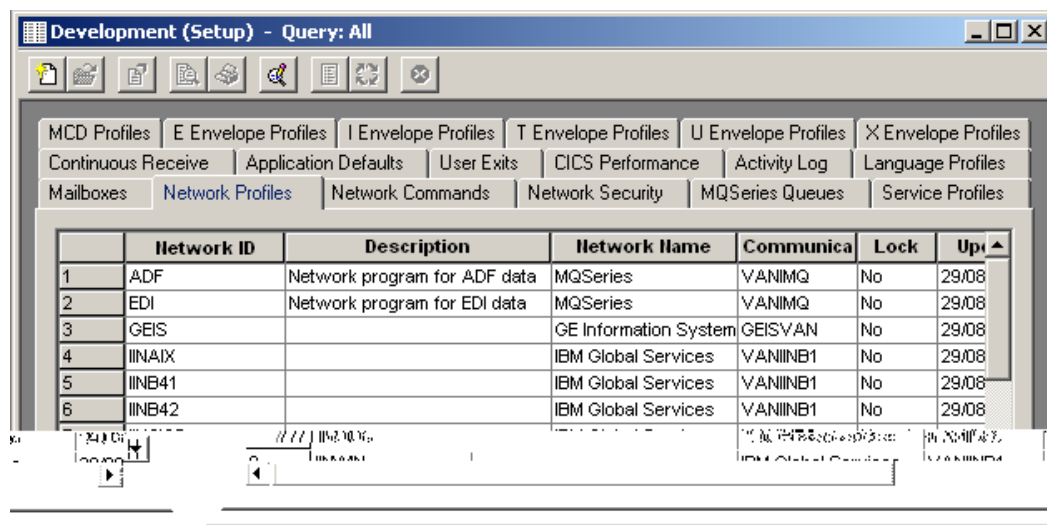
Figure 5. List window example

List windows

The rows you see displayed in a list window tab, as illustrated above, are the result of a query against the database. The column names represent information stored in the database, which you select when you define your list window queries. To change that information, you use an editor window, as described on page 38. The list window also contains the date, time, and user ID of the last update.

Modifying list window information

You can control the information that is displayed in list windows, as well as the way the information displays.

To display additional columns in a list window: Click the scroll bar on the bottom of the screen to scroll to the right or left.

To select the columns that display on the screen:

1. Click Modify Window Properties (...). Refer to Table 6 on page 42

    The Object List Window Query dialog box displays.

2. If you want to change the current query, select a query from the Current Query drop-down list.

    Columns that are available in the selected query display in the Selected Columns or Available Columns list boxes. You can select the columns you want to display in the window by completing this procedure.

    Note: You need not create a new query to change the columns that display in the list window.

3. Select the columns you want to display in the list window from the Available Columns list box.

    Columns in the Selected Columns list box already display in the window. Use the < button to move a column to the Available Columns list box so that it does not display.

    The > button moves columns back to the Selected Columns box. The << button and >> button move all the columns in one list box to the other.

    Note: The columns in the Selected Columns list box are the columns that have been set up in the current query. To modify the query or create a new one, Chapter 30, "Queries" on page 267.

4. When you have finished selecting the columns you want to display in the list window, click OK.

    The modified query is executed and the list window displays again with your new column selections.

    If you are working in a list window opened through the Navigator bar, your column selection is saved as that window's default. The next time you open that list window, you will see that column setup with the data displayed as you left it.

    You can also change the width of each column in a list window.

To change the width of a column:

1. Move the cursor pointer over the line that divides columns until the cursor changes to a double arrow.
2. Click and hold down the left mouse button.
3. Drag the arrow to either increase or decrease the width of the column and release the mouse button.

Tree windows

WebSphere Data Interchange Client also allows you to view lists in Tree View, rather than in the tabbed list window.

To use tree view

1. Click the plus (+) sign to expand the tree.
2. The queries associated with that branch of the tree display.
3. Double-click a query to run it and display items in the list window.
4. Double-click an item to open its editor window.

To set the default display to Tree view, "Selecting list window options" on page 50
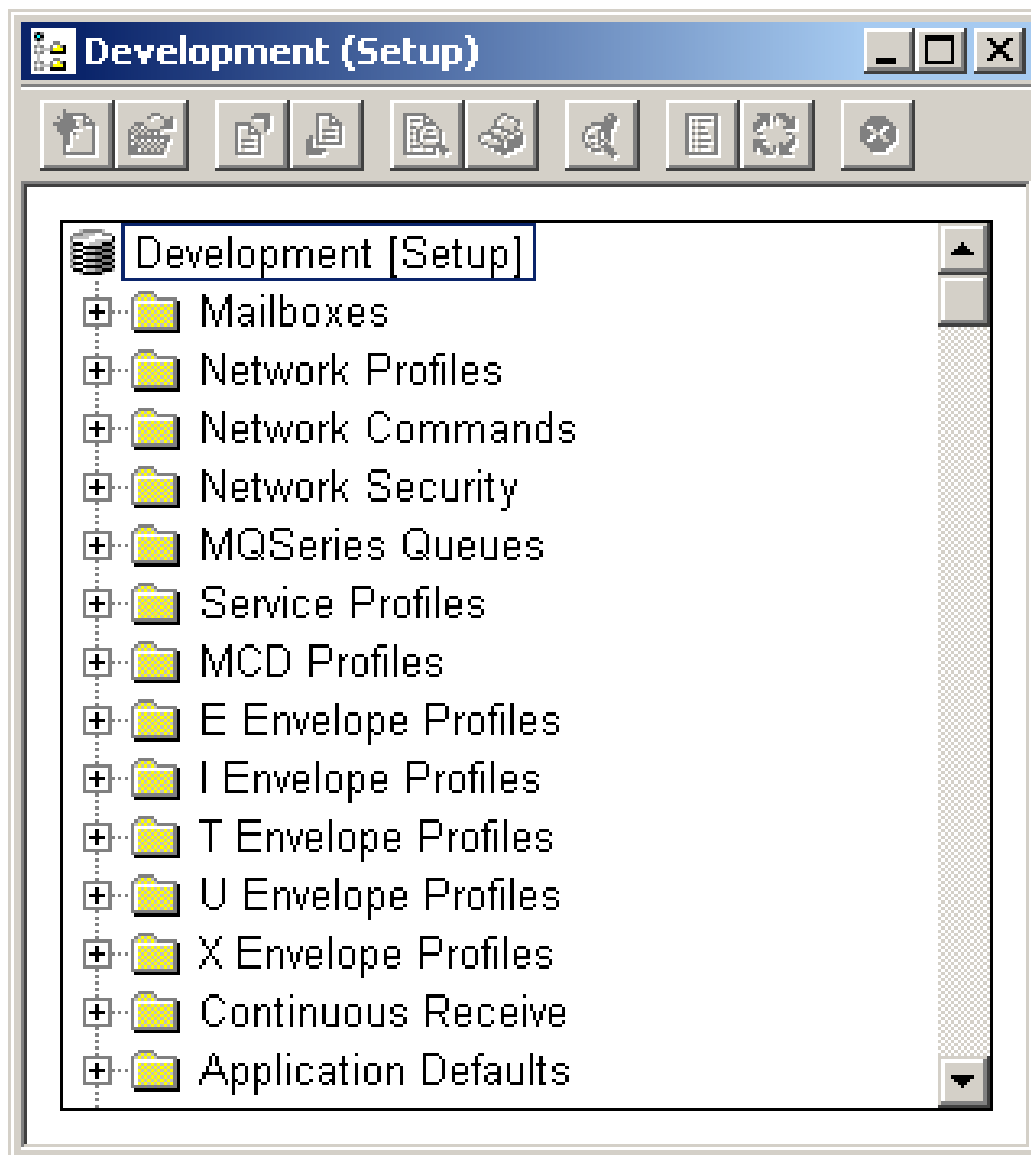
Figure 6. Tree View Example

Editor windows

Editor windows display when you open an item in a list window. They feature tabs that contain fields, drop-down lists, check boxes, and other Windows controls, as illustrated:
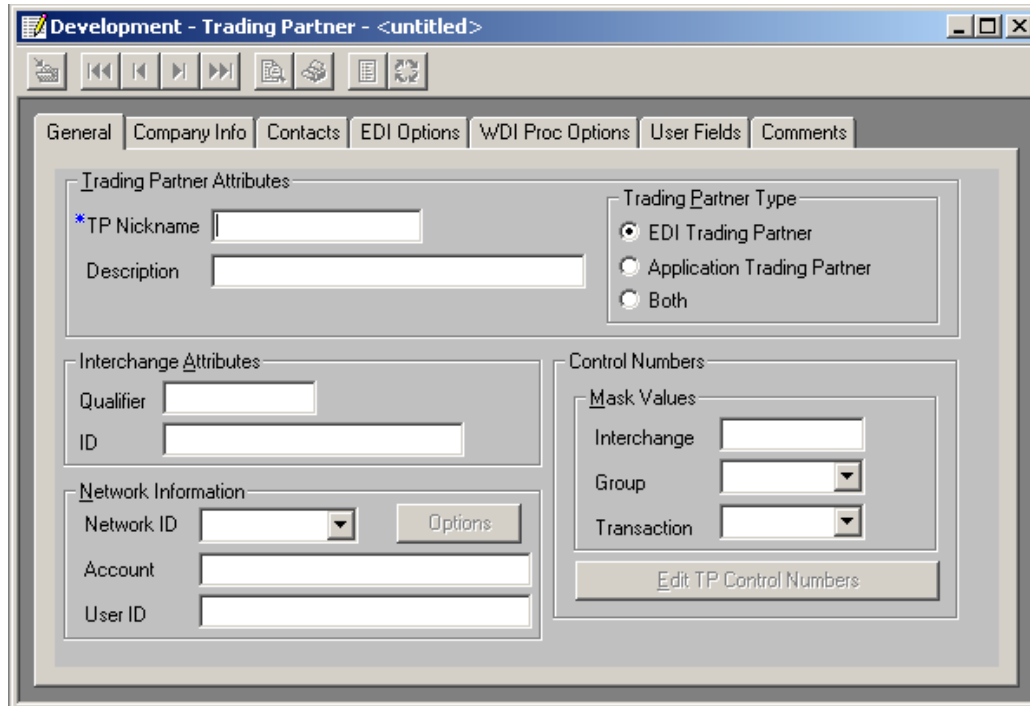
Figure 7. Editor Window example

Many of the fields and control names display as column names across the top of list windows. Enter information into editor windows using the keyboard, mouse, and Windows controls.

Note: When creating a new item, required fields in editor windows are preceded by a blue asterisk.

List and editor windows work like any other standard window in Microsoft Windows. You can minimize and maximize windows and move them around within the application. For more information on using Windows, see your Microsoft Windows documentation.

To close a window, do any of the following
- Click the X in the upper right corner, or press Ctrl+F4 keys.
- Click the Control Menu in the upper left corner of the window and select Close.
- Select Close from the File menu.

For more information on the Control Menu, see your Windows documentation.

Selecting commands

WebSphere Data Interchange Client offers three different methods of selecting commands:
- Menus

You can perform an action by selecting its command from a menu using the mouse or keyboard.

- Tool bars

  You can perform an action by clicking its button using the mouse.

- Shortcuts

  You can perform an action using keyboard or mouse shortcuts.

The following sections describe those methods.

Menus

Menu options offer the most comprehensive method of performing common WebSphere Data Interchange functions. Using the menu options you can: open new and existing items; create a query or a report; import and export components; save, close, preview, and print items; navigate through lists of items; set viewing preferences; access help; and exit WebSphere Data Interchange Client.

Most of the options in the menus are self-explanatory. The options that display under the menus (File, Actions, Edit, Navigate, View, Window, and Help) vary depending on what function you are using. Whether or not options are available (that is, whether they appear black instead of gray) often depends on, for example, which tab is in front.

Table 3, illustrates when the various menu options are available. Table 4 on page 40, illustrates what action you need to perform in order to activate various menu options.

Table 3. Availability of menu options

Options on these menus are available when. . .	. . .No window is open	. . . A list window is active	. . . An editor window is active
File Menu	Open Functional Area Open Import File Open Query List Open Report List Exit	New Open Close Open Functional Area Open Import File Open Query List Open Report List Print Print Preview Print List Properties Exit	Save Close Open Functional Area Open Import File Open Query List Open Report List Print Print Preview Exit
Actions Menu		Copy Rename Delete Export to File Export to Other System Usages Compile (Generate) Create Standard from Data Format Unlock	Export to File Usages Compile (Generate) Create Standard from Data Format

Selecting commands

Table 3. Availability of menu options (continued)

Options on these menus are available when. . .	. . .No window is open	. . . A list window is active	. . . An editor window is active
Edit Menu		Create Standard from Data Format	Cut Copy Paste
Navigate Menu			Move First Move Previous Move Next Move Last
View Menu	Status Bar Navigator Bar Preferences Message Log Event Log Systems Customize DI Client Release Migration	Stop Loading Refresh Tool bar Status Bar Navigator Bar Preferences Message Log Event Log Systems Customize	Tool bar Status Bar Navigator Bar Preferences Message Log Event Log Systems Customize
Window Menu		Cascade Tile (List of open windows)	Cascade Tile (List of open windows)
Help Menu	Contents Search for Help on How to Use Help WebSphere Data Interchange on the Web About WebSphere Data Interchange	Contents Search for Help on How to Use Help WebSphere Data Interchange on the Web About WebSphere Data Interchange	Contents Search for Help on How to Use Help WebSphere Data Interchange on the Web About WebSphere Data Interchange

Table 4. Activating menu options

To display these menu options. . .	Do this:
Actions Menu	Open any window.
Edit Menu	Open an editor window.
Navigate Menu	Open an editor window.
Open option, File menu	Highlight an item or items in the list window.
Save option, File menu	Create a new document and fill in at least one field or open an existing document and make a change.
Print Preview option, File menu	Highlight an item or items in the list window or open an existing document.
Print option, File menu	Highlight an item or items in the list window or open an existing document.
Copy option, Actions menu	Highlight an item or items in the list window. Not all items can be renamed.
Rename option, Actions menu	Highlight an item or items in the list window. Not all items can be renamed.
Delete option, Actions menu	Highlight an item or items in the list window.

Table 4. Activating menu options (continued)

To display these menu options. . .	Do this:
Export to File option, Actions menu	Highlight an item or items in the list window. (Note that not all items can be exported.)
Export To Other System option, Actions menu	Highlight an item or items in the list window. (Note that not all items can be exported.)
Usages option, Actions menu	Highlight a map, or maps, or a trading partner or trading partners in its list window, or open the editor window of an existing map or trading partner.
Compile option, Actions menu	Highlight a map or maps or a control string or control strings in their list window from the map editor or the envelope standards.
Create Standard from Data Format	Highlight a data format from a data format list window or open a data format editor.
Unlock option, Actions menu	Highlight an item or items in the list window.
Cut option, Edit menu	Highlight a piece of text.
Copy option, Edit menu	Highlight a piece of text.
Paste option, Edit menu	Cut or copy a piece of text.
Navigate menu options	Select several items from a list window and open an editor window.

Tool bars

You can use tool bar buttons as shortcuts to perform most of the frequently used WebSphere Data Interchange Client menu commands. The three WebSphere Data Interchange Client tool bars are:

- Navigator bar
- List Window tool bar
- Editor Window tool bar

Navigator Bar

To display the Navigator bar at the top of the WebSphere Data Interchange Client main screen, ensure it is checked in the View menu. Navigator bar buttons open WebSphere Data Interchange Client's functional areas. Table 5 summarizes the Navigator bar buttons.

Table 5. Navigator bar buttons

This button. . .	Does this:
	Opens the Setup List window.
	Opens the Trading Partner List window.
	Opens the EDI Standards List window.

Selecting commands

Table 5. Navigator bar buttons (continued)

This button. . .	Does this:
	Opens the Data Formats List window.
	Opens the XML List window.
	Opens the Mapping List window.
	Opens the Transaction Store List window.
	Selects the system.
	Opens Help.

List window tool bar

Each work area in WebSphere Data Interchange Client contains lists of items, which display in list windows. Each list window contains a tool bar. Table 6 illustrates the buttons that display on WebSphere Data Interchange Client's list window tool bar.

Table 6. List window tool bar buttons

This button. . .	Does this:
 New	Creates a new item.
 Open	Opens the selected item from the list window.
 Export	Exports the selected item to a file.
 Preview	Previews the print format for the selected item.

Table 6. List window tool bar buttons (continued)

This button. . .	Does this:
Print	Prints the selected item.
Properties	Modify Window Properties. Selects or modifies a query that is run to load the list window.
Usages	Displays which maps are associated with a trading partner or which trading partners are associated with a map. Use this button to create a usage or map rule, as well as to display usages or map rules.
Compile	Compiles the selected map or envelope standard into a control string or recompiles the selected control string.
Stop Loading	Stops building the list of items in a list window. This allows you to stop WebSphere Data Interchange from building long lists when you see the item you are looking for.

Editor window tool bar

Each list window contains a list of components that you create and maintain using editor windows. Each editor window contains a tool bar. Table 7 illustrates the buttons that display on WebSphere Data Interchange Client's editor window tool bar.

Table 7. Editor window tool bar buttons

This button. . .	Does this:
Save	Saves the component. Enabled after any data in the window has been changed.
Move First	Displays the first component you selected in the editor window, if you selected more than one.
Move Previous	Displays the previous component you selected in the editor window, if you selected more than one.

Selecting commands

Table 7. Editor window tool bar buttons (continued)

This button. . .	Does this:
Move Next	Displays the next component in the editor window, if you selected more than one.
Move Last	Displays the last component you selected in the editor window, if you selected more than one.
Preview	Previews the print format for the selected component.
Print	Prints the selected component.
Usages	Displays which usages and map rules are associated with a trading partner or which usages or map rules are associated with a map.
Compile	Compiles the selected map into a control string or recompiles the selected control string. Also, compiles the selected envelope standard into an envelope standard control string or recompiles the selected control string.

Shortcuts

WebSphere Data Interchange Client allows you to select menu commands using standard Windows keyboard or mouse shortcuts. You can also use the keyboard to select items in list and editor windows, as well as navigate through fields in editor windows and grids.

To select a menu command using the keyboard

1. Press the Alt key.

   The cursor moves to the menu bar.

2. Press the key corresponding to the underlined letter, usually the first, of the menu you want to display.

   The menu displays.

3. Press the key corresponding to the underlined letter, usually the first, of the command you want to execute.

   The command executes.

   Note: You can also select commands by pressing the F10 key rather than the Alt key. Use the left and right arrow keys to move from menu to menu. Use the up and down arrow keys to scroll through the options on each menu.

The keyboard makes it easy to navigate through items and components in list windows and fields in editor windows, as summarized in Table 8.

Table 8. Keyboard and mouse shortcuts

Use this shortcut. . .	To:
Ctrl+N	New
Ctrl+O	Open
Ctrl+P	Print
Alt+F4	Close
Tab	Jump from one field to the next in the editor windows and grid editors.
Shift+Tab	Move back a field in editor windows.
Ctrl+Tab	Flip from window to window.
Up and down arrows	Choose the item above or below the selected items in list windows.
Enter key	Scroll down the grid editor.
Shift+down arrow or mouse drag	Select a range of sequential items in a list window or items in an editor window.
Ctrl+left mouse click	Select discontiguous items in a list window or items in an editor window.
Ctrl+Enter	Deselect a selected item.
Double-click the left mouse button	Open a window.

Note: Because double-clicking is the fastest way to execute a command, this manual uses it as the preferred method in procedures. You can also use the shortcut keys.

Performing common file management tasks

WebSphere Data Interchange Client is designed to provide consistent actions across the user interface. Both list windows and editor windows follow consistent procedures for performing the common file management tasks of viewing, copying, editing, renaming, deleting, and printing items and components.

Procedures for importing and exporting items and components are also standard across WebSphere Data Interchange Client windows. For details, Chapter 6, "Export and Import" on page 67

Viewing an item

1. Click the button on the WebSphere Data Interchange Client Navigator bar corresponding to the functional area you require.

   The list window displays.
2. Click the tab within the list window corresponding to the item you wish to view.

   A list displaying those items for the selected tab displays.
3. Double-click the item you wish to view.

   The item displays in its editor window.

File management tasks

Copying an item

The copy function allows you to duplicate an item within the WebSphere Data Interchange system in which you are working. If you want to base a new component on an existing component, for example, copy the existing component under a new name and edit it to the new specifications. For some objects, such as Trading Partner profiles, you will need to change some of the object details when creating the new object, such as the object name.

1. Click the button on the WebSphere Data Interchange Client Navigator bar corresponding to the functional area you require.

   The list window displays.

2. Click the tab within the list window corresponding to the item you wish to view.

   A list displaying the items for the selected tab displays.

3. Click the name once of the item you want to copy.

4. Select Copy from the Actions menu.

   The Copy Object dialog box displays with one or more fields used to name the component.

5. Type in one or more new values, and click OK.

   WebSphere Data Interchange Client copies the item.

Editing an item

If you want to update data in an item, open the editor window for that item, make your changes, and save them.

1. Click the button on the WebSphere Data Interchange Client Navigator bar corresponding to the functional area you require.

   The list window displays.

2. Click the tab within the list window corresponding to the item you wish to view.

   A list displaying the items for the selected tab displays.

3. Double-click the item you want to edit.

   The item's editor window displays.

4. Change values for fields as required.

5. Click Save on the tool bar to save the changes.

Renaming an item

1. In the appropriate list window, select the item you wish to rename.

2. Select Rename from the Actions menu.

   The Rename Object dialog box displays.

3. Type in a new name and click OK.

   WebSphere Data Interchange Client renames your item.

   Note: Your database system must support Cascade Update for rename to be fully functional. If Cascade Update is not supported by your database system, Rename will only work on simple objects. DB2 does not support Cascade Update.

Deleting an item

1. In the appropriate list window, select the item you wish to delete.

2. Select Delete from the Actions menu.

   A confirmation displays.

3. Click Yes if you want to delete the item.

   WebSphere Data Interchange Client displays a message in the Execution Status window when it has completed the deletion. Click Close to close the Execution Status window.

Printing an item

1. In the appropriate list window, select the component you wish to print.
2. If you want to preview the item, click Print Preview on the tool bar.

   WebSphere Data Interchange Client shows you a preview of the printed document.
3. Click Print on the tool bar.

   A Print dialog box that allows you to select printers and other print options displays.
4. Click OK.

   WebSphere Data Interchange Client sends the item to the default or selected printer.

   Note: You can also double-click an item in its list window to open the editor window, and print from there.

Using editor window grids

Some WebSphere Data Interchange Client editor windows use a grid system for editing. In addition to entering information in rows, as in a spreadsheet, you also use standard Windows controls, such as drop-down lists and check boxes. Table 9 summarizes the procedures for using the grid editors.

Table 9. Grid editor procedures

To. . .	Do this:
Enter a new row	Type in the row designated by an asterisk (*).
Insert a new row	Select the row (by clicking the row number) which you would like to display below the row you want to insert, and click Insert. A new row designated by an * displays. If the selected row is below the current * row, the * row moves below the selected row. If the selected row is above the * row, the * row moves above the selected row.
Edit information in a row	Click the cell that you want to edit. If the cell you selected has a drop-down list, an arrow displays on the right. Click the arrow to view the drop-down list, and then select your entry from the list. If there is no drop-down list, you see a cursor when you click the cell. Type in information. On some drop-down lists, you can get a cursor by double-clicking the cell. If a cell contains numeric information, up and down arrow keys display when you double-click the cell. Use these keys to scroll to the number you want to enter in that cell.

Editor window grids

To. . .	Do this:
Edit a component	Click the row number to select the entire row, and then click Edit or double-click the row number. The appropriate Editor displays.
Delete a row	Click the row number to select the entire row, and then click Delete.

Note: You cannot edit all columns in a grid. Columns you cannot edit have a gray background.

Working with multiple systems

WebSphere Data Interchange allows you to work in as many systems as you like at the same time. This can be very convenient for comparing information in the test and production systems, for example. This section shows you how to select systems and manage their windows.

Selecting a system

If you open list windows by clicking their respective buttons on the tool bar, they automatically open in the default system. WebSphere Data Interchange Client displays the name of your current default system in the System box on the Navigator bar.

You can select the system by choosing a system from this drop-down list, or by opening items from the Open Functional Area option in the File menu.

To select a system using the Open Functional Area option
1. Select Open Functional Area from the File menu.

   The Open Functional Area window displays.
2. In the System list box on the left, click the system in which you wish to work.
3. In the Functional Area list box on the right, select the area in which you wish to work, for example Transaction Store.

   Note: Selecting an area in the Functional Area list box is the same as clicking a button on the Navigator tool bar.
4. If you wish to view your work in a tree window, rather than in list windows, click the Display as a Tree View check box.
5. Click OK.

   WebSphere Data Interchange Client opens the list window of the Functional Area you selected in the System you selected. If you select a system that you have not worked in since you started WebSphere Data Interchange Client, you may be required to log on.

Understanding window title bars

The window title bar displays important information for keeping track of which system you are working in. The name of the system is on the title bar of each WebSphere Data Interchange Client window. The name of the functional area you are working with displays to the right of the system name in parentheses. The

current query displays to the right of the functional area in list windows, the current open item displays to the right of the functional area in editor windows.

Note that you can have multiple windows from multiple systems open at the same time. Set the window background of each system to a different color using the Preferences command on the View menu, as described in "Selecting the system color" on page 50. This makes different systems easily distinguishable.

You can cascade or tile windows by selecting the appropriate option from the Windows menu.

Note: If you have multiple windows open, you might find that an object is locked when you try to work with it. Check that you do not already have the object window open by cascading the windows. If the object is already open, close the new window.

Setting preferences

WebSphere Data Interchange Client allows you to set preferences for what you view on the screen and how it is viewed. You can also set preferences for the Message Log.

Setting window preferences

When you start WebSphere Data Interchange Client, it maximizes the main window to full screen and restores any list windows that you left open upon exit. Change those defaults through the Preferences dialog box.

To set the main window display option

1. Select Preferences from the View menu.

   The Preferences dialog box displays.

2. Click the Maximize Main Window check box.

   Inserting a check sets the default window size so that it displays at maximum size when you start WebSphere Data Interchange Client.

   Removing a check sets the default so that the Main window displays on the screen at the same size you set it before you last exited WebSphere Data Interchange Client.

3. Click OK.

To set the list window default

1. Select Preferences from the View menu.

   The Preferences dialog box displays.

2. Click the Restore List Windows check box.

   Inserting a check sets the default so that list windows left open upon exit display when you restart WebSphere Data Interchange Client.

   Removing a check sets the default so that list windows left open upon exit are closed when you restart WebSphere Data Interchange Client.

3. Click OK.

Setting message log preferences

WebSphere Data Interchange Client maintains a Message Log that captures information on errors and problems. This information is used by IBM support. You can disable the Message Log and change the length of time that it stores messages.

To disable the message log

1. Select Preferences from the View menu.

   The Preferences dialog box displays.
2. Click the Disable Message Logging check box to insert a check.
3. Click OK.

   The next time you start WebSphere Data Interchange Client, the Message Log will not collect data.

To set the length of time the message log stores messages

1. Select Preferences from the View menu.

   The Preferences dialog box displays.
2. Check the Delete Messages Older than X Days check box, and type the number of days you wish to store messages.

   If you do not want to delete messages from the log, click the check box to remove the check.
3. Click OK.

   The Message Log will purge messages older than the number of days you entered when you restart WebSphere Data Interchange.

Selecting the system color

To set window color preferences

1. Select the system whose window background color you wish to change through the Systems list.
2. Select Preferences from the View menu.

   The Preferences dialog box displays.
3. Click Change in the Color Options list box.

   The standard Windows Colors dialog box displays.
4. Click the color you want used for the window background.
5. Click OK to close the Colors dialog box.
6. Click OK to close the Preferences dialog box.

   The background of all windows in the selected system displays in the color you selected.

   Note: Colors can also be changed by editing the system (View -> Systems).

Selecting list window options

WebSphere Data Interchange Client displays information in list windows. You can view information using any of the following options:

- Multiline tabs

  When you open list windows either by clicking on Navigator bar buttons or using the Open Functional Area command on the File menu, information displays in a tabbed list window. If there are too many tabs to fit into the list window, the list window displays with multiple lines of tabs.

- Single line tabs

  When you open list windows either by clicking on Navigator bar buttons or using the Open Functional Area command on the File menu, information displays in a tabbed list window. If there are too many tabs to fit into the list window, scroll buttons appear on the right side of the tabs. Use these buttons to scroll tabs into view.

- Tree view

  When you open list windows either by clicking on Navigator bar buttons or using the Open Functional Area command on the File menu, the list window displays information in a tree view.

To select the list window view

1. Select Preferences from the View menu.

   The Preferences dialog box displays.
2. Click the appropriate radio button for your view choice.
3. Click OK.

Customizing field tags

WebSphere Data Interchange Client allows for the customization of User Field 1 through User Field 10 tags on the Trading Partner User Fields tab page.

To customize the labels

1. Select Customize from the View menu.

   The Customize editor displays.
2. Double-click the list entry you want to rename.

   An editor page displays.
3. In the Displayed Field Label edit box, type the new name of the User Field.
4. Click Save, and close the window.
5. Repeat the process for the other tags you want to change.

Viewing control and status bars

WebSphere Data Interchange Client allows you to control which tool bars to view on the screen, if any. You can also choose whether to view the Status bar, which runs along the bottom of the screen and provides system information.

The Navigator bar, tool bar, and Status bar all display on the screen by default. You may hide any or all bars.

To hide a control or status bar

On the View menu, select the bar you wish to hide by clicking it to remove the check mark. That bar no longer displays on the screen. Bars preceded by a check mark on the menu display on the screen.

Getting help

WebSphere Data Interchange Client includes online, context-sensitive Help that allows you to display information about virtually any aspect of the program. In most cases, getting the help you need is as easy as clicking a Help button or pressing the F1 key.

The Help system contains information about each WebSphere Data Interchange Client screen and field and explains how to use the operations presented to you.

You also have access to the WebSphere Data Interchange Client glossary of terms, error messages, and WebSphere Data Interchange reference material. Each option on the Help menu is described briefly below.

Contents

Select this option to display the Contents screen for WebSphere Data Interchange Client Help. This screen allows you to access all of the major areas of Help by clicking the name of an area.

Search for help on

Select this option to display the Search dialog box. Type a word in the field to display a list of index entries beginning with that word. Select an index entry, and then click Display to view the information related to the index entry you selected.

How to use help

Select this option to display the Windows help topics that describe Windows Help.

WebSphere Data Interchange on the Web

Select this option to display a submenu with links to different pages at the WebSphere Data Interchange Web site, including:

- WebSphere Data Interchange Home Page
- Fact Sheet
- Downloads
- Frequently Asked Questions
- Technical Support

About WebSphere Data Interchange Client

Select this option to display a dialog box that contains the WebSphere Data Interchange Client version number, as well as the percentage of system resources and memory currently available.

Accessing online help

There are three ways to access WebSphere Data Interchange Client Help:

- Help Buttons

  Some WebSphere Data Interchange Client windows and dialog boxes include a Help button. Clicking Help displays a Help window that contains information specific to the current window or dialog box. When you are through reading the information, double-click Control Menu in the upper-left corner of the window to exit Help and continue working.

- Help Menu

  The WebSphere Data Interchange Client Help menu provides direct access to the Help Contents, as well as Help topics for major tasks like creating and editing documents. Select an option from the menu to display that portion of Help.

- F1 Key

  WebSphere Data Interchange Client's help is context sensitive, which means if you click a window, and then press the F1 key, WebSphere Data Interchange Client displays the relevant help topic.

  Some WebSphere Data Interchange Client error boxes allow you to receive additional information by pressing the F1 key.

Using WebSphere Data Interchange Client help

WebSphere Data Interchange Client Help is designed to provide you with fast access to information when you run into a problem or when you just want to know more about a particular aspect of WebSphere Data Interchange Client. The starting point of the Help system is the Help Contents screen, which shows you the type of information available.

From the WebSphere Data Interchange Client Help Contents screen, you can get information about menu commands, see field definitions, display error messages, or view the glossary. You can also go directly to a number of specific Help topics on tasks, such as creating a document and communications. You can move freely through the system by clicking links.

Links

Links allow you to jump quickly from one Help topic to another simply by clicking a specially marked word or picture. Using hypertext links, you can browse through various Help topics or move from general information to more specific information.

In WebSphere Data Interchange Client Help, as in many other Help systems, links display in green and are underlined with either a solid or a broken line. Click a link to display the corresponding Help topic.

Words or phrases underlined with a solid line take you to different topics. Keywords underlined with a broken line pop up a definition in the current topic.

Help screen buttons

The following five buttons appear at the top of the Help screen:

- Contents

  This button returns to the Help Contents screen from any other Help screen.

- Search

  This button displays the Help Search window, which contains a list of keywords and phrases. Scroll through the list to find a word or phrase you wish to select, or type a word to begin scrolling automatically. Click a keyword, and then click Show Topics, to display a list of related topics. Select a topic and click Go To to display the Help topic.

- Back

  This button displays the previous Help topic. Click repeatedly to continue moving backward through previously accessed Help screens.

- Print

  This button allows you to print the open Help screen. Click to display a print dialog box that allows you to select printers and other print options.

- Glossary

  This button displays the WebSphere Data Interchange Client glossary.

WebSphere Data Interchange Client Help includes all the features available through Microsoft Windows Help. See Windows documentation for more information about Windows Help.

Using the message log

WebSphere Data Interchange Client displays messages about errors that occur within WebSphere Data Interchange Client, such as incorrect operations performed by the user or a failure to complete the current command. After displaying an error message, WebSphere Data Interchange Client logs it in the Message Log, which you access from the View menu.

Viewing messages

You can view and print the Message Log for your information. Note, however, that the list cannot be edited, and it can only be deleted with the Message Log Purging option available in the application preferences section of WebSphere Data Interchange Client, which is described on page 50.

To view the message log

1. Select Message Log from the View menu.

   The Message Log displays. See Table 10 for an explanation of the columns.

2. Double-click the item to view the entire message.

Table 10. Message log columns

This field. . .	Contains:
Updated Date/Time	The date and time the message occurred.
Updated User ID	The user who was on the system at the time the message occurred.
Message ID	A message identification number generated by WebSphere Data Interchange Client.
Message Text	The message text. You only see a little bit of the text. To view the entire message, double-click the text.
Module	The portion of WebSphere Data Interchange Client code in which the error occurred.
Line	The line number in the code in which the error occurred.

Chapter 5. Sending and receiving data

Before you can use WebSphere Data Interchange to translate data, or to send or receive transactions, messages, or files, you must define certain information. This information describes how your system sends and receives data, how data is formatted in your application files and mapped to a standard, to whom you send data and from whom you receive data, and other pertinent information.

WebSphere Data Interchange also features *any-to-any* translation. Using any-to-any translation, you can translate data from any supported source document type to any supported target document type. Supported document types include data formats, EDI standards, and XML data.

Any-to-any translation uses a new type of *map* called a *data transformation map*. These maps use a different command syntax than the send and receive maps. You use a different PERFORM command, PERFORM TRANSFORM, to translate data.

Setup for performing any-to-any translation

The following outlines the steps necessary for any-to-any translation:

1. Install and establish the environment.

   Install the WebSphere Data Interchange Server on AIX, Windows 2000, or z/OS. See the *WebSphere Data Interchange for z/OS Installation Guide* or Chapter 3, "Server installation and setup" on page 29.

2. Define your source and target documents.

   - If any source or target documents are EDI standards, add the EDI standards.

     Select the EDI standards you wish to use and add them to your WebSphere Data Interchange database. See Chapter 24, "EDI standards" on page 185 for more information.

     You can customize the EDI standards you have applied now or at a later time. Customizing can be helpful to alter a standard to suit a particular business need. See Chapter 24, "EDI standards" on page 185 for more information.

   - If any source or target documents are data formats, define the format of your application data to WebSphere Data Interchange.

     You use data formats to define the format of your application data to WebSphere Data Interchange. WebSphere Data Interchange supports several record formats. The data format definitions can be used to describe source documents, target documents, or both.

   - If any source or target documents are XML data, import your XML DTDs (document type definitions) into WebSphere Data Interchange.

     XML DTDs describe the format of your XML data. You can import DTDs directly into WebSphere Data Interchange using the WebSphere Data Interchange Client. See Chapter 6, "Export and Import" on page 67 for more information.

     Note: You can also import COBOL copybooks that contain application data layout.

3. Define your trading partners.

   Create trading partner profiles for each trading partner with whom you do business. You need to create a trading partner profile for each unique trading partner address. Trading partners can share a profile using the Minimal Trading

Partner feature. The profile contains information used to identify the trading partner as well as other data specific to the trading partner. Examples of the information in this profile are the trading partner's network and network address. See "Understanding processes and rules" on page 132

4. If you are using EDI Standards as target documents, customize the envelope profile.

   WebSphere Data Interchange provides default envelope profiles to supply information for envelope header and trailer records. You need to customize the sender ID field of the envelope profile you will be using. If you use more than one sender ID, you need to create an additional envelope profile for each sender ID you use.

5. Create your data transformation maps.

   The data transformation maps describe how the data is to be translated from the source document to the target document. For example, it tells the translator which source elements or fields correspond to which target elements or fields.

6. Identify which trading partners will use the maps.

   As part of mapping, you define which trading partners or process will use each map. This is done by creating data transformation rules. Rules also define special options, such as the type of envelope to use and the ID of the envelope profile. Maps can be used by multiple trading partners.

Translating data

To translate your data, simply place your data in a file or MQSeries® queue and invoke WebSphere Data Interchange. Use the PERFORM TRANSFORM command to provide the necessary information to WebSphere Data Interchange, such as the input data type and location . Typically, one or more applications can place application data into one or more files or queues over time. Periodically, a task is invoked to request the WebSphere Data Interchange Utility to translate the data in one or more of these source files to the selected target format(s). You can invoke multiple tasks simultaneously or at different times to translate the source data.

When errors occur during the translation process, the error messages are placed in a log and in a print file. Input data that could not translated due to errors is placed in an exception file.

Sending and receiving data

Additional setup is required before you can send data after translation or receive data before translation. The following outlines the necessary steps:

1. Verify/create a network profile.

   Select network profiles to communicate with your trading partners, and refer to them in your mailbox profiles. A network profile includes information, such as the network ID and the communications routine that you use. WebSphere Data Interchange contains profiles for IBM Global Services and the GEIS network.

2. Define yourself to WebSphere Data Interchange.

   Mailbox (requestor) profiles identify each group or individual in your organization who uses WebSphere Data Interchange to send data, receive data, or both. Examples of the information in a mailbox (requestor) profile are the requestor's network and login information.

Sending and receiving transactions and files

To send or receive your data, WebSphere Data Interchange starts a communication routine that uses an intermediate file to communicate with the network program.

The network program is supplied by you or another provider, such as a value added network (VAN). The program sends or receives the data over the network that you defined for communicating with your trading partner.

You can use the following functions to send and receive data. You can execute these functions using either the application program interface (API) or one of the PERFORM commands.

Table 11. Functions used to send and receive data

Function	Action
PERFORM SEND	Send EDI standard transactions
PERFORM SENDFILE	Send XML or application data files that do not include an interchange header and do not contain EDI data.
PERFORM RECEIVE	Receive EDI standard transactions.
PERFORM RECEIVEFILE	Receive XML or application data files that do not include an interchange header and do not contain EDI data

You can execute the SEND, SENDFILE, RECEIVE, and RECEIVEFILE functions as a separate task from your application, or you can execute them from within your application. See *WebSphere Data Interchange Programmer's Reference* for additional information about the WebSphere Data Interchange Utility and the application program interface.

Envelope overrides for EDI data

When the target document is an EDI standard, WebSphere Data Interchange provides you with the ability to override specific values in the interchange envelope with data provided by your application. A default envelope is selected if an override is not needed.

Here are some of the ways that envelope overrides are provided:
- One of the steps in the WebSphere Data Interchange trading partner definition is defining a data transformation rule. Use the data transformation rule to select the correct mapping for translating data. You can specify an envelope override using the data transformation rule.
- You can specify values in the envelope profile member that override the values that are generated during the envelope process. WebSphere Data Interchange uses the information in the profile member to generate the interchange header and trailer segments.

When the source document is an EDI standard, WebSphere Data Interchange uses the information you specified in the envelope profile member to generate the interchange header and trailer segments for a functional acknowledgment, if requested.

Setup for sending and receiving EDI data using send and receive maps

The following outlines the steps necessary for sending and receiving data that is formatted and mapped to a standard:
1. Install and establish the environment.

Install the WebSphere Data Interchange Server on AIX, Windows 2000, or z/OS. See the *WebSphere Data Interchange for z/OS Installation Guide* or Chapter 3, "Server installation and setup" on page 29.

2. Verify/create a network profile.

   Select network profiles to communicate with your trading partners. The network profile includes information, such as the network ID and the communications routine that you use. WebSphere Data Interchange contains profiles for IBM Global Services and the GEIS network.

3. Define yourself to WebSphere Data Interchange.

   Mailbox (requestor) profiles identify each group or individual in your organization who sends data, receives data, or both using WebSphere Data Interchange. Examples of the information in this profile are the requestor's network and login details.

   You can also define a nonspecific trading partner to WebSphere Data Interchange without using mailboxes. You can use a keyword of ANY to refer to any trading partner, even if the trading partner is not found in the trading partner profile. See "Understanding processes and rules" on page 132

4. Define your trading partners.

   Create a trading partner profile for each unique trading partner address for each trading partner with whom you do business. Trading partners can share a profile using the Minimal Trading Partner feature. The profile contains information used to identify the trading partner as well as other data specific to the trading partner. Examples of the information in this profile are the trading partner's network and network address. See "Understanding processes and rules" on page 132

5. Add the EDI and envelope standards.

   Select the EDI standards you wish to use and add them to your WebSphere Data Interchange database. You can customize the EDI standards you have applied now or at a later time. Customizing can be helpful to alter a standard to suit a particular business need. See the Chapter 24, "EDI standards" on page 185 for additional information.

6. Customize the envelope profile.

   If you use EDI, then you have the option of customizing your envelope profile. WebSphere Data Interchange provides default envelope profiles to supply information for envelope header and trailer records. You need to customize the sender ID field of the envelope profile you will be using. If you use more than one sender ID, you need to create an additional envelope profile for each sender ID you use. See the Chapter 11, "Envelope profiles" on page 93 for additional information about envelope profiles.

7. Define your data format to WebSphere Data Interchange.

   Use data formats to define the format of your application data to WebSphere Data Interchange. Several record formats are supported by WebSphere Data Interchange. One data format can be used to describe input application data that is sent to a trading partner and output application data that is received from a trading partner.

8. Map your application data to a standard.

   The data format you defined to WebSphere Data Interchange in the previous steps now needs to be mapped to a standard. This means associating the fields in your application data to the data elements in a standard. You will create a map for sending data, receiving data, or both. See the Part 4, "Mapping" on page 149 for complete mapping information.

9. Identify which trading partners will use the maps.

As part of mapping, you define which trading partners will use each map. This is done by creating send and receive usages or rules. Usages also define information used when sending (send map) or receiving (receive map) a transaction, such as the type of envelope to use and the ID of the envelope profile. Maps may be used by multiple trading partners. See the Part 5, "Administration" on page 265 for complete mapping information.

Sending EDI data using send maps

The following steps outline how to send data to a trading partner.

1. Translate application data to an EDI standard
2. Envelope the translated data
3. Send the enveloped data

Translating application data to an EDI standard

The first step in sending your data to a trading partner is to invoke WebSphere Data Interchange to translate your data to an EDI standard format. You can place your data in a file before invoking WebSphere Data Interchange or you can pass the data directly to WebSphere Data Interchange.

The Transaction Store holds your data during translation and provides a facility for you to access and manage your data. For more information, see *WebSphere Data Interchange for z/OS Administration Guide*.

When errors occur during translation processing, the error messages are placed in a log and a print file. The application data that could not be translated due to the errors, is placed in an exception file.

Collecting application data for translation
Typically, one or more applications place application data into one or more application files over time. Periodically, a task is invoked to request the WebSphere Data Interchange Utility to translate the data in one or more of these application files to a selected EDI standard. You can invoke multiple tasks simultaneously or at different times to translate the application data.

Translating application data as it is generated
An application can also place application data into a file and then start the WebSphere Data Interchange Utility or the WebSphere Data Interchange API directly to request translation of the application data to a selected standard. In this scenario, the application data can consist of one or more business documents for translation to an EDI Transaction. Multiple applications that use WebSphere Data Interchange can run simultaneously.

See the *WebSphere Data Interchange Programmer's Reference* for additional information on translating application data, the WebSphere Data Interchange Utility, and the Application Programmer's Interface.

Enveloping and sending transactions

After your application data is translated successfully and placed in the WebSphere Data Interchange Transaction Store, the data must be enveloped. Enveloping involves placing an EDI standard header and trailer around your transaction in preparation for sending. The system places your transactions in an intermediate file as they are enveloped. When enveloping is complete, you can send the data to your trading partner.

To send your enveloped data, WebSphere Data Interchange starts a
communications routine that passes the enveloped data in the intermediate file to a
network program. The network program is supplied by you or another provider, such
as a value added network (VAN). The program sends the enveloped data over the
network you defined for communicating with your trading partner.

You can request the translation, envelope, and send functions individually or you
can combine them as needed. For example, you can:
- Translate
- Envelope
- Send
- Translate and Envelope
- Envelope and Send
- Translate, Envelope, and Send

You can execute the envelope and send functions as separate tasks from your
application or you can execute them from within your application. See the
WebSphere Data Interchange Programmer's Reference for additional information
about the WebSphere Data Interchange Utility and the application program
interface. In addition, you can perform the Envelope and the Envelope and Send
functions using the Transaction Store Facility. See *WebSphere Data Interchange for
z/OS Administration Guide* for additional information.

Collecting translated application data in the transaction store and enveloping and
sending the data together in one envelope at a later time may help to reduce
networking costs, instead of sending many envelopes to each trading partner.

Envelope overrides

WebSphere Data Interchange provides you with the capability to override particular
values in the interchange envelope with data provided by your application. A default
envelope is selected if an override is not needed.

Here are some of the ways that envelope overrides are provided:
- One of the steps in the WebSphere Data Interchange trading partner definition is
  defining a trading partner usage. Use the trading partner usage to select the
  correct mapping for translating data. You can specify an envelope override using
  the trading partner usage.

  For outbound processing, the specified envelope profile member can contain
  constant values that override the values generated during the envelope process.
  WebSphere Data Interchange uses the information in the profile member to
  generate the interchange header and trailer segments.

  For inbound processing, WebSphere Data Interchange uses the information in
  the specified envelope profile member to generate the interchange header and
  trailer segments for a functional acknowledgment (if requested).
- Another step in the WebSphere Data Interchange mapping process is the data
  format definition. This definition specifies records and fields in your application
  data file. WebSphere Data Interchange currently provides two definitions for the
  application data (Raw data, and Control and Data (C & D) record format). You
  can find more information about data formats and application data in Part 4,
  "Mapping" on page 149 and the *WebSphere Data Interchange Programmer's
  Reference*.
- WebSphere Data Interchange provides several translation methods to process
  your data. The application programming interface (API) is one method that you

can use to couple your application program with the WebSphere Data Interchange translation process. To use the API, your application must use the predefined control blocks to communicate information to the translation process. One of these control blocks is the translator control block (TRCB). The TRCB allows your application to specify the envelope overrides used to generate the interchange headers and trailers. You can find more information in the *WebSphere Data Interchange Programmer's Reference*.

Receiving EDI data using receive maps

The following steps outline how to receive data from a trading partner.
1. Receive the standard data
2. Deenvelope the standard data
3. Translate the standard data to an application format

Receiving and deenveloping transactions

Periodically, poll the network(s) you use to see if there are transactions waiting to be received. Receiving EDI standard transactions involves invoking WebSphere Data Interchange and requesting the *receive* function. WebSphere Data Interchange starts a communications routine that uses your network program or a program supplied by another provider. If transactions are received, they are placed into an intermediate file. You can specify to receive all transactions, or to receive only transactions from a certain trading partner, depending on the network you use.

After transactions are received, they can be deenveloped. Deenveloping processes the envelope headers and trailers from the EDI standard transactions that are contained in the intermediate file and places the results in the transaction store. Functional acknowledgments are also generated if necessary. When transactions are in the transaction store, you can manage them using the Transaction Store Facility or translate them to an application format and process them using your application. See *WebSphere Data Interchange for z/OS Administration Guide* for more information.

You can execute the receive and deenvelope functions as a separate task from your application or you can execute them from within your application. See the *WebSphere Data Interchange Programmer's Reference* for additional information about the WebSphere Data Interchange Utility and the application program interface. In addition, you can perform the Receive and Deenvelope function using the Transaction Store Facility.

Translating standard data to application format

You can use WebSphere Data Interchange to translate certain transactions that are contained in the transaction store to your application format. Transactions translated to your application format are placed in an application data file. They accumulate and remain there until your application processes them.

You can execute the translate function as a separate task from your application or you can execute translation from within your application. See the *WebSphere Data Interchange Programmer's Reference* for additional information about the WebSphere Data Interchange Utility and the application program interface. In addition, you can perform the translation function using the Transaction Store Facility.

You can request the receive, deenvelope, and translation functions individually or you can combine them as needed. For example, you can:

- Receive
- Deenvelope
- Translate
- Receive and Deenvelope
- Deenvelope and Translate
- Receive, Deenvelope, and Translate

Continuous receive

With WebSphere Data Interchange for z/OS, if you are working in the CICS environment, you can invoke continuous receive functions. These functions allow you to receive data without interruption from one or more trading partners as soon as the data is delivered to your mailbox. The WebSphere Data Interchange continuous receive function receives data as it enters your mailbox, deenvelopes the data, translates the data to a data format, and starts an application response routine to process the data.

Functional acknowledgments

For some of your transactions, you may want to know if your trading partner received, accepted, or rejected your transaction. You can obtain this information if you request a functional acknowledgment and inform your trading partner. WebSphere Data Interchange will track the pending status in the Transaction Store.

When your Trading Partner has requested functional acknowledgements from you, the functional acknowledgement is created upon request during deenveloping and indicates only that a specified set of envelopes is syntactically correct or incorrect. They are placed in the transaction store and are optionally enveloped for sending. A functional acknowledgement does not indicate that the trading partner agrees to the business terms contained in the envelope.

You can use the WebSphere Data Interchange envelope function to:

- Transmit functional acknowledgments to the network and your trading partner.
- Envelope functional acknowledgments at a later time.

If you want immediate turnaround of functional acknowledgments, your task or application can issue the send function. For more information, "Requesting, sending, and receiving functional acknowledgments" on page 143

Update status

If the transactions you send are supposed to generate network acknowledgments, you will need to issue an update status periodically. This can be done using the Transaction Store Facility, the WebSphere Data Interchange Utility, the WebSphere Data Interchange application program interface, or the Continuous Receive Facility. See *WebSphere Data Interchange for z/OS Administration Guide* and *WebSphere Data Interchange Programmer's Reference* for additional information on the WebSphere Data Interchange Utility and the application program interface.

WebSphere Data Interchange Reporting on your system

You can use several reporting and auditing tools within WebSphere Data Interchange to help you manage your system. These tools include online inquiries and displays, formatted reports, data extracts, and exception reports.

Online inquiries using the transaction store facility

You can use the Transaction Store Facility to obtain information and status about transactions and envelopes that you sent to your trading partners or received from your trading partners.

Formatted reports using the WebSphere Data Interchange utility

You can use the WebSphere Data Interchange Utility to specify search criteria and produce the following reports:

- **Activity Summary**

  Creates a summary of your inbound and outbound transaction activity. This summary includes the total number of outgoing and incoming transactions that meets your search criteria. On the outbound side, it categorizes the transactions into the number of translated transactions, enveloped transactions, and sent transactions. On the inbound side, it categorizes the transactions into the number of transactions translated and not translated.

- **Acknowledgment Image**

  Returns functional acknowledgment images for transactions that meet your search criteria.

- **Transaction Details**

  Creates detailed information about individual transactions that meets your search criteria. This includes information about your trading partner, application data, acknowledgments, and networks that relate to the individual transactions selected.

- **Transaction Image**

  Returns transaction images for EDI documents that meet the search criteria.

- **Status Summary**

  Returns a summary of information about individual transactions that meets your search criteria. This information includes:
  - Transaction handle
  - Trading partner nickname
  - Data format ID
  - Transaction status
  - Store status
  - Date enveloped
  - Interchange control number
  - Network status, group control number
  - Functional acknowledgment status

- **Status Summary2**

  Includes all the information from Status Summary plus the application control number and the internal trading partner ID for each transaction.

- **Event Log**

  Returns all entries from the WebSphere Data Interchange Event Log that meet your search criteria.

- **Query**

    Returns a list of transactions, identified by a date and time stamp called the transaction handle, that meets your search criteria.

    For more information, see the *WebSphere Data Interchange Programmer's Reference*.

Transaction and envelope data extracts using the WebSphere Data Interchange utility

Using Transaction and Envelope data extracts, you can retrieve information from the WebSphere Data Interchange Transaction Store and format the information to meet your business needs. You can use a Transaction data extract to determine what documents have not been functionally acknowledged by your trading partners. Use the Envelope data extract to view reports about which applications generated transactions to reconcile system rules.

For more information, see the *WebSphere Data Interchange Programmer's Reference*.

Management reporting data extracts

Management Reporting data extracts include information about trading partner profiles and capabilities, data maps and their rules, network activity, and transaction statistics.

Use Management Reporting data extracts to answer questions, such as how many purchase orders you sent to each of your trading partners within a specific month, or which trading partners you are trading purchase orders with at the X12 V4R1 level. WebSphere Data Interchange provides the following data extracts to gather this information:

- **Trading Partner Profile Data Extract**

    Provides complete identification information about your trading partners, such as company name address, contacts, telephone number, nickname, network name, interchange ID, account ID, and user ID. Additionally, the date of the last transmission of data is provided including the interchange, group, and transaction control numbers.

- **Trading Partner Capability Data Extract**

    Provides a subset of trading partner identification information and detailed information regarding the cumulative transactions exchanged with a trading partner, such as mapping direction, standards used, transaction ID, measurement date range, total number of transactions processed, and number of transactions with errors.

- **Network Activity Data Extract**

    Provides a record of WebSphere Data Interchange's network activity. You can use this information to verify the integrity of your EDI process, and to understand and reconcile network charges. This extract provides data about the network ID, name and account number, user ID, direction, charge code, control date, and the total number of interchange envelopes and characters of data sent or received.

- **Transaction Activity Data Extract**

    Provides a record of daily transaction activity for individual trading partners. You can use this information to verify the integrity of your EDI process, and to understand and reconcile network charges. Trading partner data provided includes a subset of trading partner identification information and detailed

information regarding the daily transactions exchanged with a trading partner. Transaction data provided includes mapping direction, standard used, transaction ID, map ID, data format ID, measurement date, total number of transactions processed, and number of transactions that had errors. Outbound functional acknowledgments will not be reported through Transaction Activity Data Extract Reports.

Data extracts are formatted as sequential files that contain fixed length records. For more information about Management Reporting data extracts, see the *WebSphere Data Interchange Programmer's Reference*.

Export/Import

You can move your maps, profiles, data formats, and more from one WebSphere Data Interchange system to another:

- From a test to a production system using export/import
- From a production system to a test system using export/import
- To exchange data with other WebSphere Data Interchange users
- To move data between a WebSphere Data Interchange client and a WebSphere Data Interchange host
- To migrate to a higher release level

Chapter 6. Export and Import

With the export and import functions, you can install EDI standards on WebSphere Data Interchange Client and transfer your work between computers and WebSphere Data Interchange systems. In stand alone mode, you use the export and import functions to move data between WebSphere Data Interchange Host and WebSphere Data Interchange Client.

You can export and import files to exchange profiles with trading partners that use WebSphere Data Interchange or with other users that are not connected to your local-area network. For example, a manufacter can export the mapping of a purchase order to a supplier so the supplier has a basis from which to work. Although the supplier cannot use the existing map, this process will reduce the work required.

About export and import

Export takes a WebSphere Data Interchange item you select from either the WebSphere Data Interchange Client or Host and writes it either to a file or to another WebSphere Data Interchange system.

Import reads export files and inserts the data into a WebSphere Data Interchange system. Export/import files on the PC are identified by the .EIF file extension.

You use the import and export functions when you need to:
- Install EDI standards on WebSphere Data Interchange Client
- Install DTDs for XML data
- Move profiles, EDI standards, data formats, and maps from WebSphere Data Interchange Host to WebSphere Data Interchange Client
- Update WebSphere Data Interchange Host databases with work done on WebSphere Data Interchange Client when using the client in stand alone mode
- Share profiles, data formats, maps, and EDI standards with other WebSphere Data Interchange users

  Note: Export/import is a critical part of all WebSphere Data Interchange Client interactions with WebSphere Data Interchange Host when using the Client in stand alone mode. For information on how export/import works in moving data between WebSphere Data Interchange Host and WebSphere Data Interchange Client, see "Moving to WebSphere Data Interchange Client" on page 22.

Export and import file specifications

The following specifications apply to export and import files:
- If you are moving Host format maps, EDI standards, and data formats from a previous release, refer to the migration information included with the product.
- The field position and length of fixed-format records are subject to change due to design changes within WebSphere Data Interchange between different releases.
- Exporting EDI standards in fixed format is not recommended because of the amount of disk space they can take up. Tagged format is recommended for exporting EDI standards.
- Export only one complete EDI standard per export or import file to keep file sizes and export times manageable.

Exporting

The WebSphere Data Interchange Client export function has two options:

- You can export to a file
- You can export to another WebSphere Data Interchange system, for example, from the Test system to the Production system

Export to a file when you want to move an item.
- To WebSphere Data Interchange Host from WebSphere Data Interchange Client
- To another WebSphere Data Interchange Client user
- From one system to another

Export to another WebSphere Data Interchange system when you want to duplicate items in various WebSphere Data Interchange systems. For example, if you created a trading partner in the Test system and want to copy to the Production system, use the Export to Other System command.

Exporting to an export and import file

When you complete work in stand-alone mode: you export to file; you use an upload utility to upload that file to the host; and then you use the host's import functions to merge that profile into the WebSphere Data Interchange Host database, so that your changes can take effect.

To export to a file:

1. To export an item to file, click it in its list window.
2. Click Export Selected Documents on the tool bar.

   The Select Export File dialog box displays with the File Name field highlighted.
3. Either select or type the name of the file you want to export into. If you type the name, use the .EIF extension. You can also select an existing file you want to export to by selecting the drive and folder where the file is located.

   To see a list of all files in a folder, select "All Files (*.*)" from the drop-down list in the List of Files field.

   Note: You may export as many items as you like to a single file. For example, you can set up a file called TPUPDATE.EIF and export all of your updated Trading Partner profiles to that file.
4. Click Open.

   If you typed in the name of a new file, the Select File Format dialog box displays.

   Note: A DTD will be in .eif format when exporting.

   a. Choose one of the following file record formats:
      - Tagged Format
      - Writes data into a file with tags representing database fields. This format is recommended because empty fields are not referenced in the file; therefore, it creates the smallest files. This format allows you to move data between versions of WebSphere Data Interchange Client.
      - Fixed Format

- Writes data into a file where each field is represented at its full length in the database. If data in the field does not fill up its whole length, blanks are appended to the data. Use this format when it is the only format your application will accept.
- Comma-Separated Values Format
- Writes data into a file in which each field is separated by a comma. Character fields appear in quotes; numeric fields do not. Use this format to load data into spreadsheet and other PC applications.

If you type in or select a file that has had data exported to it before, WebSphere Data Interchange Client exports your data in the format selected when the file was first created. Data is appended to the end of the file.

5. Click OK.
   - If you are exporting any of the profiles listed in Table 12 on page 70, WebSphere Data Interchange Client displays a dialog box asking you which referenced and associated types you want to export with the item you are exporting. Select the types by clicking the check boxes. For explanations of each of these dialog boxes, "Specifying referenced and associated types" on page 72

6. When you have finished filling out the dialog box, click OK.
   - If you are exporting any of the profiles listed in Table 13 on page 70, no dialog box displays before export.

   WebSphere Data Interchange Client exports the item to the specified file and displays an Execution Status window. When WebSphere Data Interchange Client is finished exporting the item, an Export Complete message displays in the Execution Status window.

Exporting to other systems

The Export to Other System command is useful for exporting items that have been developed and tested in the Test system to the Production system, where they can be used to exchange data with trading partners.

To export to other systems:

1. To export an item to another system, select the item and choose Export to Other System from the Actions menu.

   The Select a System dialog box displays with the available systems listed.

2. Select the system to which you want to export your item and click OK.

   A dialog box for your system's client-server middleware package may be displayed.

3. If the dialog box displays, enter your user ID and Password to gain access to the system you have selected and click OK.

   If you are exporting any of the profiles listed in Table 12 on page 70, WebSphere Data Interchange Client displays a dialog box asking you which referenced and associated types you want to send with the item you are exporting. Select the types by clicking the check boxes. For explanations of each of these dialog boxes, "Specifying referenced and associated types" on page 72 of this chapter.

4. When you have finished filling out the dialog box, click OK.

   WebSphere Data Interchange Client exports the selected item. If you are exporting any of the profiles listed in Table 13 on page 70, no dialog box displays before export.

Exporting

> **Note:** When you export an item to a system which already has an item of that type with the same name, WebSphere Data Interchange Client displays a warning dialog box. If you want to replace the existing item with the one you are exporting, click OK. Otherwise, you can cancel your export.

Table 12. Items with Associated or Referenced Types

Item	Page described
Mailbox	72
Network profile	72
Application defaults profile	72
Continuous receive profile	73
Network security profile	74
Trading partner profile	74
Data format dictionary	75
Data format	75
EDI standard dictionary	75
EDI standard transaction	75
Envelope standard	75
Mapping	76
Control strings	76

Table 13. Items with no Associated or Referenced Types

Item
Activity log profile
CICS performance profile
Code list
Contacts
E, I, T, U, and X envelope profile
Envelope control string
Forward translation table
Global variables
Language profile
WebSphere MQ queues
Network commands profile
Reverse translation table
User exits profile
XML dictionary
XML DTDs

Importing from an export and import file

The import function allows you to use an item received from another WebSphere Data Interchange user.

To import an item:

1. Select Open Import File from the File menu.

   The Select Import File dialog box displays with the File Name field highlighted.

2. You can select an existing file from which you want to import by selecting the drive, folder, and file.

   To see a list of all files in a folder, select "All Files (*.*)" from the drop-down list in the List of Files field.

3. Click Open.

   The tree view of the export/import file you selected displays with folders below representing each type of item that you have exported to that file. To see the contents of a folder, click the plus sign to the left of the folder. The contents of a folder are shown as documents beneath that folder.

4. Double-click the document you want to import.

5. Click Import on the tool bar.

   The Select a System dialog box displays with the available systems listed.

6. Select the system to which you want to import your item and click OK.

   A dialog box for your system's client-server middleware package may display for you to log in to the system.

7. If a dialog box displays, enter your user ID and password to gain access to the system you have selected and click OK.

   WebSphere Data Interchange Client imports the selected items into the specified system and displays the Execution Status window so that you can monitor the progress of the import. When WebSphere Data Interchange Client is finished importing the item, an Import Complete message displays in the Execution Status window.

   Note: When you import an item to a system that already has an item of that type with the same name, WebSphere Data Interchange Client displays a warning dialog box. If you want to replace the existing file with the one you are importing, click OK. Otherwise, you can cancel your import.

Importing a DTD file

A DTD file must be obtained and then imported into WebSphere Data Interchange.

To import a DTD:

1. Select Open Import File from the File menu.

   The Select Import File dialog box displays with the File Name field highlighted.

2. Either select or type the name of the file from which you want to import. If you type the file name, you should type the .DTD file extension. You can also select an existing file from which you want to import by selecting the drive and folder where the file is located.

   To see a list of all DTD files in a folder, select "XML DTD File (*.dtd)" from the drop-down list in the List of Files field.

3. Click Open.

   The Import XML DTD window opens with the name of the DTD file shown in the DTD Name field.

4. Select a dictionary from the drop-down list in the Dictionary Name field. Selecting a dictionary is required.

5. You can enter the root element in the Root Element Name field.

6. You can enter a description of the DTD file in the Description field.
7. Click the Import button when you complete the field entries.

Specifying referenced and associated types

Many WebSphere Data Interchange items do not work in isolation but in relation to other items. Maps, for example, work in relation to EDI standards and usages or map rules, which work in relation to Trading Partner profiles. When you export such items, you may need to export the items to which they are related. The Export and Import dialog boxes allow you to select which related types you want to export along with the primary item.

There are two types of relationships between WebSphere Data Interchange items:
- Referenced

  Referenced types are items that are referred to by the item you are exporting.
- Associated

  Associated types, on the other hand, refer to the item you are exporting.

  Attention: When importing associated and referenced types, the imported items replace existing ones with the same name without warning.

Procedures for each item with associated or referenced types follow.

Mailbox

To complete the Export Mailbox dialog box:
1. Click the Network Profile check box if you want to include the Network profile referenced in the Mailbox you are exporting.
2. Click the Network Commands check box if you want to include the Network Commands referenced in the Network profile you are exporting with the Mailbox profile.
3. Click the Remote Status Program Profile check box if you want to include the Remote Status User Exit item referenced in the Mailbox profile you are exporting.
4. When you have finished selecting the referenced types you wish to export, click OK.

   WebSphere Data Interchange Client begins to export the item.

Network profile

To complete the Export With Network Profile dialog box:
1. Click the Network Commands check box if you want to include the Network command referenced in the Network profile you are exporting.
2. When you have finished selecting the referenced types you wish to export, click OK.

   WebSphere Data Interchange Client begins to export the item.

Application defaults profile

To complete the Export With Application Defaults Profile dialog box:
1. Click the Activity Log Profile check box if you want to include the Activity Log referenced in the Application Defaults profile you are exporting.

2. Click the Monitor Exits check box if you want to include the Monitor User Exit referenced in the monitor user exit you are exporting.
3. When you have finished selecting the referenced types you wish to export, click OK.

   WebSphere Data Interchange Client begins to export the item.

Continuous receive profile

To complete the Export with Continuous Receive Profile dialog box:

1. Click the Activity Log Profile check box if you want to include the Activity Log referenced in the Continuous Receive profile you are exporting.
2. Click the Mailbox Profile check box if you want to include the Mailbox profile referenced in the Continuous Receive profile you are exporting.
   a. Click the Network Profile or Remote Status Program check boxes if you want to include Network profile or Remote Status Program user exit types referenced in the Mailbox profile you are exporting with the Continuous Receive profile.
   b. Click the Network Commands check box if you want to include the Network Commands referenced by the Network profile you are exporting with the Continuous Receive profile.
3. Click the Trading Partner Profile check box if you want to include the Trading Partner referenced in the Continuous Receive profile you are exporting.
   a. Click the Trading Partner Contacts, Network Profile or Network Security Profile check boxes if you want to include the appropriate items that are referenced in the Trading Partner profile you are exporting with the Continuous Receive profile.
   b. Click the Network Commands check box to include the Network Commands referenced in the Network profile you are exporting with the Continuous Receive profile.
   c. Click the Network Security Exits check box if you want to include the Security User Exits referenced in the Network Security profile you are exporting with the Continuous Receive profile. Security User Exits include Authentication, Encryption, Filtering, and Compression.
   d. If you include the referenced Trading Partner Profile, you may choose to include its associated types, Receive Usages, Send Usages, or Rules.
   e. Click the option button corresponding to the usages or map rules you want to include.
      • To export all usages or rules, click All.
      • To export no usages or rules, click None.
      • To select which usages or rules you want to export, click Select.

        During the export, the appropriate usages or rules dialog box displays.
   f. Select the usages or rules you want to include with the export and click OK.
4. Click the Activity Log Profile check box if you want to include the Activity Log profile referenced in the Continuous Receive profile you are exporting.
5. Click the Mailbox Profile check box if you want to include the Mailbox profile referenced in the Continuous Receive profile you are exporting.
   a. Click the Network Profile or Remote Status Program check boxes if you want to include Network profile or Remote Status Program user exit types referenced in the Mailbox profile you are exporting with the Continuous Receive profile.

Continuous receive profile

  b. Click the Network Commands check box if you want to include the Network Commands referenced by the Network profile you are exporting with the Continuous Receive profile.

 6. When you have finished selecting the referenced and associated types you wish to export, click OK.

  WebSphere Data Interchange Client begins to export the item.

Network security profile

To complete the Export with Network Security Profile dialog box:
1. Click the Network Security Exits check box if you want to include the Security user exits (Authentication, Encryption, Filtering, and Compression) referenced in the Network Security profile you are exporting.

2. When you have finished selecting the referenced type you wish to export, click OK.

  WebSphere Data Interchange Client begins to export the item.

Trading partner profile

To complete the export Trading Partner Profile dialog box:
1. Click the Trading Partner Contacts check box if you want to include the Trading Partner Contacts referenced in the Trading Partner profile you are exporting.

2. Click the Network Profile check box if you want to include the Network profile referenced in the Trading Partner Profile you are exporting.

  a. Click the Network Commands check box if you want to include the Network Commands profile referenced in the Network Profile you are exporting with the Trading Partner profile.

3. Click the Network Security Profile check box if you want to include the Network Security profile referenced in the Trading Partner Profile you are exporting.

  a. Click the Network Security Exits check box if you want to include the Network Security User Exits profile referenced in the Network Profile you are exporting with the Trading Partner profile.

4. Click the Control Numbers check box if you want to include the control number pairings.

5. You may choose to include associated types, Receive Usages, Send Usages, and Rules with the Trading Partner profile you are exporting.

  a. Click the option button corresponding to the usages or rules you want to export.

   • To export all usages or rules, click All.

   • To export no usages or rules, click None.

   • To select which usages or rules you want to export, click Select.

    During the export, the appropriate usages or rules dialog box displays.

  b. Select the usages or rules you want to export and click OK.

6. You may choose to include Usages or Rules referenced types, Translation Exits or Envelope Profiles.

7. When you have finished selecting the referenced and associated types you wish to export, click OK.

  WebSphere Data Interchange Client begins to export the item.

Data format dictionary

To complete the Export with Data Format Dictionary dialog box:

1. Click the Code Lists check box if you want to include the Code List referenced in the Data Format Dictionary you are exporting.

2. When you have finished selecting the referenced type you wish to export, click OK.

   WebSphere Data Interchange Client begins to export the item.

Data format

To complete the Export with Data Format dialog box:

1. Click the Code Lists check box if you want to include the Code List referenced in the Data Format you are exporting.

2. When you have finished selecting the referenced type you wish to export, click OK.

   WebSphere Data Interchange Client begins to export the item.

EDI standard dictionary

To complete the Export with EDI Standard Dictionary dialog box:

1. Click the appropriate check box to include that referenced item in the EDI Standard Dictionary you are exporting.

2. When you have finished selecting the referenced types you wish to export, click OK.

   WebSphere Data Interchange Client begins to export the item.

EDI standard transaction

To complete the Export with EDI Standard Transaction dialog box:

1. Click the appropriate check box to include that referenced item in the EDI Standard Transaction you are exporting.

2. When you have finished selecting the referenced types you wish to export, click OK.

   WebSphere Data Interchange Client begins to export the item.

Envelope standard

To complete the Export with Envelope Standard dialog box:

1. Click the appropriate check box to include that referenced item in the envelope standard you are exporting.

2. When you have finished selecting the referenced types you wish to export, click OK.

   WebSphere Data Interchange Client begins to export the item.

Mapping

To complete the export Mapping dialog box:

1. Click the check boxes of the following referenced types if you want to include them with the map you are exporting: Source and Target Document Definitions, Global Variables, Translation Tables, Field Exits, and Code Lists.

2. You can choose to include the associated types, control string and usages or rules, with the map you are exporting.

   a. If you want to include the control string associated with the map you are exporting, click the Control String check box.

   b. If you want to include the usages or rules associated with the map you are exporting, click one of the three option:

   c. Click the option button corresponding to the usages or rules you want to export.

      • To export all usages or rules, click All.

      • To export no usages or rules, click None.

      • To select which usages or rules you want to export, click Select.

         During the export, the appropriate export usages or rules dialog box displays.

   d. Select the usages or rules you want to send with the export and click OK.

   e. If you include usages or rules associated with your map, you may choose to send Trading Partner Profiles, Translation Exit Routines, and Envelope Profiles referenced in the Usages/Rules by clicking the respective check boxes.

   f. If you include the Trading Partner profile referenced in your map, you may choose to send its referenced types, Trading Partner Contacts, Network Profiles, and Network Security Profiles, by clicking the Trading Partner Contacts, Network Profiles, and Network Security Profiles check boxes, respectively. You may choose to send its associated types by clicking the Control Numbers check box.

   g. If you include the Network Profile referenced in your map, you may choose to send its referenced type, Network Commands, by clicking the Network Commands check box.

   h. If you include the Network Security Profile referenced in your map, you may choose to send its referenced type, Network Security User Exits, by clicking the Network Security Exits check box.

3. When you have finished selecting the referenced and associated types you wish to export, click OK.

   WebSphere Data Interchange Client begins to export the item.

Control Strings

Note: Control strings have the following restrictions:

   • They can be imported.

   • They cannot be exported in fixed or comma-delimited formats.

To complete the Export with Control Strings dialog box:

1. Click the check boxes of the following referenced types if you want to include them with the Control String you are exporting: Global Variables, Translation Tables, Field Exits, Code Lists, Maps, and User Exits.

2. You may choose to include the Usages/Rules associated with the Control String you are exporting.

   a. If you want to include usages or rules with the Control String you are exporting, click one of the three option buttons:
      - To include all usages or rules, click All.
      - To include no usages or rules, click None.
      - To select which usages or rules you want to export, click Select.

        During the export, the appropriate usages or rules dialog box displays.

        Select the usages or rules you want to export with the map control string, and click OK.

   b. If you include the usages or rules associated with your Control Strings export, you may choose to send their referenced types, Trading Partner Profiles, Translation User Exit (Pre-Translation and Post-Translation User Exits), Envelope Profiles by clicking the appropriate check boxes.

   c. If you include the Trading Partner profile referenced in the Control String you are exporting, you may choose to send its referenced types, Trading Partner Contacts, Network Profiles, and Network Security Profiles, by clicking the appropriate check boxes. You may choose to send its associated types by clicking the Control Numbers check box.

   d. If you include the Network Profiles referenced in the Trading Partner profile you are exporting, you may choose to send its referenced type, Network Commands, by clicking the Network Commands check box.

   e. If you include the Network Security Profiles referenced in the Trading Partner profile you are exporting, you may choose to send its referenced type by clicking the Network Security Profiles check box.

3. When you have finished selecting the referenced and associated types you wish to export, click OK.

   WebSphere Data Interchange Client begins to export the item.

Control strings

Part 2. Setup

79

Chapter 7. Activity log profiles

The Activity Log profile allows you to set up message files that instruct WebSphere Data Interchange to log a record of host activities, including:

- Comments on the status of an event, such as whether it is queued, sending, receiving, delivered, or completed
- Notes™ showing when a user gains access to a profile using the WebSphere Data Interchange Host interface
- Detailed descriptions of program and database errors

By default, WebSphere Data Interchange logs all activity messages in a single Activity Log profile, called EDIFFS. You need not change that profile or add another. It is more convenient to sort out activity messages, however, if you create separate Activity Log profiles for different purposes.

About activity logs

Activity log profiles allow WebSphere Data Interchange to log messages about WebSphere Data Interchange activities in a file. This section provides an overview of the purpose of activity log profiles and how you set them up.

Purpose

All WebSphere Data Interchange Host messages are placed in a single DB2 table named EDILOG. This table is logically partitioned into separate logs by the column ELAPPLID, which is referred to in WebSphere Data Interchange as the log name. By default, WebSphere Data Interchange Utility messages are inserted using the log name specified in the EDIFFS Activity Log profile. WebSphere Data Interchange Facility messages are inserted using the log name specified in the EDIIMP Activity Log profile. Although you need not create any other Activity Log profiles, you may find that it is easier to sort messages if you store them under different log names created for different purposes.

For example, most companies run at least two different systems, Test and Production. They use the Test system while setting up a trading partner in order to make sure that such things as maps and profiles are working correctly. When they are satisfied that their WebSphere Data Interchange system correctly processes the trading partner's data, they move that trading partner to the Production system.

If all activity messages that WebSphere Data Interchange logged were stored based on the default Activity Log profile, it would be difficult to separate messages related to the Test system from messages related to the Production system. Consequently, many companies at least create separate Activity Log profiles for their Test and Production systems. That allows them to track errors more easily when testing a new trading partner.

Another reason companies may use separate Activity Logs is to separate messages by application. For example, Purchasing may want to receive a record of all activity from its application. To provide that record, you would set up an Activity Log profile for the purchasing application. All activity bound to and from the purchasing system which WebSphere Data Interchange logged would then be sorted to a file identified by the Activity Log profile for the purchasing application.

Setup overview

You set up and maintain Activity Log profiles through the Activity Log List window, which you access by clicking Setup on the WebSphere Data Interchange Client Navigator bar. The Setup window, which contains tabs for WebSphere Data Interchange Client's setup profiles, displays. Click the Activity Log tab, and the Activity Log List window displays.

This window displays a list of existing Activity Log profiles. Each row contains information about an Activity Log profile, each column contains data stored in the profile. Information in the columns displays in fields, drop-down lists, and check boxes in the Activity Log Editor window. The profile list window, however, also contains the date, time and user ID of the last update.

To display additional columns, click the scroll bar on the bottom of the screen to scroll to the right or left. To alter the columns that display on the screen, or to change which query is executed to produce the list, click Modify Window Properties (...). To create new queries, refer to "Creating a query" on page 268

To view a profile or to add or change the information in these fields, double-click the row of the profile you wish to work with. The Activity Log Editor window displays, with the General tab in front.

The Activity Log Editor window contains two tabs: General and Comments. Use the General tab to enter and change information contained in the Activity Log profile. Use the Comments tab to type any comments you wish about the selected Activity Log profile.

Following are detailed procedures for creating new Activity Log profiles. For information on viewing, copying, editing, renaming, deleting, and printing profiles, "Performing common file management tasks" on page 45 For information on exporting profiles, "Exporting" on page 68

Creating Activity Log profiles

The first step in creating Activity Log profiles is to decide whether you need to create separate files to log WebSphere Data Interchange activity messages for particular systems or applications. Most companies at least create separate activity logs for their test and production systems. Some also create separate logs for different applications, such as purchasing or accounts receivable.

Create a new Activity Log profile to your WebSphere Data Interchange installation when you want to separate WebSphere Data Interchange messages by WebSphere Data Interchange system, business application, or other criteria.

To create an Activity Log profile:

1. Click Setup on the WebSphere Data Interchange Client Navigator bar.

   The Setup window displays.
2. Click the Activity Log tab.

   A list of the existing Activity Log profiles displays.
3. Click New on the tool bar.

   The Activity Log profile Editor window displays with the General tab open and all fields blank.

4. Complete the fields on the General tab. Required fields are preceded by a blue asterisk.

   Click for field descriptions.

5. Click the Comments tab and type any comments you have about the Activity Log profile into the Comments field.

6. Select Save from the File menu to save the profile.

Creating activity log profiles

Chapter 8. Application defaults profiles

An application defaults profile allows you to identify your business applications, such as purchasing and accounts receivable, to WebSphere Data Interchange and set specific WebSphere Data Interchange processing defaults for an application. You need not create an application defaults profile for each application; you can use one profile for all applications, as long as you want WebSphere Data Interchange to translate messages for each application in the same manner.

Processing defaults specified in an application defaults profile include:
- The name of the activity Log used to store processing messages
- Whether you log WebSphere Data Interchange transaction data to the Transaction Store
- Whether to gather management reporting statistics for an application

About application defaults

The application defaults profile identifies your business applications to WebSphere Data Interchange and controls certain processing features. An application defaults profile mainly determines whether copies of transactions are stored in WebSphere Data Interchange's Transaction Store database and the log files in which processing messages are stored.

When an application invokes WebSphere Data Interchange, it provides an application ID. WebSphere Data Interchange then searches for an application defaults profile to match the application ID. If WebSphere Data Interchange cannot find the matching application defaults profile, it searches for an activity log profile set up for the application. If it cannot find a specific activity log, it uses the WebSphere Data Interchange default activity log, EDIFFS.

Should you decide, for example, not to send images of functional acknowledgments for the invoicing application to the Transaction Store database, you would set up an application defaults profile for your company's invoicing system with such instructions.

For more information on the Transaction Store, see the *WebSphere Data Interchange for z/OS Administration Guide*. For more information on processing messages, see Chapter 7, "Activity log profiles" on page 81

Setup overview

You set up and maintain application defaults profiles through the application defaults list window, which you access by clicking Setup on the WebSphere Data Interchange Client Navigator bar. The Setup window, which contains tabs for WebSphere Data Interchange Client's setup profiles, displays. Click the application defaults tab, and the application defaults list window displays.

This window displays a list of existing application defaults. Each row contains information about an application defaults profile; each column contains data stored in that profile. Information in the columns displays in fields, drop-down lists, and check boxes in the application defaults editor window. The profile list window, however, also contains the date, time, and user ID of the last update.

Application defaults profiles

To display additional columns, click the scroll bar on the bottom of the screen to scroll to the right or left. To alter the columns that display on the screen, or to change which query is executed to produce the list, click Modify Window Properties (...). To create new queries, refer to "Creating a query" on page 268

To view a profile or to add or change the information in these fields, double-click the profile you wish to work with. The application defaults editor window displays, with the General tab in front.

The application defaults editor window contains two tabs: General and Comments. Use the General tab to enter and change information contained in the application defaults profile. Use the Comments tab to type any comments you wish about the selected application defaults profile.

Following are detailed procedures for creating new application defaults profiles. For information on viewing, copying, editing, renaming, deleting, and printing profiles, "Performing common file management tasks" on page 45 For information on exporting profiles, "Exporting" on page 68

Creating application defaults profiles

The first step in creating application defaults profiles is to identify the business applications you are linking to WebSphere Data Interchange. Then you must decide how you want WebSphere Data Interchange to handle the logging of each application's EDI transactions and WebSphere Data Interchange activity logs.

Create a new application defaults profile when you want to treat a particular business application's processing defaults differently from either the WebSphere Data Interchange default values or other applications you have previously set up.

To create an application defaults profile:

1. Click Setup on the WebSphere Data Interchange Client Navigator bar.
   The Setup window displays.
2. Click the application defaults tab.
   A list of the existing application defaults profiles displays.
3. Click New on the tool bar.
   The application defaults Editor window displays with the General tab open and all fields blank.
4. Complete the fields on the General tab. Required fields are preceded by a blue asterisk.

   Click [**?**]

   for field descriptions.
5. Click the Comments tab and type any comments you have about the application defaults profile into the Comments field.
6. Click Save on the tool bar to save the profile.

Chapter 9. Continuous receive profiles

A continuous receive profile allows you to initiate event-driven processes using WebSphere Data Interchange for z/OS and Expedite/CICS in the CICS environment. (Expedite/CICS is the communication package for connecting to IBM Global Services.) Event-driven processes means that the receipt of an message from a trading partner triggers the translation process. Under typical processing, your system polls your network mailbox periodically, and any messages are processed in batches.

Through continuous receive, you can automate the process of receiving and processing files sent to your mailbox by trading partners. WebSphere Data Interchange Client allows you to set up a continuous receive profile to:

- Receive and deenvelope standard data
- Translate the standard data to a business application's format
- Automatically start a response application, which receives the translated business application data into a temporary storage queue
- Automatically receive network acknowledgments

> **Note:** The continuous receive feature is used only with receive maps. It can not be used with send maps or data transformation maps.

About continuous receive profiles

continuous receive profiles allow you to automate receiving and processing of messages using WebSphere Data Interchange/MVS™-CICS. In other words, continuous receive profiles allow you to initiate event-driven processes, in which the receipt of an message from a trading partner initiates the translation process, as opposed to periodic polling of a network mailbox to check for messages.

The continuous receive process can also be used to perform event-driven EDI using WebSphere Data Interchange/MVS-CICS and WebSphere MQ message queues. For more information, "Using WebSphere MQ with continuous receive" on page 89

You need to add a continuous receive profile for each unique continuous receive session you want to run. For example, you can add a continuous receive profile for each network mailbox, or one for each transaction type, or one for a specific trading partner. You can also create continuous receive profiles to define different ways of processing transactions, including:

- Translating the data and delivering it to the receiving business application
- Saving untranslated data in the Transaction Store database
- Providing the output in C (control) and D (data) records or as raw data
- Starting a response transaction or application after WebSphere Data Interchange has finished its processing
- Automatically receiving network acknowledgments

For information about starting or stopping continuous receive requests, or for information about continuous receive cleanup, see the *WebSphere Data Interchange Programmer's Reference*.

Setup overview

You set up and maintain continuous receive profiles through the continuous receive List window, which you access by clicking Setup on the WebSphere Data Interchange Client Navigator bar. The Setup window, which contains tabs for WebSphere Data Interchange Client's setup profiles, displays. Click the continuous receive tab, and the continuous receive List window displays.

This window displays a list of existing continuous receive profiles. Each row contains information about a continuous receive profile; each column contains data stored in the profile. Information in the columns displays in fields, drop-down lists,

The continuous receive Editor window displays with the General tab open and all fields blank.

4. Complete the fields on the General tab. Required fields are preceded by a blue asterisk.

   Click ?

   for field descriptions.

5. Click the Comments tab and type any comments you have about the continuous receive profile into the Comments field.

6. Click Save on the tool bar to save the profile.

Using WebSphere MQ with continuous receive

The continuous receive process can be used to perform event-driven EDI using WebSphere MQ message queues. To do this, perform the following steps:

1. Your CICS and WebSphere MQ administrators must enable the CICS region for WebSphere MQ support. As part of this process, they must define an WebSphere MQ trigger event queue, which the WebSphere MQ supplied transaction CKTI monitors within the CICS region. For example, assume CICS1.TRIGGER is the name of this queue.

2. Your WebSphere MQ administrator must define an WebSphere MQ process via the DEFINE PROCESS command to provide WebSphere MQ with the information about the WebSphere Data Interchange transaction EDIQ. A sample of the DEFINE command follows:

   DEFINE PROCESS(EDIPROCESS) APPLICID(EDIQ) APPLTYPE(CICS)

3. Using the WebSphere MQ DEFINE QLOCAL command, your WebSphere MQ administrator must define the WebSphere MQ message queue that will receive envelopes. A sample of the command follows:

   DEFINE QLOCAL(EDIRECEIVE) INITQ(CICS1.TRIGGER) PROCESS(EDIPROCESS)

   TRIGGER TRIGTYPE(FIRST) TRIGDATA('CRPROF=MQ1 MQPROF=RECV1')

   The TRIGDATA field is critical. Within this field are two keyword-value combinations. The CRPROF keyword identifies the name of the WebSphere Data Interchange continuous receive profile to use when data is received on the message queue. The MQPROF keyword identifies a WebSphere Data Interchange WebSphere MQ Queue profile associated with this message queue. Both of these keyword-value combinations are mandatory. The order in which the two keyword-value combinations are specified is not important.

   In this example, only one WebSphere MQ message queue will be used for event driven EDI. You can define any number of queues in your environment. If WebSphere Data Interchange and WebSphere MQ information is set up correctly, WebSphere Data Interchange will process any number of queues.

   The WebSphere MQ definitions are now complete.

4. Within WebSphere Data Interchange, you must define an WebSphere MQ Queue profile. The WebSphere MQ Queue profile associates the physical WebSphere MQ message queue name with a logical name used within WebSphere Data Interchange. The WebSphere MQ Queue profile also includes information that is used when processing the message queue. In the example, the name of the WebSphere MQ Queue profile is RECV1. When creating RECV1, the full queue name will be set to EDIRECEIVE.

   For more information, seeChapter 14, "WebSphere MQ queue profiles" on page 101.

Continuous receive profile with WebSphere MQ

5. You must also define a continuous receive profile. In this example, the profile name is MQ1. The TP Nickname, Mailbox ID, MSG User Class, and Network Acks Only fields should be left blank. They are ignored when the continuous receive profile is used for WebSphere MQ processing.

WebSphere Data Interchange automatically processes all data written to the WebSphere MQ message queue as soon as WebSphere MQ dispatches the trigger event messages.

Chapter 10. CICS performance profiles

The CICS performance profile specifies whether WebSphere Data Interchange should use the host system's advanced CICS performance capabilities. WebSphere Data Interchange Client allows you to enter the information that WebSphere Data Interchange needs to increase performance under CICS.

About CICS performance profiles

CICS performance profiles allow you to define CICS performance characteristics for specific CICS systems in your WebSphere Data Interchange host environment. This section provides an overview of the purpose of CICS Performance profiles and how you set them up.

Purpose

As an online environment, CICS is designed to execute commands as quickly as possible. WebSphere Data Interchange Client allows you to set up CICS performance profiles that provide information required by WebSphere Data Interchange to optimize performance in a CICS environment.

When you have set up the profile to make CICS performance enhancements active, WebSphere Data Interchange uses core memory to hold data. When the translator requires that data, it can retrieve it from core memory, rather than the database, which increases system performance.

Setup overview

You set up and maintain CICS Performance profiles through the CICS Performance List window, which you access by clicking Setup on the WebSphere Data Interchange Client Navigator bar. The Setup window, which contains tabs for WebSphere Data Interchange Client's setup profiles, displays. Click the CICS Performance tab, and the CICS Performance List window displays.

This window displays a list of existing CICS performance profiles. Each row contains information about a CICS Performance profile; each column contains data stored in the profile. Information in the columns displays in fields, drop-down lists, and check boxes in the CICS performance editor window. The profile list window, however, also contains the date, time, and user ID of the last update.

To display additional columns, click the scroll bar on the bottom of the screen to scroll to the right or left. To alter the columns that display on the screen, or to change which query is executed to produce the list, click Modify Window Properties (...). To create new queries, refer to "Creating a query" on page 268

To view a profile or to add or change the information in these fields, double-click the row of the profile you wish to work with. The CICS Performance Editor window displays, with the General tab in front.

The CICS performance editor window contains two tabs: General and Comments. Use the General tab to enter and change information contained in the CICS performance profile. Use the Comments tab to type any comments you wish about the selected CICS Performance profile.

Creating CICS performance profiles

This section contains procedures for creating new CICS performance profiles. For information on viewing, copying, editing, renaming, deleting, and printing profiles, see "Performing common file management tasks" on page 45. For information on exporting profiles, see "Exporting" on page 68

The first step in creating CICS performance profiles is determining the name of the CICS performance regions that will use the WebSphere Data Interchange performance feature.

To create a CICS Performance profile:

1. Click Setup on the WebSphere Data Interchange Client Navigator bar.

    The Setup window displays.
2. Click the CICS Performance tab.

    A list of the existing CICS Performance profiles displays.
3. Click New on the tool bar.

    The CICS Performance Editor window displays with the General tab open.
4. Complete the fields on the General tab. Required fields are preceded by a blue asterisk.

    Click [?]

    for field descriptions.
5. Click the Comments tab and type any comments you have about the CICS Performance profile into the Comments field.
6. Click Save on the tool bar to save the profile.

Chapter 11. Envelope profiles

The envelope profiles have one field for each element in the envelope standard. The profiles provide literal or constant data for building header or trailer segments for transaction sets, messages, functional groups, and interchanges. You supply only the values that need to be populated and for which a value is not provided by another source. See the individual profile for the envelope standard you are using.

About envelope profiles

Each envelope profile has a name and a description. The remaining fields represent the data elements in the envelope standard. The field names are designed to make cross referencing easy. For example, the field UNB03 is the third data element in elements You su

Creating envelope profiles

2. Click the tab for the envelope profile (E, I, T, U, or X) that you want to create.

   A list of the existing profiles for that specific envelope profile type displays.

3. Click New on the tool bar.

   The Envelope Profile Editor window displays with the General tab open.

4. Complete the fields on the General tab, which are the same for all envelope profiles. The required field is preceded by a blue asterisk.

   Click ![?] for field descriptions.

5. To complete the specific envelope profile you are working on, refer to the section listed below for the type of profile you are creating:

   "UN/EDIFACT (E) envelope profile"

   "ICS (I) envelope profile"

   "UN/TDI (T) envelope profile" on page 95

   "UCS (U) envelope profile" on page 95

   "X12 (X) envelope profile" on page 95

   Note: Each envelope profile window has a comments tab.

UN/EDIFACT (E) envelope profile

After completing the General tab:

1. Click each of the tabs, in turn, completing the fields as needed.

   Tabs

   Interchange Header (UNB)

   Interchange Trailer (UNZ)

   Functional Group Header (UNG)

   Functional Group Trailer (UNE)

   Message Header (UNH)

   Message Trailer (UNT)

   Click ![?] for field names and descriptions.

2. Click Save on the tool bar to save the profile.

ICS (I) envelope profile

After completing the General tab:

1. Click each of the tabs, in turn, completing the fields as needed.

   Tabs

   Interchange Header (ICS)

   Interchange Trailer (ICE)

   Functional Group Header (GS)

   Functional Group Trailer (GE)

   Transaction Set Header (ST)

   Transaction Set Trailer (SE)

   Click ![?] for field names and descriptions.

2. Click Save on the tool bar to save the profile.

UN/TDI (T) envelope profile

After completing the General tab:

1. Click each of the tabs, in turn, completing the fields as needed.

   Tabs

   Interchange Header (STX)

   Interchange Trailer (END)

   Message Header (MHD)

   Message Trailer (MTR)

   Click ![?] for field names and descriptions.

2. Click Save on the tool bar to save the profile.

UCS (U) envelope profile

After completing the General tab:

1. Click each of the tabs, in turn, completing the fields as needed.

   Tabs

   Interchange Header (BG)

   Interchange Trailer (EG)

   Functional Group Header (GS)

   Functional Group Trailer (GE)

   Transaction Set Header (ST)

   Transaction Set Trailer (SE)

   Click ![?] for field names and descriptions.

2. Click Save on the tool bar to save the profile.

X12 (X) envelope profile

After completing the General tab:

1. Click each of the tabs, in turn, completing the fields as needed.

   Tabs

   Interchange Header (ISA)

   Interchange Trailer (IEA)

   Functional Group Header (GS)

   Functional Group Trailer (GE)

   Transaction Set Header (ST)

   Transaction Set Trailer (SE)

   Click ![?] for field names and descriptions.

2. Click Save on the tool bar to save the profile.

X12 (X) envelope profile

Chapter 12. Language profiles

The language profile is used to handle language and character set issues on the server which is used to perform translations. It does not effect the PC on which WebSphere Data Interchange Client is running. A language profile contains language variables, such as the formats you use for the date and time. Initially, it contains a member for the language version you install. For example, ENU is the profile for US English. You can add additional members or tailor some parts of the member that WebSphere Data Interchange supplies.

About language profiles

language profiles are used when you need unique language variables for translation. The language profile is used by the WebSphere Data Interchange translator. It is not used in WebSphere Data Interchange Client.

Setup overview

You set up and maintain language profiles through the language profiles list window, which you access by clicking Setup on the WebSphere Data Interchange Client Navigator bar. The Setup window, which contains WebSphere Data Interchange Client's setup profiles, displays. Click the language profiles tab, and the language profile list window displays.

This window displays a list of existing language profiles. Each row contains information about a language profile; each column contains data stored in the profile. Information in the columns displays in fields, drop-down lists, and check boxes in the language profile editor window. The profile list window, however, also contains the date, time, and user ID of the last update.

To display additional columns, click the scroll bar on the bottom of the screen to scroll to the right or left. To alter the columns that display on the screen, or to change which query is executed to produce the list, click Modify Window Properties (...). To create new queries, refer to "Creating a query" on page 268

To view a profile or to add or change the information in these fields, double-click the row of the profile you wish to work with. The language profile editor window displays, with the General tab in front.

The language Profile editor window contains two tabs: General and Comments. Use the General tab to enter and change information contained in the language profile. Use the Comments tab to type any comments you wish about the selected language profile.

Following are detailed procedures for creating new language profiles. For information on viewing, copying, editing, renaming, deleting, and printing profiles, see "Performing common file management tasks" on page 45 For information on exporting profiles, see "Exporting" on page 68

97

Chapter 13. Mailbox profiles

A mailbox profile allows you to identify to WebSphere Data Interchange the groups in your organization that receive documents. Mailbox profiles, therefore, are set up for mailbox users, not the network mailboxes themselves. More than one user can use the same network mailbox, but each user must have his own mailbox profile.

About 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.

A Mailbox profile may identify the owner of a network mailbox, but it does not only identify mailbox owners. Many individuals or groups can use the same network mailbox. In order to process documents to meet the requirements of a particular mailbox user, WebSphere Data Interchange allows you to set up a custom mailbox profile for each user.

Setup overview

You set up and maintain mailbox profiles through the Mailbox List window, which you access by clicking Setup on the WebSphere Data Interchange Client Navigator bar. The Setup window, which contains WebSphere Data Interchange Client's setup profiles, displays. Click the Mailboxes tab, and the Mailbox List window displays.

This window displays a list of existing mailbox profiles. Each row contains information about a mailbox profile; each column contains data stored in that profile. Information in the columns displays in fields, drop-down lists, and check boxes in the Mailbox Editor window. The profile list window, however, also contains the date, time, and user ID of the last update.

To display additional columns, click the scroll bar on the bottom of the screen to scroll to the right or left. To alter the columns that display on the screen, or to change which query is executed to produce the list, click Modify Window Properties (...). To create new queries, refer to "Creating a query" on page 268

To view a profile or to add or change the information in these fields, double-click the row of the profile you wish to work with. The Mailbox Editor window displays, with the General tab in front.

The Mailbox Editor window contains two tabs: General and Comments. Use the General tab to enter and change information contained in the Mailbox profile. Use the Comments tab to type any comments you wish about the selected Mailbox profile.

Following are detailed procedures for creating new Mailbox profiles. For information on viewing, copying, editing, renaming, deleting, and printing profiles, "Performing common file management tasks" on page 45 For information on exporting profiles, "Exporting" on page 68

Creating Mailbox profiles

The first step in creating Mailbox profiles is determining:

- What individuals or groups require WebSphere Data Interchange services.
- Which network mailboxes those individuals or groups will use.

For example, you may receive a request from a user or group within your organization that requires the WebSphere Data Interchange services to connect to a trading partner. That user or group requires a Mailbox profile.

You must decide which network mailbox will best meet that user's needs. That decision may depend on the trading partner's network communications capabilities and which network they can use to communicate with your organization.

A Mailbox profile may be tied to a Network profile. A new trading partner relationship may require that you create a new Network profile, as well as a new Mailbox profile. You may also find that you can add a trading partner to existing Mailbox profiles, as long as the user requiring a new trading partner connection already has a Mailbox profile and the new trading partner can communicate under that profile's specifications.

You create a new Mailbox profile when you receive a request for WebSphere Data Interchange services from a user that either is not already connected to WebSphere Data Interchange or requires different communication specifications to meet the needs of a new trading partner relationship.

To create a Mailbox profile:

1. Click Setup on the WebSphere Data Interchange Client Navigator bar.

   The Setup window displays.

2. Click the Mailboxes tab.

   A list of the existing Mailbox profiles displays.

3. Click New on the tool bar.

   The Mailbox Editor window displays with the General tab open and all fields blank.

4. Complete the fields on the General tab. Required fields are preceded by a blue asterisk in WebSphere Data Interchange Client.

   Click ? for field descriptions.

5. Click the Comments tab and type any comments you have about the Mailbox profile into the Comments field.

6. Click Save on the tool bar to save the profile.

Chapter 14. WebSphere MQ queue profiles

WebSphere Data Interchange Client's WebSphere MQ profile is used to associate logical names with real WebSphere MQ message queues. To use the WebSphere MQ support in WebSphere Data Interchange, you must have WebSphere MQ for z/OS Version 5 Release 3, or CICS, installed, or, for AIX or Windows 2000, WebSphere MQ Version 5 Release 3, and have WebSphere MQ running so that WebSphere Data Interchange can access the message queues.

About WebSphere MQ queue profiles

The WebSphere MQ queue profile provides a way to associate processing options WebSphere Data Interchange will use whenever the WebSphere MQ profile is supplied. WebSphere MQ message queues can be used in place of most sequential files, and they can be the source to obtain a document or the target when a document will be written; and in CICS, WebSphere MQ queues can be used in event-driven processes. You can use WebSphere MQ queue profiles as part of a logical network where enveloped data is sent to and received from WebSphere MQ message queues.

Setup overview

You set up and maintain WebSphere MQ queue profiles through the WebSphere MQ queues List window, which you access by clicking Setup on the WebSphere Data Interchange Client Navigator bar. The Setup window, with tabs for WebSphere Data Interchange Client's setup profiles, displays. Click the WebSphere MQ queues tab, and the WebSphere MQ queues List window displays.

This window displays a list of existing WebSphere MQ queue profiles. Each row contains information about a WebSphere MQ queue profile; each column contains data stored in the profile. Information in the columns displays in fields, drop-down lists, and check boxes in the WebSphere MQ queues Editor window. The profile list window, however, also contains the date, time, and user ID of the last update.

To display additional columns, click the scroll bar on the bottom of the screen to scroll to the right or left. To alter the columns that display on the screen, or to change which query is executed to produce the list, click Modify Window Properties (...). To create new queries, refer to "Creating a query" on page 268

To view a profile or to add or change the information in these fields, double-click the row of the profile you wish to work with. The WebSphere MQ queue Editor window displays, with the General tab in front.

The WebSphere MQ queues Editor window contains two tabs: General and Comments. Use the General tab to enter and change information contained in the WebSphere MQ queue profile. Use the Comments tab to type any comments you wish about the selected WebSphere MQ queue profile.

Following are detailed procedures for creating new WebSphere MQ queue profiles. For information on viewing, copying, editing, renaming, deleting, and printing profiles, see "Performing common file management tasks" on page 45 For information on exporting profiles, see "Exporting" on page 68

Creating WebSphere MQ queue profiles

Create a new WebSphere MQ queue profile for each queue you will use.

To create a WebSphere MQ queue profile:

1. Click Setup on the WebSphere Data Interchange Client Navigator bar.

   The Setup window displays.

2. Click the WebSphere MQ queues tab.

   A list of the existing WebSphere MQ queue profiles displays.

3. Click New on the tool bar.

   The WebSphere MQ queues Editor window displays with the General tab open and all fields blank.

4. Complete the fields on the General tab.

   Click ![?]

   for field descriptions.

5. Click the Comments tab and type any comments you have about the WebSphere MQ queue profile in the Comments field.

6. Select Save from the File menu to save the profile.

Chapter 15. Network profiles

Network profiles define for WebSphere Data Interchange the characteristics of the networks you use for communications with trading partners. WebSphere Data Interchange Client is shipped with the network profiles required to communicate with several major networks (Table 14), and if you use those networks, you need not create any new network profiles.

If you are not using any of those networks, you need to create a new network profile. You can also bypass networks altogether and make a direct point-to-point connection between WebSphere Data Interchange and your trading partner's system.

Network profiles are also used to assist in identifying WebSphere MQ message queues used to send and receive documents.

Attention: Network profiles might need to have a corresponding set of network commands profiles to define the commands required to communicate with the network. To create network commands profiles, see Chapter 16, "Network commands profiles" on page 107. For further details in communicating with networks, see the *WebSphere Data Interchange Programmer's Reference*.

About network profiles

WebSphere Data Interchange supports two means of communicating with trading partners: network communications and direct communications. Network communications move data from WebSphere Data Interchange to a mailbox supported by a network service provider or a company. Direct communications are just that: communication links between WebSphere Data Interchange and a trading partner's system.

Create a network profile when using WebSphere MQ to exchange EDI data with your trading partners. For WebSphere MQ communication with trading partners, the network profile identifies the WebSphere MQ Queue profiles associated with the message queues used to send and receive documents. For more information, see Chapter 14, "WebSphere MQ queue profiles" on page 101.

WebSphere Data Interchange reads network parameters through network profiles. Any network you use must have a profile defined for WebSphere Data Interchange. To make network communications easier, WebSphere Data Interchange Client is shipped with profiles for IBM Global Services and the General Electric Information Services (GEIS), as listed in Table 14.

Table 14. Network profiles shipped with WebSphere Data Interchange Client

Profile	Communication Routine	Network Program	Used With:
GEIS	GEISVAN	DSXMIT2	General Electric Information Services Company
IINAIX	VANIINB1	IEBASE	Expedite for AIX V4.5
IINWIN	VANIINB1	IEBASE	Expedite for Windows V4.6
IINB41	VANIINB1	IEBASE	Expedite Base/MVS Version 4 Release 1

Network profiles

Table 14. Network profiles shipped with WebSphere Data Interchange Client (continued)

Profile	Communication Routine	Network Program	Used With:
IINB42	VANIINB1	IEBASE	Expedite Base/MVS Version 4 Release 2 and higher
IINCICS	VANINFC	EXPOICMD	Expedite/CICS
MQSAMP	VANIMQ	EDIMQSR	Sample WebSphere MQ Network

If you are not using IBM Global Services or the General Electric Information Services (GEIS) network, or if you want to use a point-to-point connection that bypasses the mailbox, or use WebSphere MQ queues to send and receive data, you must create a network profile.

WebSphere Data Interchange can communicate with any generalized network or make point-to-point communications directly with a trading partner's system. For more information on communicating with other networks and making point-to-point connections, see the *WebSphere Data Interchange Programmer's Reference*.

Setup overview

You set up and maintain network profiles through the Network Profiles List window, which you access by clicking Setup on the WebSphere Data Interchange Client Navigator bar. The Setup window, which contains WebSphere Data Interchange Client's setup profiles, displays. Click the Network Profiles tab, and the Network Profiles List window displays.

This window displays a list of networks your company can use to communicate with trading partners. Each row contains information about a network profile; each column contains data stored in that profile. Information in the columns displays in fields, drop-down lists, and check boxes in the Network Profiles Editor window. The profile list window, however, also contains the date, time, and user ID of the last update.

To display additional columns, click the scroll bar on the bottom of the screen to scroll to the right or left. To alter the columns that display on the screen, or to change which query is executed to produce the list, click Modify Window Properties (...). To create new queries, refer to "Creating a query" on page 268

To view a profile or to add or change the information in these fields, double-click the row of the profile you wish to work with. The Network Profiles Editor window displays, with the General tab in front.

The Network Profiles Editor window contains two tabs: General and Comments. Use the General tab to enter and change information contained in network profiles. Use the Comments tab to type any comments you wish about the selected network profile.

Following are detailed procedures for creating network profiles. For information on viewing, copying, editing, renaming, deleting, and printing profiles, see "Performing common file management tasks" on page 45. For information on exporting profiles, see "Exporting" on page 68.

Creating network profiles

The first step to working with network profiles is determining which networks your trading partners use, or which WebSphere MQ message queues documents will be received from or sent to. You need to create network profiles for any networks not listed in Table 14 on page 103. You also need to create a network profile to identify WebSphere MQ message queues that you will receive documents from or send documents to.

When you know which network profiles you need to create, gather the information required to complete the profile. Review the fields that display in the network profile. You need the name of a technical contact at the network for which you are creating a profile to supply some of the information required.

Create a new network profile to communicate with a network not listed in Table 14 on page 103.

To create a network profile:

1. Click Setup on the WebSphere Data Interchange Client Navigator bar.

   The Setup window displays.
2. Click the Network Profiles tab.

   A list of the existing network profiles displays.
3. Click New on the tool bar.

   The Network Profiles Editor window displays with the General tab open and all the fields blank.
4. Complete the fields on the General tab. Required fields are preceded by a blue asterisk.

   Click

   ?

   for field descriptions.
5. Click the Comments tab and type any comments you have about the selected network profile into the Comments field.
6. Click Save on the tool bar to save the profile.

Creating network profiles

Chapter 16. Network commands profiles

A network commands profile is used to prepare the network commands file that WebSphere Data Interchange sends to your network interface program.

You only need to create network commands profiles if you are setting up WebSphere Data Interchange to connect to networks for which WebSphere Data Interchange does not include network profiles. WebSphere Data Interchange is shipped with network profiles and network commands profiles for IBM Global Services and the General Electric Information Services network.

Attention: Each network commands profile is associated with a network profile. To create network profiles, see Chapter 15, "Network profiles" on page 103. For further details in communicating with networks, see the *WebSphere Data Interchange Programmer's Reference*.

About network commands

Every network requires certain commands to accomplish tasks like sending and receiving data. Those commands require information from WebSphere Data Interchange, such as a user ID or password. Network commands profiles define the commands required by the network and maps the WebSphere Data Interchange data they require to the network command file.

For example, a network may require a command called SENDX12 to send X12 files to the network. In order to send X12 data to that network, you must first define a Network profile for that network, and then define the network commands profiles that instruct WebSphere Data Interchange how to create the SENDX12 command. The SENDX12 command may require that WebSphere Data Interchange pass a user ID to the network. The network commands profiles for the SENDX12 command tells WebSphere Data Interchange where to map the data in WebSphere Data Interchange's User ID field to the network input file so that you can send X12 data to the network.

The network commands profiles map data within WebSphere Data Interchange into commands fed to a network message program, which, in turn, communicates with a network interface program. The network commands profiles are used to prepare commands to pass to the network's communications routines by specifying where WebSphere Data Interchange can locate the various bits of data required by the network.

Creating network commands sometimes requires making changes to the WebSphere Data Interchange code itself. If you find that you cannot create all the network commands you need through the network commands profile, contact your WebSphere Data Interchange technical support representative.

Note: One or more network commands profiles are used to construct each network command.

Setup overview

You set up and maintain network commands profiles through the network commands List window, which you access by clicking Setup on the WebSphere Data Interchange Client Navigator bar. The Setup window, which contains

Creating network commands profiles

The network commands Editor window displays with the General tab open and all the fields blank.

4. Complete the fields on the General tab.

    Click ?

    for field descriptions.

5. Select Save from the File menu to save the profile.

Creating network commands profiles

Chapter 17. Network security profiles

The network security profile allows WebSphere Data Interchange to support security applications. You can protect data transfers by adding encryption and authentication processing through security applications shipped with WebSphere Data Interchange or through third-party security applications.

WebSphere Data Interchange allows you to set up filters that screen data for special characters. The system also supports file compression.

Attention: A third-party or in-house security application entered into a network security profile must have a corresponding entry in a user exit profile. (Security applications shipped with WebSphere Data Interchange do not.) For information on user exit profiles, see Chapter 18, "User exit profiles" on page 115.

About network security profiles

Network security profiles allow you to add encryption, authentication, compression, and filtering applications to your WebSphere Data Interchange system. The network security profile provides WebSphere Data Interchange with the information it needs to pass messages to security applications for processing before sending data on the outbound side and before translating data on the inbound side.

Security services are often called "cryptographic services" and are defined below:

- **Encryption**

  Refers to a process of scrambling a message before it is sent so that it cannot be read if intercepted by an unauthorized person during transmission. The receiver must have the appropriate key to decipher an encrypted message. Both the sender and receiver must use compatible encryption applications.

- **Authentication**

  Refers to a process of verifying the identity of the sender and certifying the validity of the contents. When messages are authenticated, the receiver knows that the sender is who he says he is and that the contents of the message have not been corrupted during transmission. Both the sender and receiver must use compatible authentication applications.

- **Compression**

  Refers to a process of shortening the length of data. By compressing files, you shorten transmission times and reduce disk usage when storing files.

- **Filtering**

  Refers to a process of screening data before it is transmitted. You filter data to make sure that it does not contain characters that perform special functions on the network. If a network is sensitive to a particular character, you must run filtering when you use encryption services to make sure that the encrypted file does not contain the invalid character.

For more information on using security with WebSphere Data Interchange, see the *WebSphere Data Interchange Programmer's Reference*.

Note: WebSphere Data Interchange provides an authentication program named IBMNSPA (load module EDITRAA). IBMNSPA does not require a corresponding user exit profile.

Network security profiles

WebSphere Data Interchange provides an encryption interface program named IBMNSPE (load module EDITREE). IBMNSPE does not require a corresponding user exit profile.

You can set up network security profiles to handle either encryption or authentication, or both. Filtering and compression work only in conjunction with encryption; you cannot set them up as separate processes. You can set up as many network security profiles as you need to handle security arrangements made with trading partners.

Note: To associate the same network security profile with many trading partners, type the appropriate network security profile ID in the network security profile ID field of the trading partner profile.

Setup overview

You set up and maintain network security profiles through the network security list window, which you access by clicking Setup on the WebSphere Data Interchange Client Navigator bar. The Setup window, which contains WebSphere Data Interchange Client's setup profiles, displays. Click the Network Security tab and the network security list window displays.

This window displays a list of existing network security profiles. Each row contains information about a network security profile; each column contains data stored in the profile. Information in the columns displays in fields, drop-down lists, and check boxes in the network security editor window. The profile list window, however, also contains the date, time, and user ID of the last update.

To display additional columns, click the scroll bar on the bottom of the screen to scroll to the right or left. To alter the columns that display on the screen, or to change which query is executed to produce the list, click Modify Window Properties (...). To create new queries, refer to "Creating a query" on page 268

To view a profile or to add or change the information in these fields, double-click the row of the profile you wish to work with. The network security editor window displays, with the General tab in front.

The network security editor window contains two tabs: General and Comments. Use the General tab to enter and change information contained in network security profiles. Use the Comments tab to type any comments you wish about the selected network security profile.

Following are detailed procedures for creating new network security profiles. For information on viewing, copying, editing, renaming, deleting, and printing profiles, "Performing common file management tasks" on page 45 For information on exporting profiles, "Exporting" on page 68

Creating network security profiles

The first step in creating network security profiles is determining whether you want to add security services to your messages. You must make this decision in conjunction with trading partners, as a trading partner must use compatible security software to send and receive encrypted and authenticated messages.

When you have decided to use security services (also called cryptographic services), you need to decide which security applications to use. You may use the

programs supplied with WebSphere Data Interchange for both encryption and authentication or third-party security software.

Attention: If you use an in-house or third-party security program, you must first set up a user exit profile for the program. For details, see Chapter 18, "User exit profiles" on page 115

For more information on using security with WebSphere Data Interchange, see the *WebSphere Data Interchange Programmer's Reference*.

You create a new Network Security profile when you want to use security services when exchanging files with a trading partner. You may need to set up different Network Security profiles to meet different trading partner requirements.

For example, say you have set up a Network Security profile that supports both encryption and authentication. If another trading partner that wants to use security services only wants to use encryption, then you need to create a new Network Security profile for that trading partner. You can use the same Network Security profile for trading partners that have the same security requirements.

To create a network security profile:

1. Click Setup on the WebSphere Data Interchange Client Navigator bar.

   The Setup window displays.
2. Click the Network Security tab.

   A list of the existing Network Security profiles displays.
3. Click New on the tool bar.

   The Network Security Editor window displays with the General tab open and all the fields blank.
4. Complete the fields on the General tab. Required fields are preceded by a blue asterisk.

   Click ❓ for field descriptions.
5. Click the Comments tab and type any comments you have about the selected network security profile into the Comments field.
6. Click Save on the tool bar to save the profile.

Creating network security profiles

Chapter 18. User exit profiles

A user exit profile provides WebSphere Data Interchange with the information it needs to call other programs. For example, if you want to send and receive encrypted messages, you need to call a separate program to perform cryptographic services. WebSphere Data Interchange identifies that security program through a user exit profile.

Note: Some utilities supplied with WebSphere Data Interchange do not require user exit profiles. This is indicated in the reference information for the utilities.

About user exit profiles

User exit profiles contain the information that WebSphere Data Interchange needs to identify separate programs that WebSphere Data Interchange calls to provide supplemental processing. You need a user exit profile for each external program or exit routine you use.

WebSphere Data Interchange supports programs for:
- Network communications
- Security
- Data Mapping
- Network message handling
- Point-to-Point network communications

Setup overview

You set up and maintain user exit profiles through the user exit list window, which you access by clicking Setup on the WebSphere Data Interchange Client Navigator bar. The Setup window, which contains WebSphere Data Interchange Client's setup profiles, displays. Click the user exits tab, and the user exit list window displays.

This window displays a list of existing user exit profiles. Each row contains information about a user exit profile; each column contains data stored in the profile. Information in the columns displays in fields, drop-down lists, and check boxes in the user exit editor window. The profile list window, however, also contains the date, time, and user ID of the last update.

To display additional columns, click the scroll bar on the bottom of the screen to scroll to the right or left. To alter the columns that display on the screen, or to change which query is executed to produce the list, click Modify Window Properties (...). To create new queries, refer to "Creating a query" on page 268

To view a profile or to add or change the information in these fields, double-click the row of the profile you wish to work with. The user exit editor window displays, with the General tab in front.

The user exit editor window contains two tabs: General and Comments. Use the General tab to enter and change information contained in the user exit profile. Use the Comments tab to type any comments you wish about the selected user exit profile.

User exit profiles

Following are detailed procedures for creating new user exit profiles. For information on viewing, copying, editing, renaming, deleting, and printing profiles, see "Performing common file management tasks" on page 45 For information on exporting profiles, see "Exporting" on page 68

Creating user exit profiles

The first step in creating user exit profiles is identifying the external program or utility that you need WebSphere Data Interchange to call. Remember that some utilities shipped with WebSphere Data Interchange do not require user exit profiles.

You create a new user exit profile whenever you start using another external program for supplemental processing.

To create a Data Transformation user exit profile:

1. Click Setup on the WebSphere Data Interchange Client Navigator bar.

   The Setup window displays.

2. Click the user exits tab.

   A list of the existing user exit profiles displays.

3. Click New on the tool bar or select New from the File menu.

   The Create a User Exit Profile wizard displays.

4. Type a user exit name in the Exit Name field.

   You can use both letters and numbers to identify your exit. Letters display in capitals. You cannot type spaces within the name.

   You can enter a more complete description of the user exit n the Description field.

   Click Next to continue. The Exit Type wizard page displays.

5. Select the appropriate radio button to create a Data Transformation User Exit profile.

   Click Next to continue. The Program Name and Language wizard page displays.

6. Select the language the User Exit profile is written in from the Progam Language drop-down list, the name of the DLL the exit is located in, and the name of the user exit routine.

   Click Next to continue. The Input/Output Information wizard page displays.

7. Select the appropriate radio button to indicate the format of the strings that will be passed to and from the exit routine. You should also specify the maximum length of the output that the user exit program can produce.

   Click Next to continue. The confirmation wizard page displays.

8. Click Finish to confirm your selections, click Back to return to the previous page, or click Cancel to exit the Wizard without creating a user exit profile.

9. When you have created a user exit profile, it appears in the User Exits list window. Double-click on the profile to open the User Exit Editor. You can use the fields in the General tab to change the properties of the exit profile.

   Click [?] for field descriptions.

10. Click the Comments tab and type any comments you have about the selected user exit profile into the Comments field.

11. Select Save from the File menu to save the new user exit profile.

To create a Send/Receive user exit profile:

1. Click Setup on the WebSphere Data Interchange Client Navigator bar.

   The Setup window displays.

2. Click the user exits tab.

   A list of the existing user exit profiles displays.

3. Click New on the tool bar or select New from the File menu.

   The Create a User Exit wizard displays.

4. Type a user exit name in the Exit Name field.

   You can use both letters and numbers to identify your exit. Letters display in capitals. You cannot type spaces within the name.

   You can enter a more complete description of the user exit n the Description field.

   Click Next to continue. The Exit Type wizard page displays.

5. Choose whether you are creating a Data Transformation User Exit or a Send or Receive User Exit by selecting the appropriate radio button.

   Click Next to continue. The Program Name and Language wizard page displays.

6. Select the language the User Exit is written in from the Progam Language drop-down list.

   Enter the name of the load module the user program or exit routine is stored in. The Exit Uses wizard page displays.

   Click Next to contine. The Exit Uses wizard page displays.

7. Click the relevant check-boxes to select the types of user exits the program can be used for.

   Click Next to continue. The Confirmation wizard page displays.

8. Click Finish to confirm your selections, click Back to return to the previous page, or click Cancel to exit the Wizard without creating a user exit profile.

9. When you have created a user exit profile, it appears in the User Exits list window. Double-click on the profile to open the User Exit Editor. You can use the fields in the General tab to change the properties of the exit profile.

   Click for field descriptions.

10. Click the Comments tab and type any comments you have about the selected user exit profile into the Comments field.

11. Select Save from the File menu to save the new user exit profile.

Creating user exit profiles

Chapter 19. Message Content Descriptor profiles

With the Message Content Descriptor (MCD) profile, you can map the names of message definitions in other products, such as WebSphere Message Repository Manager (MRM) and Crossworlds®, to the names of message definitions within WebSphere Data Interchange when communicating using the MQSeries Rules and Formatting Header 2 (MQRFH2). The MCD profile is also used when communicating with Java Message Service (JMS) clients. For more information about how the fields in the MCD map to the JMS API see the *WebSphere Data Interchange Programmer's Reference*.

Setup overview

To set up and maintain MCD profiles through the MCD Profiles List window, click **Setup** on the WebSphere Data Interchange Client Navigator bar. The Setup window, which contains tabs for WebSphere Data Interchange Client's profiles, opens. Click the **MCD Profiles** tab to open the MCD Profiles List window.

This window displays a list of existing MCD profiles. Each row contains information about an MCD profile and each column contains data stored in the profile. Information in the columns displays in fields, drop-down lists, and check boxes in the MCD Profiles Editor window. The profile list window, however, also contains the date, time, and user ID of the last update.

To display additional columns, click the scroll bar on the bottom of the screen to scroll to the right or left. To alter the columns that display on the screen, or to change which query is executed to produce the list, click **Modify Window Properties**. To create new queries, refer to "Creating a query" on page 268.

To view a profile or to add or change the information in these fields, double-click the row of the profile you wish to work with. The MCD Profiles Editor window opens, with the General tab in front.

The MCD Profiles Editor window contains two tabs: General and Comments. Use the General tab to add or change information contained in the MCD profile. Use the Comments tab to add any comments you wish about the selected MCD profile.

Following are detailed procedures for creating new MCD profiles. For information on viewing, editing, and deleting profiles, see "Performing common file management tasks" on page 45. For information on exporting profiles, see "Exporting" on page 68.

Creating MCD profiles

Create a new MCD profile when exchanging documents using WebSphere MQ message queues or JMS, and the message containing the document is prefixed with an MQRFH2 header. The MQRFH2 header contains an MCD folder. The name of a WebSphere Data Interchange document definition occurs in an MCD profile, along with the corresponding values from the MCD of the MQRFH2 header.

1. On the WebSphere Data Interchange Client Navigator bar, click **Setup**.

   The Setup window opens.

2. Click the **MCD Profiles** tab.

   A list of the existing MCD profiles displays.

Creating MCD profiles

3. On the tool bar, click New.

   The MCD Profiles Editor window opens with the General tab in front and all fields blank.

4. Complete the fields on the General tab. Required fields are preceded by a blue asterisk.

   Click [?] for field descriptions.

5. Click the **Comments** tab and type any comments you have about the MCD profile into the **Comments** field.

6. From the File menu, click **Save** to save the profile.

   WebSphere Data Interchange saves the new MCD profile to the database.

7. Close the editor.

Chapter 20. 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.

Though this option is designed to work in a general manner and has many possible applications, it was primarily designed for situations where an input source contains both documents that require single translation and documents that require double translation. With command chaining, you can set up a double translation by routing the output from the first translation to a file with the same name as a Service profile.

The name you give 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, chaining the commands together.

Substitution keywords

You can substitute values of elements in the <mcd> and <usr> folders of the RFH2 header of a message received from a WMQ queue into a PERFORM command in a service profile. You can also substitute some of the values associated with previous commands in a command chain into the next command in the chain.

Specify the name of the keyword that contains the value you want to substitute into the command preceeded by an ampersand (&). For example, to substitute in the name of the Data Format from the <Set> and <Type> elements of the <mcd> folder you would enter:

```
PERFORM TRANSFORM WHERE SYNTAX(D) DICTIONARY(&SET) DOCUMENT(&TYPE)
```

If you want the values translated using an MCD profile before substituting them in (see chapter on MCD profiles) then you would enter:

```
PERFORM TRANSFORM WHERE SYNTAX(&SYNTAX) DICTIONARY(&DICTIONARY) DOCUMENT(&DOCUMENT)
```

If you want to set the value of the BATCHSET keword from an element in the user folder named ″abc″, enter

```
PERFORM TRANSFORM WHERE BATCHSET(&abc)
```

The keywords you can use to access the elements of the <mcd> folder are:

&MSD Message service domain. This is the name space that contains the name specified in the three other fields in the <mcd> (Set, Type and Fmt). It corresponds to the value of the <msd> element in the <mcd> folder of the RFH2 header. The name spaces that have already been allocated are:

> **mrm** The WebSphere MQ Integrator (WMQI) Message Repository Manager (MRM) name space. This means that the meta-data definition of the attached message can be found in the WebSphere MQ Integrator MRM under the name contained in the <Set>, <Type> and <Fmt> elements. You normally use this name space when communicating with WebSphere MQ Integrator, but you can also use the xml name space (see below).

Substitution keywords

none The meta data for the attached message is not specified. Do not specify values for the <Set>, <Type> and <Fmt> elements if you set the <msd> to none.

wdi The WebSphere Data Interchange name space. This means that the meta data definition of the attached message can be found in WebSphere Data Interchange. The dictionary name is contained in the <Set> element (for example ″EDI00B″ is the WebSphere Data Interchange dictionary for the EDIFACT 00.B standards). The document name is contained in the <Type> element (for example ″ORDERS″ for the EDIFACT purchase order message. The syntax (adf, edi, or xml) is contained in the <Fmt> element.

xml The message is a self-defining message in XML syntax. The values of any other elements in the <mcd> folder are ignored and the meta data name (if required) is taken from the DOCTYPE tag in the XML prolog of the message.

&SET Message set name. This is the value of the <Set> folder in the <mcd> folder of the RFH2 header. If the value of the <msd> element (the namespace) is ″mrm″, then this element is the WebSphere MQ Integrator MRM message set name. If the name space is ″wdi″, then it is the dictionary name.

&TYPE
Message set type. This is the value of the <Set> folder in the <mcd> folder of the RFH2 header. If the value of the <msd> element (the namespace) is ″mrm″, then this element is the WebSphere MQ Integrator MRM message Type name. If the name space is ″wdi″, then it is the document name.

&FORMAT
Message format. This is the value of the <Fmt> folder in the <mcd> folder of the RFH2 header. If the value of the <msd> element (the namespace) is ″mrm″, then this element is xml for xml messages, pdf for MTI bitstream messages or a custom wire format identifier. If the name space is ″wdi″, then it is the document syntax (adf, edi, or xml).

The keywords you can use to refer to the meta data name (within the WebSphere Data Interchange name space) of the input file of a service profile are:

&SYNTAX
The syntax of the input message of the service profile. It will be either adf (data format), edi or xml.

&DICTIONARY
The dictionary that contains the definition of the data in the input message. If the input file contained a data format syntax message, then this is the data format dictionary that contains the data format definition that describes the data in the input message, for example SAP_R3_IDOC_V4 if the input message was a version 4 IDOC

&DOCUMENT
The name of the definition of the input message. If the input file contained a data format syntax message, then this is the name of the data format that describes the data in the input message, for example ORDERS01 if the input message was an ORDERS01 message from a SAP R/3 system.

Note: There is only one set of values for the syntax, dictionary and document of each instance of input to a service profile. If you use these keywords, then if your input contains multiple messages in a single instance (i.e. a file with 10

purchase orders in it, or a queue where one MQGET retrieves a buffer with 10 purchase orders in it), then all the messages must use the same values for the keyword. Therefore, if you use the &SYNTAX keyword, then all the messages in a single instance of the input must be of the same syntax (adf, edi or xml).

Setup overview

To set up and maintain Service profiles through the Service Profiles List window, click **Setup** on the WebSphere Data Interchange Client Navigator bar. The Setup window, which contains tabs for WebSphere Data Interchange Client's setup profiles, opens. Click the **Service Profiles** tab, and the Service Profiles List window opens.

This window displays a list of Service profiles. Each row contains information about a Service profile and each column contains data stored in the profile. Information in the columns displays in fields, drop-down lists, and check boxes in the Service Profiles Editor window. The profile list window, however, also contains the date, time, and user ID of the last update.

To display additional columns, click the scroll bar on the bottom of the screen to scroll to the right or left. To alter the columns that display on the screen, or to change which query is executed to produce the list, click **Modify Window Properties**. To create new queries, refer to "Creating a query" on page 268.

To view a profile or to add or change the information in these fields, double-click the row of the profile you wish to work with. The Service Profiles Editor window opens, with the General tab in front.

The Service Profiles Editor window contains seven tabs: General, Common Files, Input Files, Output Files, Network Files, Export/Import Files, and Comments. Use the General tab to enter and change information contained in the Service profile. Use the other tabs to add other pertinent information about the Service profile. Whether you are running on a Windows or AIX server, be sure to follow the rules for names and path names for that server. You can use the relative or the absolute path name. The relative path name is to the current directory that the task was started from, or where the trigger started the task.

"Creating Service profiles" provides detailed procedures for creating new Service profiles. For information on viewing, copying, editing, renaming, deleting, and printing profiles, see "Performing common file management tasks" on page 45. For information on exporting profiles, see "Exporting" on page 68.

Creating Service profiles

Create a new Service profile when you need to set up a WebSphere Data Interchange PERFORM command to be invoked when an output file is closed at the end of other WebSphere Data Interchange command processing.

Within WebSphere Data Interchange, files have logical and physical names. At runtime, the logical names are mapped to the physical names.

Plan the chain of command, and then create the Service profile for each of the steps in your chain.

1. On the WebSphere Data Interchange Client Navigator bar, click **Setup**.

Creating Service profiles

The Setup window opens.

2. Click the **Service Profiles** tab.

   A list of the existing Service profiles displays.

3. On the tool bar, click **New**.

   The Service Profiles Editor window opens with the General tab in front and all fields blank.

4. Complete the fields on the General tab. Required fields are preceded by a blue asterisk.

   Click [?] for field descriptions.

5. Click the **Common Files** tab.

   The fields on the Common Files tab display.

   Note: If a file is not specified here, the file from the previous command is used.

   Tip: It is usually best not to specify the print file, so that all the messages for all the commands in the chain will go to the same file.

6. Fill in the fields as needed. The Common Files tab contains only fixed names.

7. Click the **Input Files** tab.

   The fields on the Input Files tab display.

   Note: This tab is only used if the input to the command is different than the file that triggered the command. The input file is usually the name of the file that invoked this command. That file is automatically available.

8. Fill in the fields as needed. The Input Files tab contains logical and physical names.

   Click [?] for field descriptions.

9. Click the **Output Files** tab.

   The fields on the Output Files tab display.

   Note: You must specify all possible output files for the command.

10. Fill in the fields as needed. The Output Files tab contains logical and physical names.

    Click [?] for field descriptions.

11. Click the **Network Files** tab.

    The fields on the Network Files tab display.

12. Fill in the fields as needed. The Network Files tab contains fixed names. Click

    [?]

    for field descriptions.

13. Click the **Export/Import Files** tab.

    The fields on the Export/Input Files tab display.

    Note: You can specify all the object files to be the same file.

14. Fill in the fields as needed. The Export/Import Files tab contains fixed names.

Click ![?] for field descriptions.

15. Click the **Comments** tab and add any comments you have about the Service profile into the Comments field.

16. From the file menu, click **Save** to save the profile.

    WebSphere Data Interchange saves the new MCD profile to the database.

17. Close the editor.

Creating Service profiles

Part 3. Trading partners

127

Chapter 21. Trading partners

Managing trading partner profiles is one of the most essential tasks. Whenever your organization begins exchanging documents with a new trading partner, you need to create a new trading partner profile. Trading partner profiles identify your trading partners to WebSphere Data Interchange and specify the details of how to exchange documents with each trading partner.

Overview of trading partner communications

To exchange data with your trading partners, WebSphere Data Interchange needs information about the network, your organization, and your trading partners. After you define the information, WebSphere Data Interchange can use a communications routine to send the data to the network. Your trading partners retrieve the data from the network to their system. These tasks are necessary to exchange data with your trading partners.

1. Set up the WebSphere MQ Queue profile

   This profile defines logical names with real WebSphere MQ Queues. The profile provides a way to associate the processing options that WebSphere Data Interchange uses whenever you specify an WebSphere MQ Queue profile member. See Chapter 14, "WebSphere MQ queue profiles" on page 101 for more information.

2. Set up the network profile

   This profile defines which networks you use to exchange information with your trading partners, and which communication routine you use. See Chapter 15, "Network profiles" on page 103 for more information.

3. Set up the mailbox (requestor) profile

   This profile defines the individuals or group of individuals, such as an accounting department, who use WebSphere Data Interchange. You can define as many individuals or groups as your organization needs. See Chapter 13, "Mailbox profiles" on page 99 for more information.

4. Set up the trading partner profile

   This profile defines your application trading partners and EDI trading partners, and includes information about account numbers or account IDs, user IDs, who pays for network charges, and other optional information.

You can eliminate the network and go directly from a WebSphere Data Interchange system to another system using a point-to-point setup. For more information about this type of setup, see the chapter on interfacing with other networks and applications in the *WebSphere Data Interchange Programmer's Reference*.

WebSphere Data Interchange also supports exchanging data with your trading partner using an WebSphere MQ message queue. You must define an WebSphere MQ profile number for each message queue. For detailed information on setting up WebSphere MQ communications, see "Interfacing WebSphere Data Interchange with WebSphere MQ" in the *WebSphere Data Interchange Programmer's Reference*.

CICS systems can establish a process that automates receiving transactions. See the *WebSphere Data Interchange Programmer's Reference* for details.

Trading partners

WebSphere Data Interchange uses profiles to store the information it needs to communicate with your trading partners. Although WebSphere Data Interchange is shipped with many profiles, this chapter describes only the profiles that you need to establish trading partner communications.

WebSphere Data Interchange provides the profiles. It also provides several members for some profiles that you can use without changing. For other profiles, you must either add profile members or update profile members that are provided. You must have the authority to work with the profiles.

About trading partner profiles

Trading partner profiles allow WebSphere Data Interchange to maintain information about your company's trading partners and to define the essential features of your connections. This section provides an overview of the purpose of trading partner profiles and how you set them up.

Purpose

WebSphere Data Interchange Client maintains essential information on your trading partners through trading partner profiles. For example, the trading partner profile for a trading partner called JB Smith & Company would contain basic information such as company name, address, telephone number, and so on, as well as information used in translation, such as trading partner Nickname, Network ID, Account and User ID information, and Control Numbers.

Setup overview

You set up and maintain trading partner profiles through the trading partner List window, which you access by clicking Trading Partner on the WebSphere Data Interchange Client Navigator bar.

The trading partner list window contains two tabs, trading partners and contacts. To work with a trading partner or contact, click the appropriate tab to display a list of current trading partner or contact profiles. Each tab displays a list of existing profiles. Each row contains information about a profile; each column contains data stored in the profile. Information in the columns displays in fields, drop-down lists, and check boxes in the trading partner or contact Editor window. The profile list window, however, also contains the date, time, and user ID of the last update.

To display additional columns, click the scroll bar on the bottom of the screen to scroll to the right or left. To alter the columns that display on the screen, or to change which query is executed to produce the list, click Modify Window Properties (...). To create new queries, refer to "Creating a query" on page 268

Note: When you have used WebSphere Data Interchange Client for a while, your list of trading partner profiles may grow to the thousands. To shorten the length of time it takes to display the list, create a query that filters the list so that it displays only the set of trading partners you wish to view, and only the columns you really need to see.

To view a profile or to add or change the information in these fields, double-click the row of the profile you wish to work with, or click the trading partner profile you want to work with and then click Open in the window's tool bar. The profile's editor window displays, with the General tab in front.

The trading partner editor window contains seven tabs, each of which displays specific information about only the selected trading partner profile. You enter specific information by clicking the tabs and filling in the fields. When you save any changes, WebSphere Data Interchange Client displays this information in the trading partner profiles list.

Note: To customize the field tags on the User Fields tab page, see "Customizing field tags" on page 51.

To view the contacts editor, click the Contacts tab in the trading partner list window. Note that the contacts editor is not the same as the Contacts tab on the trading partner editor window. The contacts editor allows you to compile and maintain a list of people and organizations you contact, as described on 142. The Contacts tab on the trading partner editor lists the contacts associated with the trading partner. The tab can be used to associate additional contacts with the trading partner or remove existing associations, as shown on 132.

Creating trading partner profiles

This section contains detailed procedures for creating new trading partner and contact profiles. For information on viewing, copying, editing, renaming, deleting, and printing profiles, see "Performing common file management tasks" on page 45 For information on exporting profiles, see "Exporting" on page 68

To create trading partner profiles:

1. Click trading partner on the WebSphere Data Interchange Client Navigator bar.

   The trading partners list window displays. Click the Trading Partners tab if it is not already in front. This displays a listing of existing trading partner profiles.
2. Click New on the tool bar.

   The trading partner Editor window displays, with the General tab in front.
3. Complete the TP Nickname field. This is the only required field on this screen, as indicated by the blue asterisk. This field names the trading partner profile.
4. Enter information in the other fields on the General Tab as desired.

   Click ?

   for field descriptions.
5. Click Options. One of two dialog boxes displays, depending on the network you use. For more information on networks, see the *WebSphere Data Interchange for z/OS Administration Guide*.

   Note: The Options button only becomes available after you make a selection from the Network ID drop-down list. The IBM Global Services Addressing dialog box displays.

   The network you use to communicate with a trading partner has been set up through a Network profile. A network displays in the list only after you create a Network profile for it. For details, see Chapter 15, "Network profiles" on page 103

   Click ?

   for field descriptions.

Creating trading partner profiles

1. Click Edit TP Control Numbers if you want to view, create, or edit the control numbers pair information. The Control Numbers List window displays with the following information: TP nickname, receiver ID, receiver qualifier, transaction ID, interchange control number, group control number, and transaction control number.

   a. To view the information, click the left and right arrows on the scroll bar at the bottom of the Control Numbers List window.

   b. To change information about control number pairs, double click the selection on the list window, and make any changes on the General tab page. Click Save.

   c. To add control number pairs, click New. Fill in the required fields of Receiver Attributes and any optional fields you want to add, then click Save.

2. Enter information in the fields on the Company Info Tab, as desired.

3. Click the Contacts tab, and select any contacts you want to associate with this trading partner from the Contact Name drop-down list.

   The column in the grid fills in with data from the Contact profile you selected. Your contacts must be added before you can select them within your trading partner profile.

   To add a Contact to the list, see "Creating contacts" on page 142

4. Click the Comments tab and type any comments you have about the selected trading partner into the Comments field.

5. The fields on the EDI Options tab, DI Proc Options tab, and the User Fields tab are optional.

   Click ⟦?⟧

   for field descriptions.

   Note: User Fields tab fields can be customized by the user. You can store anything you consider relevant to the trading partner in the 10 fields on this tab.

6. Click Save on the tool bar.

   WebSphere Data Interchange Client saves the new trading partner profile.

Understanding processes and rules

For WebSphere Data Interchange to transform a document from the source format to the target format, a map is needed. Maps tell WebSphere Data Interchange how elements in the source document correspond to the elements in the target document. The overall process of determining which map to use to transform a given document is known as map resolution. The simplest form of map resolution is to always use the same set of routing and transformation instructions for a particular source document. This is common when exchanging documents between two internal applications, and is usually referred to as Application Integration or Enterprise Application Integration (EAI). However, this is typically not sufficient for Business to Business (B2B) requirements.

In a B2B scenario, routing and transformation requirements for a given type of source document also typically depend upon the sender and receiver of the document. For example, a purchase order that came from Company X's SAP system in IDOC format may need to be translated into cXML if it is destined for Company A because that is what Company A expects. If that same purchase order is instead destined for Company B, the format must be transformed into an EDIX12

850. In this B2B scenario, the type of transformation depends on both the document type (purchase order in SAP IDOC format) and the receiver (Company A or Company B).

Additionally, the destination is usually specified within the content of the document itself, not in a special header or separate document prepared specifically for WebSphere Data Interchange. The content of the document must be parsed to determine its destination and the type of transformation. This process of examining the contents of a document to determine how to route and transform it is usually referred to as Content Based Routing and Transformation. It is one of the supported map resolution processes within WebSphere Data Interchange, and is the most commonly used method for B2B applications. Routing and transformation instructions are usually referred to as rules, and within WebSphere Data Interchange are also known as usages because they determine when to use a specific map.

To continue the previous example, in order to specify to WebSphere Data Interchange how to properly transform the SAP IDOC purchase order for Company A and Company B, we must create some rules (or usages). The simplest way to do this within WebSphere Data Interchange is to enter two rules: one that specifies if a SAP IDOC formatted purchase order is sent to Company A, then transform it to cXML, and one that specifies if a SAP IDOC formatted purchase order is sent to Company B, then transform it to an EDI X12 850. Since this is a very simple example, it is most appropriate to use a very simple set of rules. Many customers will have B2B environments that are much more complicated than this example of one document (the purchase order) and two trading partners (Company A and Company B). These customers will require a more complex set of rules to support their environment.

Now consider a hypothetical, but realistic example of a large scale B2B environment.

Company X has 10,000 trading partners instead of two, and instead of just one document they need to exchange 10 different types of documents (catalog update, purchasing forecast, purchase order, purchase order acknowledgment, purchase order change, purchase order change acknowledgment, advanced ship notice, receiving advice, invoice and remittance advice). Consider how many rules Company X would have to create to link those 10,000 trading partners to the 10 different documents. If there is a rule for each trading partner, for each document, then the formula is number of rules = number of documents x the number of trading partners, or 10 X 10,000 = 100,000. Entering and maintaining 100,000 rules can be an onerous task. This total can be dramatically reduced because many of those 10,000 trading partners exchange exactly the same documents and require the same transformations. Only the routing differs between them.

These 10,000 trading partners can be broadly classified into two categories: those that require their documents to be in cXML format and those that require them in EDI format. If you create rules between the categories and the documents, rather than between the trading partners and the documents, then you have only 10 documents x 2 categories = 20 rules. Twenty is a far more manageable number than 100,000, so WebSphere Data Interchange allows you to categorize trading partners by what is known as a process. WebSphere Data Interchange also allows you to create a rule between a process (or category of trading partner) and a map, as well as between a trading partner and a map.

Processes and rules

The term "process" refers to a collection of maps to be used to interact with a set of trading partners. Such a set of maps is commonly referred to as a public process because it is a sequence of pubic (external to your company) document exchanges which implement some business process. Examples of such a business process are production purchasing or health care claim processing. The sequence of steps that take place inside your company to implement the business process are known as a private process. Processes in WebSphere Data Interchange can be either public or private processes.

Versioning is another important aspect of processes and is accomplished in WebSphere Data Interchange using naming conventions. Suppose that after a number of years of using your cXML process, you want to update the process to take advantage of some new features in the latest versions of the cXML documents with a few of your old trading partners and the new trading partners you add. You do not need to exploit the new features in the latest versions of the cXML documents with the vast majority of your current cXML trading partners, so you do not want to alter the way you transact business with them. So, you create a new version of the cXML process named cXML-V2 and switch the specific current customers that you need on the new version to the cXML-V2 process and add all new trading partners to the new version, as well.

Finally, suppose that some particular trading partner has a special requirement for a single document; for example, they cannot use the EDI 850 the way you normally send it. You create a map that transforms your SAP IDOC purchase order into an EDI 850 just the way they require it. Is this a new version of your purchasing process? Not really, it is just a single trading partner that had a specific requirement on a single document. Consequently, instead of creating a new version of your EDI process, you can override the map used when sending a purchase order to only that trading partner by creating a rule that links the trading partner to the specialized map.

Note: Trading partner rules, including generic rules, are considered to be of higher precedence than process rules. Therefore, process rules will only be used if no trading partner rules apply.

Understanding minimal trading partners

Though processes address the problem of minimizing the number of rules required to correctly transform a given set of documents to be exchanged with a given set of trading partners, it does not address the issue of reducing the number of trading partner profiles required. For example, in the previous example Company X has 10,000 trading partners. Implementing processes reduced the required number of rules from 100,000 to 20, but did not reduce the number of trading partners in the trading partner profile. There were still 10,000 trading partners to be entered and maintained.

The most fundamental purpose of the trading partner profile is to identify the network address that a message to a trading partner should be routed to, or the network address that appears as the sender of a message from that trading partner. If the WebSphere Data Interchange user is willing to provide the network addresses of the sender and receiver for each outbound document passed into WebSphere Data Interchange for processing, and accept network addresses as the identifiers for the sender and receiver of inbound documents, the trading partner profile is not required. So, using the Minimal trading partners feature of WebSphere Data

Interchange, a user with many trading partners can significantly reduce the number of trading partner profiles, and consequently, rules needed for a given set of trading partners and maps.

When implementing Minimal trading partners, the network addresses are stored in your applications instead of WebSphere Data Interchange; thus, you might need to modify the application before you begin using Minimal trading partners. The network address has two parts referred to in the trading partner Profile as the interchange ID and qualifier. The user's application must pass in the values to be used for the interchange ID and qualifier on the C record if C and D records are being used, or in predefined elements within the document to be transformed.

If you are using the Minimal trading partners feature, you can still create profiles for trading partners and use them as you normally would. Indeed, this is part of the architecture of the Minimal Trading Partners feature because it is how you override the standard behavior for trading partners that have special requirements or need special processing that cannot be handled by the Minimal trading partner feature. The Minimal trading partners feature and the trading partner profile are, therefore, complimentary rather than mutually exclusive.

Minimal trading partners scenario

Consider once again Company X, which has 10,000 trading partners. Company X must create its own common trading partner profile to specify the delimiters to be used when sending to these trading partners. This profile is the default trading partner profile, including the delimiters, to be used when Company X sends a message or transaction to a trading partner not defined in the profile. This trading partner profile will also be the default trading partner profile used when Company X receives a message or transaction from a trading partner not defined in the profile. So the absolute minimum number of trading partner profiles that could ever be defined is one. That one profile, the default trading partner profile, would be used for all trading partners.

The name of the default trading partner profile is ANY because it can be used for any trading partner that does not have a trading partner profile of its own. The ANY trading partner profile is shipped with WebSphere Data Interchange and does not have to be created, although it should be updated to reflect the preferences in your environment.

If one of Company X's trading partners (Company A) requires a set of delimiters different than the other 9,999 trading partners, Company X must create a trading partner profile for Company A specifying the unique delimiters to be used. Company X does not have to create and maintain trading partner profiles for the other 9,999 trading partners because they are willing to accept the delimiters defined in the default trading partner Profile. When Company X sends to Company A, the profile created for Company A will override the set of delimiters in the default trading partner Profile, which is used for the other 9,999 trading partners.

With the possibility that some trading partners will appear in the trading partner Profile and some will not, it is necessary to classify trading partners with respect to whether they appear in the trading partner Profile. In WebSphere Data Interchange, trading partners are considered to be either KNOWN or UNKNOWN. A known trading partner is one that has a trading partner profile. Referring again to the scenario, when Company A was included in the trading partner profile, it became known to WebSphere Data Interchange. The other 9,999 trading partners remain unknown to WebSphere Data Interchange. The set of all trading partners, whether

Minimal trading partners

known or unknown, is referred to in WebSphere Data Interchange as ANY trading partner. These classifications become keywords in WebSphere Data Interchange that can be used to refer to these classes of trading partners in messages, rules, and usages.

Understanding generic rules (usages)

An alternative to using processes to reduce the number of rules you must create and maintain is to use generic rules (or usages). Generic rules are particularly useful for reducing the number of rules when you are using the minimal trading partners feature since you cannot use processes with it. Generic rules allow you to use the keywords ANY and KNOWN in place of a trading partner name on a trading partner-based rule. These rules have the disadvantage in that you cannot create arbitrary classes of trading partners like NON-PROD-PROCESS-V1 for the trading partners associated with the first version of your non-production purchasing process. Trading partners are simply either KNOWN or UNKNOWN, and rules can be created that apply to either specific trading partners, all KNOWN trading partners (the ones in the trading partner Profile), or ANY trading partner (all trading partners, both KNOWN and UNKNOWN). The reason you cannot use processes with the Minimal trading partners feature is because when an unknown trading partner sends you a message, there is no way for WebSphere Data Interchange to know what process that trading partner is associated with because WebSphere Data Interchange knows nothing about the trading partner.

Trading partner-based rules relate trading partners to maps. At the most fundamental level, a rule is the combination of a map name, a sending trading partner name, and a receiving trading partner name. In addition to the fundamentals, rules allow you to specify a set of translation options that apply to the relationship. An example of a translation option is Acceptable Error Level, which affects whether the translator accepts or rejects a given message.

To continue the scenario of Company X and its 10,000 trading partners, consider how many usages you must create to link those trading partners to a single map. Since you know that Company A required a set of delimiters different than the other 9,999 trading partners, you must include Company B and its unique delimiters in the trading partner profile. So far, because there is no difference in the translation options to be used for any of the 10,000 trading partners, only one usage would be required for all. In words, the usage relationship required would read: When Company X sends this type of data to ANY trading partner (either the KNOWN trading partner Company A in the trading partner profile or the 9,999 unknown trading partners not in the trading partner profile), use this map to translate the data. You would code this in the send usage by specifying ANY as the sending trading partner and the keyword ANY for the receiving trading partner. This type of usage is known in WebSphere Data Interchange as a generic-generic rule because both trading partners are generic (the keyword ANY).

Suppose yet another company (Company B) requires an Acceptable Error Level that is different than the other 9,999 trading partners. To be able to refer to Company B, you must add Company B to the trading partner profile even though all their trading partner profile options are from the default options. Finally, you must add a usage and specify the Acceptable Error Level to be used when Company X sends to Company B. To code this in the send usage, specify ANY as the sending trading partner and Company B as the receiving trading partner. This type of usage is known in WebSphere Data Interchange as a generic-specific usage, because the sending trading partner is generic (the keyword ANY) while the receiving trading partner (Company B) is specific. You now have two usages that specify two

different sets of translation options that could be used when Company A sends to Company C: the anybody-to-anybody usage and the anybody-to-Company C usage. To determine which one to use, the translator arranges all candidate usages into a hierarchy and uses the one that most applies; in this case, anybody to Company C.

Translator hierarchy

The hierarchy used by the translator has four general classes of combinations, listed in order of precedence.

*   Specific sending trading partner, specific receiving trading partner (specific-specific)

    WebSphere Data Interchange translates this as meaning: Use this map when this sender trades this document with this receiver.
*   Generic sending trading partner, specific receiving trading partner (generic-specific)

    WebSphere Data Interchange translates this as meaning: Use this map whenever any known or unknown sending trading partner trades this document with this specific receiving trading partner.
*   Specific sending trading partner, generic receiving trading partner (specific-generic)

    WebSphere Data Interchange translates this as meaning: Use this map whenever this sending trading partner trades this document with any known or unknown trading partner.

    Note: The external trading partner always takes precedence over the internal trading partner, so which of the two that takes precedence depends on whether the external trading partner (Company A in the previous examples) is the sender or the receiver of the document.
*   Generic sending trading partner, generic receiving trading partner (generic-generic)

    WebSphere Data Interchange translates this as meaning: Use this map whenever any known or unknown trading partner trades this document with any other known or unknown trading partner.

Determining the trading partner when receiving

It is important to understand the search sequence used when an interchange is parsed to determine which trading partner is going to receive this interchange (during the send process), or which trading partner has sent this interchange (during the deenvelope process). WebSphere Data Interchange uses the following steps to identify the trading partner:

1. Using the qualifier and ID from the interchange, WebSphere Data Interchange searches the trading partner profile members for a match in the **Interchange qualifier** and **Interchange ID** fields.
2. If the interchange ID is 15 or fewer bytes, WebSphere Data Interchange parses the interchange ID into a 7-byte account number and an 8-byte user ID, and then searches the trading partner profile members for a match in the **Account number** and **User ID** fields.
3. If the interchange ID is 16 or fewer bytes, WebSphere Data Interchange parses the interchange ID into an 8-byte account number and an 8-byte user ID, and then searches the trading partner profile members for a match in the **Account number** and **User ID** fields.

4. If the interchange ID is 35 or fewer bytes, WebSphere Data Interchange parses the interchange ID into a 32-byte account number and a 3-byte user ID, and then searches the trading partner profile members for a match in the **Account number** and **User ID** fields.

5. WebSphere Data Interchange parses the interchange ID into an account number and a user ID separated by either blanks, periods, or slashes, and then searches the trading partner profile members for a match in the **Account number** and **User ID** fields.

6. Using the interchange ID, WebSphere Data Interchange searches the trading partner profile members for a match in the **Contact phone** field.

Provide a value in the **Interchange qualifier** and **Interchange ID** fields of each trading partner profile member for the best response during any send or receive process.

Note: With 0 enveloping (no interchange, or send rule envelope type = 0, or ISA06 is blank), WebSphere Data Interchange uses the trading partner nickname.

Determining the interchange ID and qualifier when sending

During send map processing, the following search sequence is used in an attempt to locate a member in the trading partner profile (TPPROF) for the sender:

1. If the application **TP nickname** field names a field and the field contains a value, or if the **APPLTID** field of the C record contains a value, then WebSphere Data Interchange uses that value as the name of the trading partner profile member for the sender.

2. When neither the **TP nickname** field nor the **APPLTID** field contains a value, then:
   - If the Interchange Sender ID and ID qualifier fields name a field and the field contains a value, OR
   - If the ISID and ISIDQUAL fields in the C&D records contain values

   These values are used to search the trading partner profile members to find an entry with a matching Interchange ID and ID Qualifier. WebSphere Data Interchange uses the value in the associated TP nickname field as the trading partner profile member for the sender.

At this point, if no matching entry is found, the sending trading partner is considered to be unknown and WebSphere Data Interchange uses the default trading partner profile (with the nickname of ANY) as the trading partner profile member for the sender. Refer to "Minimal trading partners scenario" on page 135 for more information.

WebSphere Data Interchange uses the following search sequence to locate a member in the trading partner profile (TPPROF) for the sender.

1. The sender ID qualifier and sender ID are used to search the trading partner profile members (in TPPROF) to find an entry with a matching Interchange Qualifier and Interchange ID. If a matching entry is found, the value in the **TP nickname** field is used as the trading partner profile for the sender.

2. If no match is found, the trading partner sender ID is parsed as follows:

3. If the trading partner sender ID is 15 or fewer bytes in length, the interchange ID is parsed into a 7-byte account number and an 8-byte user ID.

4. If the sender ID is 16 bytes in length, the interchange ID is parsed into a 8-byte account number and an 8-byte user ID.

5. If the sender ID is 35 bytes in length, the interchange ID is parsed into a 32-byte account number and a 3-byte user ID.

6. If the sender ID contains separators (such as blanks, periods, or slashes), the value is parsed into an account number and user ID based on the separators found.

Based on the parsed value, the trading partner profile members are searched for a matching **Account number** and **User ID**.

1. If no match is found, the interchange ID is used to search the trading partner profile members for a matching **Contact Phone**.

At this point, if no matching entry is found, the sending trading partner is considered to be unknown and the default trading partner profile (with the nickname of ANY) is used as the trading partner profile member for the sender.

When the trading partner profile for the sender has been determined, the same search sequence is performed using the Interchange Receive ID and ID Qualifier values in the data to attempt to locate the trading partner profile nickname for the receiver.

Handling a trading partner in multiple environments

You might need to translate and send data to one trading partner from both z/OS and CICS. For example, the capability to translate and send from both environments is very helpful when one department processes data in z/OS and another in CICS, but both sets of data go to the same trading partner. Another case is when data is processed in CICS, but is switched to z/OS in emergency situations. WebSphere Data Interchange provides a method for performing all functions in either environment for the same trading partner.

When defining the trading partner profile member, you must choose a *primary* and a *secondary* environment. The primary environment is determined by the value in the **Network ID** field of the trading partner profile member. To use CICS as the primary environment, enter IINCICS in the **Network ID** field. To use z/OS as the primary environment, enter one of the following in the **Network ID** field:

- IINB41
- IINB42

The environment you specified in the **Network ID** field becomes the primary environment, the other becomes the secondary environment. For example, if you specified a CICS network ID as the primary environment, z/OS becomes the secondary environment.

In the primary environment, you can use all functions as documented in the *WebSphere Data Interchange Programmer's Reference*. All functions can also be performed in the secondary environment, except for the following WebSphere Data Interchange Utility combination functions:

- PERFORM TRANSLATE AND SEND
- PERFORM ENVELOPE AND SEND
- PERFORM REENVELOPE AND SEND

Handling a trading partner in multiple environments

In the secondary environment, enveloping and sending functions must be split apart when using the WebSphere Data Interchange Utility. During these combination-send functions, the network IDs to be sent are retrieved from the applicable trading partner profile members, NOT from the mailbox (requestor) profile members. If the combination-send function is attempted in the secondary environment and a trading partner profile member contains a primary environment network ID, that network ID is used. The send will fail because a CICS network program cannot be executed in z/OS, and an z/OS network program cannot be executed in a CICS environment.

To solve this problem, the translation and send functions must be executed separately. In the following example, the primary environment is z/OS, and the secondary environment is CICS. The application data in transient data queue IN01 is destined for trading partners whose trading partner profiles specify IINB41 as the network ID. The file ID override tells WebSphere Data Interchange to write all envelopes to a temporary storage queue named EDIQDAT. EDIQDAT is the default name of the envelope TS queue supplied with network profile IINCICS. Mailbox profile IINCICSREQ has IINCICS as the network ID. This example would be executed in CICS, the secondary environment.

```
PERFORM TRANSLATE AND ENVELOPE
WHERE APPFILE(IN01) APPTYPE(TD) FILEID(EDIQDAT)
PERFORM SEND WHERE REQID(IINCICSREQ)
```

Because the translation and sending functions are split apart, WebSphere Data Interchange is not concerned with networks affected by the enveloping function. Data is written to TS queue EDIQDAT, and the separate send function instructs WebSphere Data Interchange to send the data using the IINCICS network profile.

Specifying usages and rules

You can view and create send usages, receive usages, and data transformation map rules through the Trading Partner List window.

Viewing usages and rules

1. In the trading partner List window, click the trading partner for which you want to view usages or rules.
2. Click View Usages on the tool bar.

   WebSphere Data Interchange Client runs a query that displays the Usages and Rules List window, which contains three tabs. Click the Send Map Usage tab to display a list of Send Usages associated with the trading partner. Click the Receive Map Usage tab to display a list of Receive Usages associated with the trading partner. Click the Data Transformation Map Rule tab to display rules associated with trading partner.

Creating usages and rules

Create a usage or rule when you need to create a new association between a map and a trading partner.

1. In the trading partner List window, click the trading partner for which you want to create a usage or rule.
2. Click View Usages on the tool bar.

   The Usages and Rules List window displays.
   Send map usage

   a. Click the Send Map Usage tab.

   b. Click New on the toolbar.

      The Send Map Usage Editor window displays with the General tab open and all the fields blank.

   c. Complete the fields on the General tab. Required fields are preceded by a blue asterisk.

      Click

      for field descriptions.

   d. The fields on the Exit Routines tab, Envelope Attributes tab, and the DI Options tab are optional.

      Click

      for field descriptions.

Receive map usage

   a. Click the Receive Map Usage tab.

   b. Click New on the toolbar.

      The Receive Map Usage Editor window displays with the General tab open and all the fields blank.

   c. Complete the fields on the General tab. Required fields are preceded by a blue asterisk.

      Click

      for field descriptions.

   d. The fields on the Attributes tab and the DI Options tab are optional.

Data Transformation map rule

   a. Click the Data Transformation map rule tab.

   b. Click New on the toolbar.

Usages and rules: creating

The Data Transformation Map Rule Editor window displays with the General tab open and all the fields blank.
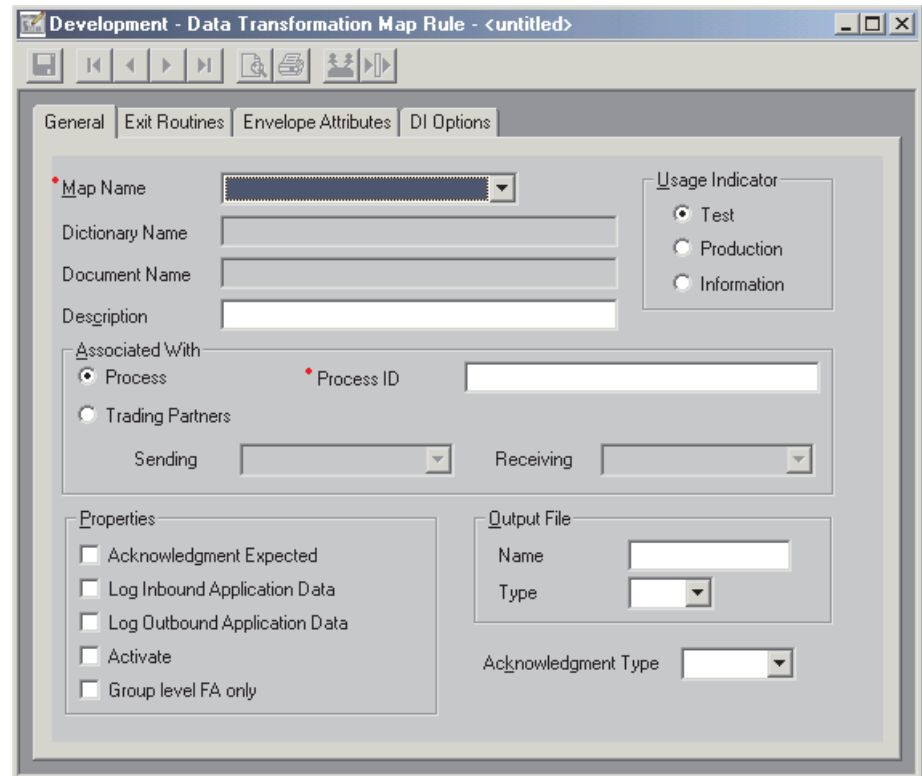
c. Complete the fields on the General tab. Required fields are preceded by a blue asterisk.

Click for field descriptions.

d. The fields on the Exit Routines tab, Envelope Attributes tab, and the DI Options tab are optional.

3. When you have finished entering all values required in the usage or rule, click Save on the tool bar to save the usage or rule.

Creating contacts

WebSphere Data Interchange Client uses Contacts to maintain information on the various contacts for your trading partners. For example, a contact can be the person who manages the business for a trading partner, an analyst at the trading partner's company, or a third-party administrator that manages operations on behalf of a trading partner.

Although it is useful to associate contacts with particular trading partners, it is not necessary. If you like, you can use the Contacts tab as your WebSphere Data Interchange address book to store the names and addresses of all of your WebSphere Data Interchange contacts, regardless of their association with trading partners.

Note that contacts have no role in WebSphere Data Interchange's translation functions. They are provided in WebSphere Data Interchange Client as a convenience.

You set up and maintain information on your trading partner contacts in the same way you set up and maintain information on trading partners. To view a Contact, double-click the appropriate row in the Contacts tab to display the Contact Editor window.

The Contact Editor window contains two tabs, each of which displays information only about the contact selected from the contacts list. You enter information about contacts by clicking the tabs and filling in the fields. WebSphere Data Interchange Client places the information in the Contact list. For information on viewing, copying, editing, renaming, deleting, and printing Contacts, see "Performing common file management tasks" on page 45 For information on exporting Contacts, see "Exporting" on page 68

Note: The Contact List window displays all contacts that satisfy your query. The Contacts tab within the trading partner Editor window displays only those contacts associated with that particular trading partner.

Adding a contact

1. Click trading partner on the WebSphere Data Interchange Client Navigator bar.

    The trading partners List window displays.
2. Click the Contacts tab if it is not already in front.

    The Contacts List window displays, displaying a listing of existing contacts.
3. Click New on the tool bar.

    The Contacts Editor window displays, with the General tab displayed.
4. Type in a Name, the only required field, and any other information you want to maintain on the contact.
5. Click the Comments tab, and type any comments you have about the selected contact into the Comments field.
6. Click Save on the tool bar.

    WebSphere Data Interchange Client saves the new contact.

Requesting, sending, and receiving functional acknowledgments

For some transactions you send, you might want the receiver to reply, and acknowledge receipt of the transaction. The reply is called a functional acknowledgment. Typically, it conveys information about any syntactical errors found, and acceptance or rejection of the message.

The contents of the functional acknowledgement are defined by the X12 997 or UCS 999 transaction set, or by the EDIFACT CONTRL message. This section only applies for those who use the X12, UCS, or EDIFACT standards.

This section explains the steps the requestor and the sender must take to exchange functional acknowledgments, and explains how the exchange occurs.

The steps you must take depend on whether you are using Data Transformation maps, or Send/Receive maps.

Requesting functional acknowledgments

To request functional acknowledgments, follow these steps:

1. Tell your trading partner the type of transaction for which you want to receive an acknowledgment.

2. For send and data transformation maps, in the rule for the map, set the Acknowledgment expected field to Y. This tells WebSphere Data Interchange to set ″pending functional acknowledgment″ status for these transactions when they are sent.

Sending functional acknowledgments for Data Transformation Maps

To generate functional acknowledgments for a data transformation map:

1. Click on the mapping icon in the navigation bar to open the Mapping Functional Area.
2. Select the Data Transformation Maps tab
3. Select the Data Transformation map to be acknowledged from the list window.
4. Click the View Usages Icons to view the Data Transformation Map Rules for the map you have selected.
5. To create a new rule, click the Create New Document icon.

   Alternatively, you can double-click on an existing rule to open the Data Transformation Map Rule window.
6. Clik the WDI Options tab.
7. Select a Functional Acknowledgment Map, Functional Acknowledgment Envelope Type and Functional Acknowledgment Profile Name from the drop-down lists on the WDI Options tab.

WDI is preloaded with maps for all the commonly used functional acknowledgments. The preloaded functional acknowledgment map names begin with $DT_FA. This is followed by the name of the functional acknowledgment message and the version and release of the message. For example, Version 3 Release 7 of the 997 functional acknowledgment message is named $DT_FA997V3R7.

You must specify the type of envelope to be used, and an envelope defaults profile to provide the default values for the component elements of the service segments used by the selected envelope. From the Envelope Attribute tab, select the type of envelope to be used from the Envelope Type drop-down list and select the envelope defaults profile from the Profile Name drop-down list.

Note: For data transformation maps, a default functional acknowledgment is generated when a message is received and no rule exists in WDI to determine how to process the message. This ensures all inbound EDI messages are acknowledged. The default functional acknowledgment specifies that the message was rejected. The reason for rejection is set to ″no partner agreement″ or something similar depending on the reason codes available for the particular kind of functional acknowledgment being created. The type of default acknowledgment returned is the acknowledgment defined by the standards agency responsible for the meta-data for the inbound message. For example, if an EDIFACT message is received, then a CONTRL functional acknowledgment will be generated.

Some acknowledgment messages support the concept of a ″group level functional acknowledgment″. In a group level functional acknowledgment, functional groups are acknowledged, rather than individual messages. It is up to the receiver of a group level functional acknowledgment to recall what individual messages were sent in the functional group being acknowledged and update their status accordingly. If the type of functional acknowledgment message you are using supports the ″group level only″ concept, then you can check the Group Level FA

Sending functional acknowledgments for Data Transformation Maps

Only box on the data transformation rule to instruct the functional acknowledgment map to generate acknowledgments for functional groups only.

Functional acknowledgments are normally written to the envelope file logical name taken from the network profile associated with the trading partner to whom the functional acknowledgment is being sent. You can overridde this by including the FUNACKFILE and optionally the ACKTYPE keywords on the PERFORM TRANSFORM command. If you do not use the FUNACKFILE keyword and there is no value for the Envelope File field in the network profile, functional acknowledgments are written to the logical name QDATA. For more information on the Envelope File field of the Network Profile see Chapter 15, "Network profiles" on page 103. For more information on the FUNACKFILE and ACKTYPE keywords, see the *WebSphere Data Interchange Programmer's Reference*.

The logical name to which the functional acknowledgments are written must be associated with a physical name at runtime. You must make an appropriate association for your platform and method of invoking WebSphere Data Interchange. For example if you are triggering WebSphere MQ queues using the WebSphere Data Interchange Adpater for WebSphere MQ and Service Profiles, and you want to send the functional acknowledgments to the envelope file ″ABC″ named in the Network Profie, you should associate the logical name ″ABC″ with a physical file name on the Output Files tab of the appropriate Service Profile. For more information on Service Profiles, see Chapter 20, "Service profiles" on page 121. For more information on triggering WebSphere Data Interchange WebSphere MQ queues using the WebSphere Data Interchange Adapter for WebSphere MQ, see the *WebSphere Data Interchange Programmer's Reference*.

To write functional acknowledgments to a file, you must invoke another command to send the acknowledgments. If you want to send the functional acknowledgments immediately and you are using command chaining, you can chain to a PERFORM SEND command to invoke send processing. Otherwise you can stack (submit multiple concatenated commands) the PERFORM SEND command after the PERFORM TRANSFORM command, or you can submit a PERFORM SEND command immediately after the PERFORM TRANSFORM command completes. For more information on command chaining, see Chapter 20, "Service profiles" on page 121.

Receiving functional acknowledgments for Data Transformation maps

Functional acknowledgments that are received and processed using the PERFORM TRANSFORM command are processed like any other message. If there is a rule to process them, then whatever the rule specifies is executed. The functional acknowledgment can be mapped to messages that your applications can understand and the target message routed to all the applications that need to know that a message that was sent has been acknowledged by the receiver.

Sending functional acknowledgments for receive maps

Note: For a receive map, a receive usage must exist for a transaction that is deenveloped before a functional acknowledgment is created.

To generate functional acknowledgments for a receive map:

1. Click the Mapping icon in the Navigation bar to open the Mapping Functional Area

2. Select the Receive Maps tab

Receiving functional acknowledgments for Data Transformation maps

3. Select the Receive Map to be acknowledged

4. Click the View Usages icon in the navigation bar to open the Receive Map Usage list window.

5. To create a new usage, click the Create New Document icon.

   Alternatively, double-click on an existing usage to open the Receive Map Usage window.

6. Enter an Acknowledgment type from the drop-down list.

   The acknowledgment type is one of the following: 997, 997V35, 999, CONTRL, or CONTV21. This informs the translator to build the functional acknowledgment.

   To generate a 997 functional acknowledgment, enter either 997V35 for Version 3 Release 5 and later, or 997 for the version and release prior to Version 3 Release 5.

   To generate a CONTRL functional acknowledgment, enter either CONTV21 for Version 2 Release 1 and later, or CONTRL for the version and release prior to Version 2 Release 1.

7. Specify an envelope defaults proflie by selecting one from the Envelope Profile drop-down list.

By default, functional acknowledgments are written to the logical name QDATA (for 997), QDATAU (for 999), or QDATAE (for CONTRL). You can change the name by updating the network profile member in which it appears or by including the FUNACKFILE keyword in the deenveloping request. The FUNACKFILE keyword overrides the name of the transaction data queue in the network profile. For more information, see the *WebSphere Data Interchange Programmer's Reference*.

On z/OS, the JCL for the job that receives and translates the transaction to be acknowledged must contain a DD statement with the name of the file. For example:

```
//QDATA DD DSN=IIN.TRANS,DISP=MOD
```

On the distributed platforms you should include a set file statement with the name of the file. For example:

```
set file(QDATA,/usr/local/b2b/funcaks/purchasing.fak)
```

If you are using command chaining and Service Profiles, you can associate QDATA with a filename on the Output Files tab of a Service Profile.

You should remember the following points when setting up your functional acknowledgments:

- Use the enveloping options file (FAENV) if you need to override values placed in the envelope segments. See the *WebSphere Data Interchange Programmer's Reference* for details.

- Use the Inbound envelope on FA flag to indicate the use of the inbound envelope information for enveloping the Functional Acknowledgment.

- Send the acknowledgment, along with any other transactions in the file, to the trading partner. WebSphere Data Interchange does not automatically send the file of acknowledgments. You must issue the send command separately.

Receiving functional acknowledgments for send maps

The trading partner who requested the acknowledgment receives the 997, 999, or CONTRL, along with other transactions that are in the same envelope. The translator translates all the transactions and puts them in the application file. (See

the note below for an exception.) It also logs the appropriate transaction images, depending on user choices and on any errors that may occur. Transaction Store pairs the functional acknowledgment with the transaction that requested it and updates the status of that transaction.

Notes:

1. WebSphere Data Interchange uses the functional acknowledgment to update the status information.

2. WebSphere Data Interchange interrogates the version and release in the UNH segment of the received EDIFACT CONTRL functional acknowledgment to determine the format of the received CONTRL message. The version and release must be equal to or greater than Version 2 Release 1 in order for WebSphere Data Interchange to accurately interpret the CONTRL message and reconcile it with messages awaiting the acknowledgments. Any version prior to 2 is interpreted using the older version of the CONTRL message. You should notify all your trading partners who send the new format CONTRL message that the UNH must contain the correct version and release.

If you do not want to envelope functional acknowledgments immediately, use the FADELAY keyword on the deenveloping request. The values for this keyword are:

Y   Indicates that functional acknowledgments are to be put in the Transaction Store. They will be enveloped and sent at a later time.

N   Indicates that functional acknowledgments are to be enveloped and written to the envelope file immediately. You need a separate command to send them. The enveloped functional acknowledgments are written to one of the following logical names in order of precedence:

   1. The file specified by the FUNACKFILE keyword, if present

   2. The transaction data queue name from the network profile (second choice)

   3. QDATA, QDATAU, or QDATAE (depending on the envelope type) by default

Interchange acknowledgments (TA1 segment)

WebSphere Data Interchange ignores the TA1 segment within an X12 envelope being deenveloped and does not create the TA1 segment in X12 envelopes when creating the envelope.

Note: You can use functional acknowledgments rather than interchange acknowledgments to ensure your trading partners get your transactions. If you receive X12 envelopes with TA1 segments and no groups or transactions, you may be left with orphan interchange envelopes in the Transaction Store. This condition only happens if envelope level recovery is used. To remove orphan interchange envelopes in the Transaction Store, use the following SQL statement:

```
DELETE FROM EDIENU32.EDITSEV WHERE SUBSTR(TRLIMAGE,1,6)=IEA "0 ";
```

Interchange acknowledgments

Part 4. Mapping

149

Chapter 22. Data formats

The term "data format" defines the layout of your application data. It is a document definition. The word "data", of course, refers to the information itself. The word "format" refers to the physical layout of information in the file, such as field names and lengths.

WebSphere Data Interchange needs a description of the data format for each business application that generates data that will be translated, or uses data that has been translated. An application's data must be described to WebSphere Data Interchange so that it can be used as a source or target for translation.

WebSphere Data Interchange Client's Data Format editor allows you to describe your application's data to WebSphere Data Interchange. When you have created a data format, you then create a map between the application's data format and the source or target document. Creating a data format, then, is the first step in the mapping process if your map uses or creates data to be used by your application.

You usually need to create a data format for each different business document that is used or created by WebSphere Data Interchange. A single data format can be mapped to multiple documents.

If you use EDI or XML to exchange invoices, for example, you need to create a data format for your invoicing system so that WebSphere Data Interchange understands how your invoicing system structures an invoice. From that data format, you create a map so you can translate your application's data to the 810 transaction in the X12 standard, or the XML format that your trading partner uses. For details on mapping, Chapter 25, "Creating a map" on page 197.

This chapter assumes that you know the layout of the application data you are using.

Creating a data format

This section provides the basic steps you follow to create a data format for an application. The broad steps are:

1. Understand how your application data is structured.

   Get a copy of your application's record layout and study its format.

2. Fill out a data format work sheet.

   This form helps you to enter information on your data format into WebSphere Data Interchange Client's data format editors. Refer to the sample on page 156 and the worksheet example on page 157.

3. Use the WebSphere Data Interchange Client data format editors to create the data format.

   WebSphere Data Interchange Client has data format editors for Data Format Dictionaries, Record ID Info, Data Formats, Loops, Records, Structures, and Fields. See "Use the WebSphere Data Interchange Client data format editors" on page 159.

Each step is detailed in the sections that follow.

Understanding how your application data is structured

Understanding how your application data is structured requires three steps:

1. Obtain a useful copy of your application data layout.
2. Optionally, structure that data so that WebSphere Data Interchange Client can use it.
3. Determine the data components used by your application.

Obtain application data layout

First, you need to obtain a copy of each application's record layout. The record layout can come directly from the program code listings or any other documentation that shows the beginning and ending position of each field in the record. For each record, the information should show:

- Physical attributes of each field
- Content of each field
- Position of each field in the record
- Length of each field
- Data type of each field
- Relationship between the records

Many users will be able to obtain COBOL copybooks that specify that information for their applications.

Structure application data

The next step in understanding your application data is structuring application data in a format that WebSphere Data Interchange Client will accept. WebSphere Data Interchange can accept two types of data record formats: *raw data records* and *control and data records*, as described below.

- Raw data:

  Each record in raw data format identifies itself by containing a unique record identifier (a record ID). Raw data can be either comma-separated or fixed-position. For fixed-position data, the identifier starts in the same position and extends for the same length in each record. In comma-separated data, each value is separated by a comma. Comma-separated data is used only with data transformation maps. Raw data that is fixed-position can be used in data transformation maps, send maps and receive maps. The Record ID is actually a field in the record. As long as records contain identifiable record IDs, you can use application data without modification.

  In the illustration below, which is an example of fixed-position data, several records of application data contain the record ID in the first three positions. In this example, HDR is the record ID of the header record, NAM is the name, and so on.

Table 15. Application data with record identifier

```
HDR0123456 092091C321
NAMSmithson, Patricia Jeanne
SSN5555555555
PRVCity Regional Clinic
PID05050505-X505
ADR555 Cedar Road
ADRAny City IL
ADR61001-1101
TOT1555.00
```

When your data has no record identifier for WebSphere Data Interchange to associate with each record, you have two choices. You can modify your application data to contain a record identifier or you can modify it to use control (C) and data (D) records.

- Control and Data (C and D) Format:

  If no record ID clearly specifies the type of information contained in a record, that record structure can be indicated by using control and data records. Also use C and D records when you need to use multiple data formats in a single file or you need to use overrides offered in the C record that are not offered in raw data. You also can use this type of format to override fields within service segments (such as ISA, GS, UNB, and UNH).

  C and D format is only supported by send and receive maps.

  There are a number of ways to structure C and D records. Table 16 on page 154, for example, shows an example of a C and D record in which:

  - A C or D is in the first column (byte)
  - Record name is in the next 16 columns
  - Application data starts in column 18

  This example also shows the use of a Z record to indicate the end of the document. For more about Z records, see the *WebSphere Data Interchange Programmer's Reference*.

Creating a data format

Table 16. Sample Application Records with C and D Records

```
          1         2         3         4         5         6
123456789012345678901234567890123456789012345678901234567890
CSPSTT16 AFTSU09   IL
DPOHDR       P0123456 092091C321
DP0NOTE      INCOMPLETE INVOICE INFORMATION SLOWS REIMBURSEMENT
DP0NOTE      PATIENTS WITH MULTIPLE CLAIMS NEED COMPLETE HISTORY
DINVITEM     005500550055
DITEMDESC    SURGICAL PROCEDURE
DITEMDIAGN   CARPAL TUNNEL PAIN
DINVITEM     005500550055
DITEMDESC    SURGICAL PROCEDURE
DITEMDIAGN   LIGAMENT INFLAMMATION
DNAME        SMITHSON, PATRICIA JEANNE
Z
CSPSTT16 AFTSU09   IL
```

Note: For more information on converting data to C and D records, see the *WebSphere Data Interchange for z/OS Administration Guide*.

Determine application data components

You must determine the data components used by your application so that WebSphere Data Interchange can identify the data and accurately pass it to the translator. A "data component" is a grouping of related data fields, such as the field names that make up line items of a health-care claim or the ship-to address of a purchase order. WebSphere Data Interchange needs information on:

- Which fields in your application data form the various components WebSphere Data Interchange uses
- The order in which components occur
- The number of times each component occurs

That information allows WebSphere Data Interchange to identify the data you pass to the translator from your application. When you use the data format editors to create a data format for an application's data, you show relationships between the various components and show the number of times each component can occur in a transaction.

WebSphere Data Interchange Client uses four components to express the characteristics of the application's data for mapping:

- Fields
- Structures
- Records
- Loops

Each of these components is described below.

Fields: Fields are fundamental pieces of data, such as prices or item numbers or first names. Raw data that is fixed position can be used in data transformation maps, send maps and receive maps. In COBOL records, they are stored in a single variable.

Structures: A structure is a group of related data fields, which is probably unique to your company. When multiple fields always display together, you can designate the group as a structure and give it a structure name. Structures not only contain fields, but can also contain other structures.

For example, a purchase order line item contains price, quantity, and product ID fields. Those three fields always display together. Further, they are used not only on purchase orders but also on invoices. So you could create a structure called a line-item structure that consists of price, quantity, and product ID fields. A purchase order line item, then, can consist of a line item structure plus a requested ship date field, while an invoice line item can consist of a line item structure plus a requested payment date field. You may even repeat that structure within a purchase order line item record.

Note: Structures cannot be used for comma-separated data.

Records: A record is a set of related fields as they are defined in an application's data. Every record in your application must be defined in your data format, assuming you want to map all records. Records contain fields and structures. An example of a record that identifies a patient in a hospital's claims-management system is described in the following example.

You are a health care provider and you want to send your claim information in a single health-care claims EDI standard transaction to the insurer or payer of the claim. You submit claims each day for all patients with the same insurer or payer. You would look at your application's data and determine which records are required to send the data for a patient claim, say a patient record followed by claim line-item records. The patient is identified as R20, and the claim information is identified as R42 and R73, as shown in Table 17.

Table 17. Sample Record

```
R20  01623792001 ALBERT              MICKEY
XF07261925S31PO BOX 11

R42 0101623792001     CO19200002510{0005{21000000150{0001{155000002500000A

R42 0201623792001     PR1  00006100{0000{   00000000{0000{
  000000000000
R73 01623792001          48521PATIENT IS RECOVERING
FROM SURGERY AS NEEDED
R20 01623792001      LOUISE          MOONIE
XF07261925S31PO BOX 11

R42 0302623792001     OA1  00006100{0002{8900000100{0206{115000005000000E

R73 01623792001          48521PATIENT SHOULD RETURN
IN 2 MONTHS FOR LAB RESULT
```

In this example, there are multiple patient records and each patient record contains that patient's claim information. Because this group of records is related and repeats, you would create a loop consisting of the patient record and the repeating claim line-item records. Because loops can repeat, and the health care claims EDI standard transaction defines a repeating patient claim loop, you could send claims for multiple patients in a single health care claims EDI standard transaction to a single insurer or payer.

Loops: A loop is a group of records that repeat. Loops can occur within other loops; this is referred to as a nested loop. Loops are used for repeating such things as line items, as in the following example:

Data formats: loops

You want to include claims for multiple patients in a single health-care claims transaction. You would look at your application's data and determine which records are required to send the data for a patient claim, say a patient record followed by claim line-item records.

To pay multiple patients in the same transaction you would create a loop consisting of a patient record and a repeating claim line-item record. Because loops can repeat, you could send claims for multiple patients by repeating the patient claim loop for every patient.

When deciding how best to structure your application's data into data components, keep the following component hierarchy in mind:

- Loops can contain other loops or records.
- Records can contain structures or fields.
- Structures can contain other structures or fields.
- Fields are fundamental units of data.

   The example of a data format worksheet in the following section will help you to determine how to structure your application's data into data format components.

Filling out a data format worksheet

A data format worksheet can make it easier to decide how to structure application data. This example shows a purchase order record definition and how its data format worksheet looks.

Table 18. Sample Working Storage Record Definition of a Purchase Order

```
01 SHIPTO
   05   RECORD-IDENTIFIER
        10  PURCHASE-ORDER-NUMBER      PIC X(08).
        10  RECORD-ID                  PIC X(02).
   05   COMPANY-NAME                   PIC X(30).
   05   COMPANY-DUNS                   PIC X(10).
   05   COMPANY-ADDRESS
        10  STREET                     PIC X(30).
        10  CITY                       PIC X(15).
        10  STATE                      PIC X(02).
        10  ZIPCODE                    PIC 9(09).
   05   COMPANY-PHONE                  PIC 9(10).
01 DETAIL
   05   RECORD-IDENTIFIER
        10  PURCHASE-ORDER-NUMBER      PIC X(08).
        10  RECORD-ID                  PIC X(02).
   05   ITEM-NUMBER                    PIC X(10).
   05   ORDER-QUANTITY                 PIC 9(09).
   05   ORDER-UNITS                    PIC X(01).
   05   UNIT-PRICE                     PIC 9(09)V99.
   05   UNIT-DISCOUNT                  PIC 9(09)V99.
   05   EXTENSION                      PIC 9(09)V99.
*THERE ARE UP TO 3 DETAIL DESCRIPTION RECORDS PER DETAIL RECORD.
01 DETAIL-DESCRIPTION.
   05   RECORD-IDENTIFIER
        10  PURCHASE-ORDER-NUMBER      PIC X(08).
        10  RECORD-ID                  PIC X(02).
        10  DESCRIPTION
PIC X(30).
01 TOTAL
   05   RECORD-IDENTIFIER
        10  PURCHASE-ORDER-NUMBER      PIC X(08).
        10  RECORD-ID                  PIC X(02).
   05   TOTAL-AMOUNT                   PIC 9(09)V99.
```

Table 19. Data Format Component Relationship Worksheet Example

Parent Type	Parent Name	Child Type	Child Name	Max Use or Occurs	Record ID (REC Only)	Field Data Type (FIELD Only)	Field Length (FIELD Only)
data format	SAMPLE-PO	REC	SHIP-TO	1	SH		
		LOOP	DETAIL-LOOP	1000			
		REC	TOTAL	1	TT		
REC	SHIP-TO	STRUCT	RECORD-IDENTIFIER	1			

Data format worksheet

Table 19. Data Format Component Relationship Worksheet Example (continued)

Parent Type	Parent Name	Child Type	Child Name	Max Use or Occurs	Record ID (REC Only)	Field Data Type (FIELD Only)	Field Length (FIELD Only)
		FIELD	COMPANY-NAME			CH	30
		FIELD	COMPANY-DUNS			CH	10
		STRUCT	COMPANY-ADDRESS	1			
		FIELD	COMPANY-PHONE			NO	10
STRUCT	RECORD-IDENTIFIER	FIELD	PURCHASE-ORDER-NUMBER			CH	8
		FIELD	RECORD-ID			CH	2
STRUCT	COMPANY-ADDRESS	FIELD	STREET			CH	30
		FIELD	CITY			CH	15
		FIELD	STATE			CH	2
		FIELD	ZIPCODE			NO	9
LOOP	DETAIL-LOOP	REC	DETAIL	1	DE		
		REC	DETAIL-DESCRIPTION	3	DD		
REC	DETAIL	STRUCT	RECORD-IDENTIFIER	1			
		FIELD	ITEM-NUMBER			CH	10
		FIELD	ORDER-QUANTITY			NO	9
		FIELD	ORDER-UNITS			CH	1
		FIELD	UNIT-PRICE			N2	11
		FIELD	UNIT-DISCOUNT			N2	11
		FIELD	EXTENTION			N2	11
REC	DETAIL-DESCRIPTION	STRUCT	RECORD-IDENTIFIER	1			
		FIELD	DESCRIPTION			CH	30
REC	TOTAL	STRUCT	RECORD-IDENTIFIER	1			
		FIELD	TOTAL-AMOUNT			N2	11

Table 20. Data Format Component Relationship Worksheet

Parent Type	Parent Name	Child Type	Child Name	Max Use or Occurs	Record ID (REC Only)	Field Data Type (FIELD Only)	Field Length (FIELD Only)

Use the WebSphere Data Interchange Client data format editors

When you have filled out the data format worksheet, you are ready to create the data format on WebSphere Data Interchange Client. To create the data format and its components, use the data format editors, as described in this and following sections.

You use the data format editors to create the various components that make up a data format. In using the editors to create a data format, first decide whether to work from the top down or from the bottom up. If you work from the top down, after you create a data format dictionary, you create the larger components first, then

work down to the smaller. If you work from the bottom up, after you create a data format dictionary, you create smaller components first, and then work up to larger.

Whichever way you choose to work, you must create a data format dictionary as your first step. You cannot create components for a nonexistent dictionary and store them elsewhere until the dictionary is created. When you have created the dictionary, you can create the smaller components such as structures and fields before creating the larger components. In practice, it is easier to create the larger components, such as loops and records first, and then work down to the smaller components until your data format is complete.

This section
- Provides a generic description of the procedures for using the data format editors
- Steps through the process of using the editors to create a new data format.

Accessing data format editors

You use the data format list window to gain access to the data format component editors. Each tab contains a list of components. Click the tab corresponding to the component you wish to work with. From that list, you can select the specific component you wish to work with and open its editor window by clicking the component and then clicking Open.

To access a data format editor:

1. Click Data Format on the WebSphere Data Interchange Client Navigator bar.

   The Data Format List window, which contains tabs for the data format components, displays.

2. Click the tab of the data format component you wish to work with.

   The list window for that component displays.

   This window displays a list of existing components of the type you selected. Each row contains a single component; each column contains data stored in the component. Information in the columns displays in fields in the respective editor windows. The list window, however, also contains the date, time, and user ID of the last update.

   To display additional columns, click the scroll bar on the bottom of the screen to scroll to the right or left. To alter the columns that display on the screen, or to change which query is executed to produce the list, click Modify Window Properties (...). To create new queries, refer to "Creating a query" on page 268

3. To view an existing component or to add or change its information, double-click the row of the item you wish to work with.

   The editor window displays, with the General tab or details tab in front, depending on which component type you opened. You add information or make changes to the data format component through its tabs, as described in the following sections.

   Note: All of the seven component editors have a General tab and a Comments tab. The Data Format, Loops, Records, and Structures tabs also contain a Details tab, which allows you to set up the specifications for those components. The data formats editor has an Overview tab, which provides a visual representation of the entire data format.

Following are detailed procedures for creating data format components. For information on viewing, copying, editing, renaming, deleting, and printing data format components, see "Performing common file management tasks" on page 45

For information on using the grid editors that display in some data format editors, see "Using editor window grids" on page 47 For information on exporting data format components, see "Exporting" on page 68

The data format component editors are described in the following sections in the order in which you use them when creating a data format from scratch following a top-down approach.

Using the data format dictionary editor

A data format dictionary is a name for a group of other components. Following are detailed instructions for creating a new data format dictionary. For information on viewing, copying, editing, renaming, deleting, and printing data format dictionaries, see "Performing common file management tasks" on page 45 For information on exporting data format dictionaries, see "Exporting" on page 68

Creating a data format dictionary: Create a new data format dictionary when you set up your first data format for a particular application. You can then reuse data format components for that application when you create subsequent data formats if they are created within that dictionary. You can also import COBOL copybooks using the data format dictionary editor.

1. At the data format dictionary list window, click New Document on the tool bar.

   The data format dictionary editor window displays with the General tab in front and the fields blank.

2. Type a name in the Dictionary Name field. This is a required field, as indicated by the blue asterisk.

   The first and last characters can be A-Z and 0-9. All middle characters can be A-Z, 0-9, _, and -. You cannot type spaces within the name. The name can be 30 characters long.

   You can enter a more complete description of the data format dictionary in the Description field.

   Note: When you save the dictionary, the Data Formats, Loops, Records, Structures, and Fields buttons become available in the List Of group box. Click those buttons to display list windows that contain the components associated with this dictionary. When you first create a dictionary, the lists are empty.

3. Click Save on the tool bar to save the dictionary.

   After you have saved your dictionary, the Dictionary Name becomes read-only.

Importing a COBOL copybook: Importing allows you to take a COBOL source file and use it to create WebSphere Data Interchange fields, structures, and records, and then save them in this Data Format Dictionary.

1. Use the procedure described in "Using the data format dictionary editor" to create a dictionary to store the new records and fields. The dictionary must be saved before you proceed. After you save your dictionary, the Dictionary Name becomes read-only and the Import COBOL Copybook button is enabled.

2. Click Import COBOL Copybook.

   The Select COBOL Copybook File dialog box displays.

3. Enter the correct path and file name, and click Open.

4. Click Close when the status dialog box indicates the import is complete.

   The path and file name of the copybook file display in the Copybook File field, and the Record ID info drop-down list and New button are enabled.

Importing a COBOL copybook

5. Select an existing data format in this dictionary from the drop-down list, or click New.

    If a valid data format is selected, the Record ID Info and its New button are disabled. If you click New, the data formats editor displays. See "Using the data format editor" on page 164. Be sure you save the new data format before returning to the data format dictionary editor.

6. Select the existing record ID info object from the drop-down list, or click New. This is a mandatory field. The record ID info object provides information on how the record is identified to WebSphere Data Interchange.

    If you click New, the data format record ID information editor displays. See "Using the data format record ID information editor" on page 163. Be sure you save the new record ID information before returning to the data format dictionary editor.

7. Check the Convert 88s to Codelists box, if you want to convert 88 entries to code lists. One code list will be created for each set of 88-level statements that are associated with the COBOL data name. The names assigned to the code lists are determined by values in the Prefix and Starting Number fields. Checking this box enables the Prefix and Starting Number fields.

    *   **Prefix** - Defaults to the first five characters of the data format name, and is added when the check box is selected for the first time.
    *   **Starting Number** - Defaults at 1, has a range of 0 to 999.

8. Click Save on the tool bar to save the COBOL objects in the dictionary.

    Note: When you save the dictionary, the Data Formats, Loops, Records, Structures, and Fields buttons become available in the List Of group box. Click those buttons to display list windows that contain the components associated with this dictionary. When you first create a dictionary, the lists are empty.

9. Click Records to display a list of data format records belonging to the data format dictionary. You will need to set a record ID for each record added by the import COBOL copybook process.

    a. Edit each record by double-clicking it. This will open the data format record editor, which will display the selected record. Select the General tab and set the record ID for the record. See "Using the data format record editor" on page 166

    b. Click Save on the tool bar.

    c. Repeat for every record on the list.

    d. Close the list.

10. Back in the data format dictionary editor, click Data Formats to obtain a list of data formats belonging to the dictionary. You may need to update data format information.

    a. Edit each Data Format by double-clicking it. This will open the data format editor, which will display the selected data format. See "Use the WebSphere Data Interchange Client data format editors" on page 159.

    b. Click the Raw Data tab if you need to make changes to the raw data information.

    c. Click the Details tab to add loops that apply. See "Creating a loop" on page 165.

    d. Click Save on the tool bar.

    e. Repeat for every data format on the list that was added by the import COBOL copybook process.

    f.   Close the data format list window.

11.   Close the data format dictionary editor by selecting Close from the File menu.

Using the data format record ID information editor

The data format record ID information profile allows WebSphere Data Interchange to identify records in application data. The profile indicates whether the data format will use raw data format records or C and D format records. When your records are structured using raw data format, the data format record ID information profile also specifies the location and length of the Record ID.

Unlike data format dictionaries and other components, record ID information profiles are global for a system; they are not tied to a specific data format dictionary but can be used in any data format dictionary in the system. If your company structures all application record ID information in the same way or always uses C and D records, then you only need to create one record ID information profile for use with any data format.

Following are detailed instructions for creating a new data format record ID info profile. For information on viewing, copying, editing, renaming, deleting, and printing data format record ID info profiles, see "Performing common file management tasks" on page 45.

Creating a data format record ID information profile: Create a new data format record ID information profile when you set up your first data format or when you set up an data format whose record IDs are structured differently than your existing data formats. You can use the same data format record ID information profile for any data format whose record IDs have the same structure or that are set up as C and D records.

1.   At the data format record ID information list window, click New Document on the tool bar.

     The data format record ID information editor window displays with the General tab in front and the fields blank.

2.   Type a name in the Record ID Info Name field. This is a required field, as indicated by the blue asterisk.

     The first and last characters can be A-Z and 0-9. All middle characters can be A-Z, 0-9, _, and -. You cannot type spaces within the name. The name can be 30 characters long.

     You can enter a more complete description of the data format record ID information profile in the Description field.

3.   Indicate whether data formats that specify this data format record ID information profile will be use raw data format records or C and D format records.

    • If your application uses raw data format:

    a.   Type the position of the first character of the Record ID in the Position field.

    b.   Type the length of the Record ID in the Length field.

    c.   Select the Data Type from the drop-down list. Data Types are listed in Table 22 on page 174.

    • If your application uses C and D format:

      The Position, Length, and Data Type fields automatically fill with the correct values.

4.   Click Save on the tool bar to save the profile.

     After you have saved your record ID information, the Record ID Info and field becomes read-only.

Creating data format record ID information profile

To find other data format components that use the current data format record ID information profile:

1. Click Where Used.

   A list window displays containing Data Format, Data Format Loop, and Data Format Record tabs.

2. Click the tabs of each data format component to display a list of components containing the current Record ID Information profile.

   You may open any component by double-clicking it.

   An empty list window means that the data format record ID information profile is not used in any data format component of the list window's type.

Using the data format editor

The data format editor allows you to define and structure the components that make up a data format. A data format is a document definition. It defines the layout of your application's data. From the data format editor, you name the data format, identify properties related to the data format, and create and edit the associations to loops and records. It also gives you a visual of the data format from which you can directly navigate to each of its component editors.

The data format editor window contains five tabs: General, Details, Overview, Raw Data, and Comments.

- The General tab lets you set the name of the data format, select a dictionary and set properties of the data format.
- The Details tab lets you add or change loops and records associated with the selected data format and edit information about the association.
- The Overview tab lets you display a representation of the entire data format.
- The Raw Data tab lets you enter and change information specific to Raw Data Format translation.
- The Comments tab lets you type comments about the selected data format.

The editor window also contains a Where Used button, which will list all send and receive maps that use the current data format.

Following are detailed instructions for creating a new data format. For information on viewing, copying, editing, renaming, deleting, and printing data formats, see "Performing common file management tasks" on page 45 For information on exporting data formats, see "Exporting" on page 68

Creating a data format: Create a new data format when you need to define the new layout of application data that will be used in translation.

Using the data format loop editor

Use the Data Format Loop Editor window to enter new loops into a Data Format Dictionary or to edit existing loops. Loops are entered into a data format using the Data Format Editor when you create or edit a data format. From the Data Format Loop Editor, you can create and edit the loop's associations with loops and records, as loops can contain loops and records.

The Data Format Loop Editor window contains three tabs: General, Details, and Comments. Use the:

- General tab to name the loop, select its dictionary and set loop properties.
- Details tab to add or change loops and records associated with the selected loop and to edit the information about the association.

- Comments tab to type any comments you wish about the selected loop.

The editor window also contains a Where Used button, which allows you to find other data format components that use the currently selected loop.

Following are detailed instructions for creating a new loop. For information on viewing, copying, editing, renaming, deleting, and printing loops, see "Performing common file management tasks" on page 45

Creating a loop: Create a new loop when your application data has two or more records that are grouped together and each group can occur more than once.

1. At the data format loop list window, click New on the tool bar.

   The data format loop editor window displays with the General tab in front and the fields blank.

2. Type a name in the Loop Name field. This is a required field, as indicated by the blue asterisk.

   The first and last characters can be A-Z and 0-9. All middle characters can be A-Z, 0-9, _, and -. You cannot type spaces within the name. The name can be 30 characters long.

   You can enter a more complete description of the loop in the Description field.

3. Select the data format dictionary in which you want the loop to occur using the Dictionary Name drop-down list. This is a required field.

4. Select the record ID information profile you wish to use for this loop through the Record ID Info drop-down list. This is a required field.

5. Click the Details tab to enter information on the loops and records contained in this loop.

   a. From the Type drop-down list, select either Record or Loop, depending on which you are entering.

      Loops may contain both loops and records, but must begin with a record. The value of Max Use for the first record must be one (1).

   b. Either select the loop or record name from the Loop/Record Name drop-down list, or type in a loop or record name to create a new loop or record. The drop-down list displays loops and records that are in the same dictionary and use the same Record ID information profile as this loop.

      The remaining columns fill in with their default values.

      If you have not created any loops or records using their respective editors, this drop-down list will be empty. You may create a record or loop in this grid by typing in a name.

   c. Enter the maximum number of times the loop can be used in the Max Use column. If the loop or record can repeat infinitely either enter the number 32767 or click the Infinite check box to insert a check mark.

   d. For new records, enter the Record ID in the Record ID column. For existing records, WebSphere Data Interchange Client displays the Record ID as a read-only field. If you would like to edit the record ID, you can do so from the Data Format Record Editor window.

   e. For new records and loops, you may type in a more detailed description of the record or loop in the Description column. For existing records and loops, WebSphere Data Interchange Client displays the description as a read-only field. If you would like to edit the description you can do so from that particular loop or record's editor window.

      For information on how to use the editor window's grid, see "Using editor window grids" on page 47.

Creating a data format loop

6. When you have completed entering information, click Save on the tool bar to save the loop.

   After you have saved your loop, the Loop Name, Record ID Info, and Dictionary Name fields become read-only.

To find other data format components that use the current data format loop:

1. Click Where Used.

   A list window containing Data Format and Data Format Loop tabs displays.

2. Click the tabs of each data format component to display a list of components containing the current loop.

   You may open any item in the list by double-clicking it.

   An empty list window means that the loop is not used in any other data format component of the list window's type.

Using the data format record editor

Use the data format record editor window to enter new records into a data format dictionary or to edit existing records. From the data format record editor, you create and edit the record's associations with fields and structures, as records can contain fields and structures.

The data formats record editor window contains three tabs: General, Details, and Comments. Use the:

- General tab to name the record, select its dictionary, and set properties.
- Details tab to add or change fields and structures associated with the selected record and to edit the information about the association.
- Comments tab to type any comments you wish about the selected record.

The editor window also contains a Where Used button, which allows you to find other data format components that use the currently selected record.

Following are detailed instructions for creating a new record. For information on viewing, copying, editing, renaming, deleting, and printing records, see "Performing common file management tasks" on page 45.

Creating a record: Create a new record when your application data layout requires one. Records can contain fields and structures.

1. At the data format record list window, click New on the tool bar.

   The data format record window displays with the General tab in front and the fields blank.

2. Type a name in the Record Name field. This is a required field, as indicated by the blue asterisk.

   The first and last characters can be A-Z and 0-9. All middle characters can be A-Z, 0-9, _, and -. You cannot type spaces within the name. The name can be 30 characters long.

   If you wish, you may enter a more complete description of the record in the Description field.

3. Select the data format dictionary in which you want the record to occur using the Dictionary Name drop-down list. This is a required field.

4. Select the record ID information profile you wish to use for this record through the Record ID Info drop-down list. This is a required field.

5. Type the name of the Record ID in the Record ID field. This is a required field.

Note: If you are using C and D records, this field is not available. Records are identified by the record name.

6. Click the Details tab to enter information on the fields and structures contained in this record.

   a. From the Type drop-down list, select either Field or Structure, depending on which you are entering.

      Records may contain both fields and structures.

   b. Select either the field or structure name from the Structure/Field Name drop-down list, which displays fields and structures that are in the same dictionary.

      The remaining columns fill in with their default values.

      If you have not created any fields or structures using their respective editors, this drop-down list will be empty. You may create a field or structure in this grid by typing in a name.

   c. Enter the number of times the structure is used in the Occurs column. Fields are fixed at one (1). Alternately, you can indicate that the number of times a structure repeats is dictated by the value found in one of its peer fields. Do this by selecting a field from the Occurs Depending drop-down list.

      Occurs Depending is only supported for data transformation maps.

   d. For new fields, select the data type from the Data Type drop-down list. Type in the field length in the Field Len column. For definitions of WebSphere Data Interchange data types, see Table 22 on page 174. For existing fields, WebSphere Data Interchange Client displays the Data Type and Field Len fields as a read-only. If you want to edit these fields, you can do so from the data format field editor.

   e. For a new field or structure, you may type in a more detailed description of the field or structure in the Description column. For an existing field or structure, WebSphere Data Interchange Client displays the description as a read-only field. If you would like to edit the description you can do so from that particular field or structures' editor window.

      For information on how to use the editor window's grid, see "Using editor window grids" on page 47

7. When you have completed entering information, click Save on the tool bar to save the record.

   After you have saved your record, the Record Name, Record ID Info, and Dictionary Name fields become read-only.

To find other data format components that use the current data format record:

1. Click Where Used.

   A list window containing Data Format and Data Format Loop tabs displays.

2. Click the tabs of each data format component to display a list of components containing the current data format record.

   You can open any item in the list by double-clicking it.

   An empty list window means that the data format record is not used in any other data format component of the list window's type.

Using the data format structure editor

Use the data format structure editor window to enter new structures into a data format dictionary or to edit existing structures. From the data format structure editor, you create and edit the structure's associations with fields and structures, as structures can contain fields and structures.

Data format structure editor

The data formats structure editor window contains three tabs: General, Details, and Comments. Use the:

- General tab to name the structure, select its dictionary, and set properties.
- Details tab to add or change fields and structures associated with the selected structure and to edit the information about the association.
- Comments tab to type any comments you wish about the selected structure.

The editor window also contains a Where Used button, which allows you to find other data format components that use the currently selected structure.

Following are detailed instructions for creating a new structure. For information on viewing, copying, editing, renaming, deleting, and printing structures, see "Performing common file management tasks" on page 45.

Creating a structure: Create a new structure when your application data has two or more contiguous fields that should logically occur together. For example, a structure describing a date may contain fields for the year, month and day. In another example, Name, Address, City, State, and Zip occur three times in a single record in the same order. You can define those fields as a structure one time and say that it repeats three times in the record.

1. At the data format structure list window, click New on the tool bar.

   The data format structure editor window displays with the General tab in front and the fields blank.

2. Type a name in the Structure Name field. This is a required field, as indicated by the blue asterisk.

   The first and last characters can be A-Z and 0-9. All middle characters can be A-Z, 0-9, _, and -. You cannot type spaces within the name. The name can be 30 characters long.

   You can enter a more complete description of the structure in the Description field.

3. Select the data format dictionary in which you want the structure to occur using the Dictionary Name drop-down list. This is a required field.

4. Click the Details tab to enter information on the fields and structures contained in this structure.

   a. From the Type drop-down list, select either Structure or Field, depending on which you are entering.

   b. Select either the field or structure name from the Structure/Field Name drop-down list, which displays fields and structures that are in the same dictionary.

      The remaining columns fill in with their default values.

      If you have not created any fields or structures using their respective editors, this drop-down list will be empty. You may create a field or structure in this grid by typing in a name.

   c. Enter the number of times the structure is used in the Occurs column. Fields are fixed at one (1). Alternately, you can indicate that the number of times a structure repeats is dictated by the value found in one of its peer fields. Do this by selecting a field from the Occurs Depending drop-down list.

      Occurs Depending is only supported for data transformation maps.

   d. For new fields, select the data type from the Data Type drop-down list. Type in the field length in the Field Len column. For definitions of WebSphere Data Interchange data types, see Table 22 on page 174. For existing fields,

WebSphere Data Interchange Client displays the Data Type and Field Len fields as read only. If you want to edit those fields, you can do so from the Data Format Field Editor window.

e. For new fields and structures, you may type in a more detailed description of the field or structure in the Description column. For existing fields and structures, WebSphere Data Interchange Client displays the description as a read-only field. If you would like to edit the description, you can do so from that particular field or structure's editor window.

For information on how to use the editor window's grid, see "Using editor window grids" on page 47

5. When you have completed entering information, click Save on the tool bar to save the structure.

After you have saved your structure, the Structure Name and Dictionary Name fields become read-only.

To find other data format components that use the current structure:

1. Click Where Used.

A list window containing Data Format Record and Data Format Structure tabs displays.

2. Click the tabs of each data format component to display a list of components containing the current structure.

You may open any item in that component by double-clicking it.

An empty list window means that the structure is not used in any other data format component of the list window's type.

Using the data format field editor

Use the data format field editor window to enter new fields into a data format dictionary or to edit existing fields. From the data format field editor, you can set up and maintain properties of a data format field.

The data formats field editor window contains two tabs: General and Comments. Use the:

- General tab to enter the field name, select its dictionary, and set properties of the field.
- Comments tab to type any comments you wish about the selected field.

The editor window also contains a Where Used button, which allows you to find other data format components that use the currently selected field.

Following are detailed instructions for creating a new field. For information on viewing, copying, editing, renaming, deleting, and printing fields, see "Performing common file management tasks" on page 45.

Creating a field: Create a new field when your application data requires one.

1. At the data format field list window, click New Document on the tool bar.

The data format field editor window displays with the General tab in front and the fields blank.

2. Type a name in the Field Name field. This is a required field, as indicated by the blue asterisk.

The first and last characters can be A-Z and 0-9. All middle characters can be A-Z, 0-9, _, and -. You cannot type spaces within the name. The name can be 30 characters long.

Data format field editor

If you wish, you may enter a more complete description of the field in the Description field.

3. Select the data format dictionary in which you want the field to occur using the Dictionary Name drop-down list. This is a required field.

4. Select the data type from the Data Type drop-down list. For definitions of WebSphere Data Interchange data types, see Table 22 on page 174. This is a required field.

5. Type the length of the field in the Field Length field. This is a required field.

   If this field is required by a trading partner or the EDI standard but may not always contain a value, you can use WebSphere Data Interchange literals or mapping commands to enter information into the field.

   Note: This is supported in send and receive maps only. It is ignored in data transformation maps.

   a. Type the name of the literal you wish to use in the Literal field. For a list of WebSphere Data Interchange literals or mapping commands, search in WebSphere Data Interchange Client Help on the keyword "literals".

   b. Literals and mapping commands are validated differently by the translator depending on whether they are used by send maps or receive maps. If you want this value validated, click either or both the Send Map Validation check box or the Receive Map Validation check box, depending which map types the literal or mapping command will be used in.

6. You may associate code lists with fields to validate the data they contain against values in a specific list. Select the list you wish to validate this field against from the Code List drop-down list. This field is used only in send and receive maps.

   For information on creating User Code Lists, see "Using the code list editor" on page 194.

7. When you have completed entering information, click Save on the tool bar to save the field.

   After you have saved your field, the Field Name and Dictionary Name fields become read-only.

To find other data format components that use the current field:

1. Click Where Used.

   A list window containing Data Format Record and Data Format Structure tabs displays.

2. Click the tabs of each data format component to display a list of components containing the current field.

   You may open any item in the list by double-clicking it.

   An empty list window means that the field is not used in any other data format component of the list window's type.

Navigating data format component editors

WebSphere Data Interchange Client's data format editor windows are designed to provide maximum flexibility. You can move from editor to editor with ease so that you can tailor your navigation to the requirements of your work.

This section provides information on the various paths from one editor to the next.

Data format dictionary editor paths

The buttons at the bottom of the data format dictionary General tab generate list windows for each data format component that occurs in the current dictionary. From those list windows, you can open the editor for components in the list, where you can edit the current component or create a new one of that type.

Data Formats
> The data formats associated with this dictionary.

Loops The loops associated with this dictionary.

Records
> The records associated with this dictionary.

Structures
> The structures associated with this dictionary.

Fields The fields associated with this dictionary.

Data format editor paths

From the data format editor, you can move to any other component of a data format. The most powerful navigational tool in the data format editor window is its Overview tab, which displays a visual representation of your data format:
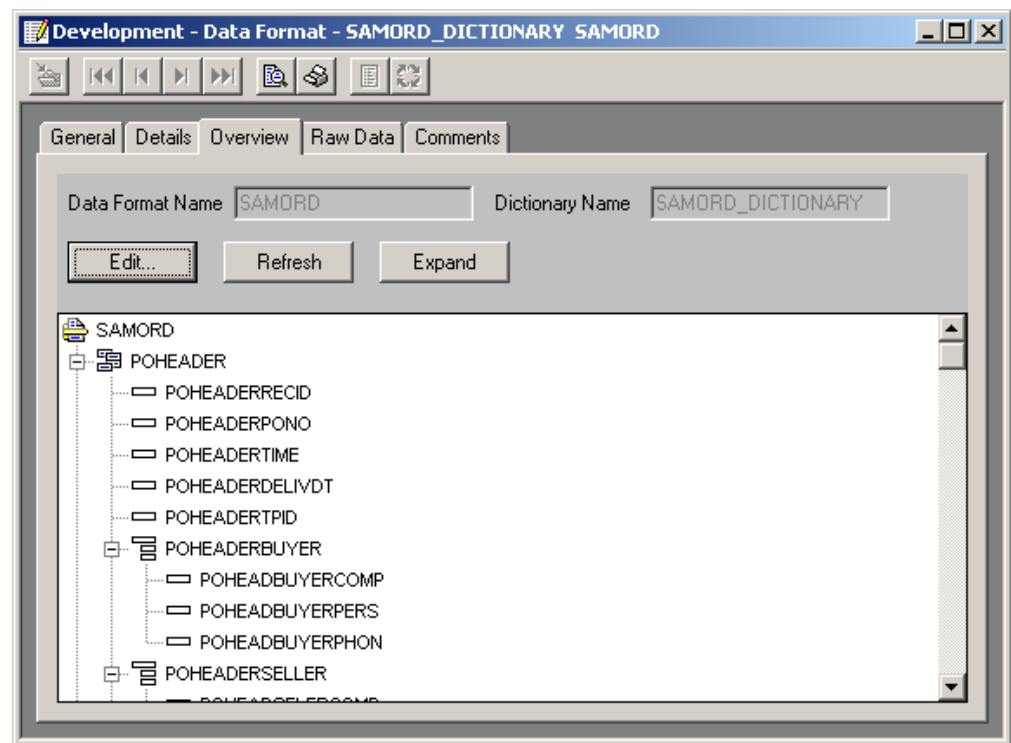
Table 21 defines the symbols on the data format editor Overview tab.

Table 21. Data format symbols

This symbol. . .	Represents:
	A loop
	A record

Navigating between editors

Table 21. Data format symbols (continued)

This symbol. . .	Represents:
	A structure
	A field

Click the + sign to expand a section of the data format.

Click the - sign to collapse that section of the data format.

The Expand button expands the node which is currently selected. To expand the entire tree, select the root node and click Expand.

The Overview tab allows you to navigate through the data format as follows:
- Double-clicking a loop starts the data format loop editor for the loop on which you clicked.
- Double-clicking a record starts the data format record editor for the record on which you clicked.
- Double-clicking a structure starts the data format structure editor for the structure on which you clicked.
- Double-clicking a field starts the data format field editor for the field on which you clicked.

From its Details tab, the data format editor window allows you to start the data format loop editor and the data format record editor.
- To start the data format loop editor, double-click the number of the row containing the loop you wish to edit. You may also select the row and then click Edit.
- To start the data format record editor, double-click the number of the row containing the record you wish to edit. You may also select the row and then click Edit.

Data format loop editor paths

Within the data format loop editor, you can navigate to data formats, loops, and records, as follows:

From the data format loop editor General tab, you can display a list of data formats and loops in which the current loop is used.

Click Where Used to display a list window containing Data Format and Data Format Loop tabs. Double-clicking entries in those lists starts the respective editors.

From the data format loop editor Details tab, you can start the data format loop and data format record editors.

To start the data format loop editor, double-click the number of the row containing the loop you wish to edit. You may also select the row and then click Edit.

To start the data format record editor, double-click the number of the row containing the record you wish to edit. You may also select the row and then click Edit.

Data format record editor paths

Within the data format record editor, you can navigate to data formats, loops, structures, and fields, as follows:

From the data format record editor General tab, you can display a list of data formats and loops in which the current record is used.

Click Where Used to display a list window containing Data Format and Data Format Loop tabs. Double-clicking entries in those lists starts the respective editors.

From the data format record editor Details tab, you can start the data format structure and data format field editors.

To start the data format structure editor, double-click the number of the row containing the structure you wish to edit. You may also select the row and then click Edit.

To start the data format field editor, double-click the number of the row containing the field you wish to edit. You may also select the row and then click Edit.

Data format structure editor paths

Within the data format structure editor, you can navigate to records, structures, and fields, as follows:

From the data format structure editor General tab, you can display a list of records and structures in which the current structure is used.

Click Where Used to display a list window containing Data Format Record and Data Format Structure tabs. Double-clicking entries in those lists starts the respective editors.

From the data format structure editor Details tab, you can start the data format structure and data format field editors.

To start the data format structure editor, double-click the number of the row containing the structure you wish to edit. You may also select the row and then click Edit.

To start the data format field editor, double-click the number of the row containing the field you wish to edit. You may also select the row and then click Edit.

Data format field editor paths

Within the data format field editor, you can navigate to records and structures, as follows:

From the data format field editor General tab, you can display a list of records and structures in which the current field is used.

Click Where Used to display a list window containing Data Format Record and Data Format Structure tabs. Double-clicking entries in those lists starts the respective editors.

Reusing data format components

WebSphere Data Interchange Client allows you to reuse data format components.

The data format dictionary is the mechanism that allows you to reuse components.

- When you create a component in a data format dictionary, you can use that component in any *other* data format associated with that dictionary, but you cannot reuse that component in the *same* component.
- You cannot use a loop or structure within itself, directly, or indirectly. That causes a circular reference.
- Names of all components within a dictionary must be unique.
- All data format names must be unique, regardless of dictionary.

  Attention: Because you can reuse components, it is important that you check where each component is used when you change it. Your changes may propagate through several data formats. Use the Where Used button on the general tabs to see which data format components will be affected by any change you make.

Data types for data formats

The following table lists the valid data types permitted in data format fields. These data types describe the contents of the fields in your application's data. It also shows the valid mapping data type associated with the data format data type. The EDI standard data types column shows equivalent EDI standard data types.

Most data format data types use the same format for storing, displaying, and printing data. However, sometimes the storage format is different from the format used to display or print data. These differences are noted in Table 22.

Note: Data types that contain binary data should not be used for comma-separated data formats. This can result in unpredictable behavior since the binary data

Table 22. Data types for data formats (continued)

Data format	EDI standard data types	Data Format description
AC	A AN ID Nn Rn DT TM	Application control A field that contains a control number by which the application identifies the transaction. A purchase order number is an example. The data itself is alpha-numeric. A data format can contain only one field of this type. During transaction mapping, you can specify the application control as a concatenation of up to eight fields. The concatenated application control overrides the AC data type. This data type does not apply to record IDs. An AC data type is assumed to be the same as AN during the value validation at translate time.
AN	A AN ID Nn Rn DT TM	Alphanumeric You can use any combination of characters up to the length of the field.
Bn	AN ID Nn Rn DT TM	Binary (unsigned) Storage format: Data with a binary format with n implied decimal places. A value of 2.3 defined as a 2-byte B2 field would be stored as 1110 0110 (X'E6' or decimal 230). This is the same format as IT or In data, but binary data is not signed and, therefore, all values are considered positive.
BN	AN ID Nn Rn DT TM	Binary (unsigned) Any combination of 0-9 without a sign (+ or -). Storage format: The binary equivalent of a numeric value in either two or four bytes, depending on the length of the field. Example: The value 23 is stored as 0000 0000 0001 0111 (X'0017').

Data types for data formats

Table 22. Data types for data formats (continued)

Data format	EDI standard data types	Data Format description
CH	A AN ID Nn Rn DT TM	Character Any combination of characters up to the length of the field.
DT	DT	Date; does not apply to record IDs A string of 5 to 8 digits, depending on the date format that is used. The acceptable date formats are: ddmmyy, ddmmyyyy, ddyymm, ddyyyymm, ddyy, ddyyyy mmddyy, mmddyyyy, mmyydd, mmyyyydd yymmdd, yyyymmdd, yyddmm, yyyyddmm, yydd, yyyydd
FN		File name. A field that contains the name of a file whose entire contents are mapped to a binary segment. When not mapped to a binary segment, a field with data type FN is treated as if the data type were AN.
Hn	AN ID Nn Rn DT TM	Hexadecimal Hexadecimal data with n implied decimal places. This format is treated as a Bn field when mapped to a numeric data element and as an HX field when mapped to an alpha data element.
HX	AN ID Nn Rn DT TM	Hexadecimal Any combination of 0-9 and A-F up to twice the length of the field. Storage format: Hexadecimal, where the length of the field determines the number of bytes used to hold the value.
ID		Identifier The ID data type is equivalent to an AN data type.

Table 22. Data types for data formats (continued)

Data format	EDI standard data types	Data Format description
In	AN ID Nn Rn DT TM	Integer (signed) Storage format: Data with a binary format with n implied decimal places. A value of 2.3 defined as a 4-byte I2 field would be stored as 0000 0000 1110 0110 (X'E6' or decimal 230).
IT	AN ID Nn Rn DT TM	Integer (signed) Storage format: The binary equivalent for a positive number or the two's complement binary equivalent for a negative number, in two or four bytes, depending on the length of the field. Example: The value +23 is stored as 0000 0000 0001 0111 (X'0017'). The value -23 is stored as 1111 1111 1110 1001 (X'FFE9').
Ln	AN ID Nn Rn DT TM	Decimal (leading sign) Zoned decimal data with n implied decimal places and a leading sign.
N	AN ID Nn Rn DT TM	Numeric Any combination of 0-9 and an optional sign (+ or -). The length includes the sign. When mapping data elements defined as data type N in UN/EDIFACT standards, use data type R.
Nn	AN ID Nn Rn DT TM	Numeric Any combination of 0-9, an implied decimal point with n places to the right of the decimal, and an optional sign (+ or -). Using N alone is the same as using N0 (N zero). The length includes the sign. Example: N2 for a value of 23949 is interpreted as 239.49.

Data types for data formats

Table 22. Data types for data formats (continued)

Data format	EDI standard data types	Data Format description
PD	AN ID Nn Rn DT TM	Packed decimal Any combination of 0-9 with a sign (+ or -). The length defines the number of bytes used to hold the value in external format (minus the sign position). Storage format: The packed decimal equivalent, followed by the sign in the low-order 4 bits of the last byte. The sign is either 1111, 1100, or 1010 for a positive value; or, 1101 or 1011 for a negative value. Example: The value +123 is stored as 0001 0010 0011 1111 (X'123F'). The value -123 is stored as 0001 0010 0011 1101 (X'123D').
Pn	AN ID Nn Rn DT TM	Packed decimal Packed decimal data with n implied decimal places.
R	AN ID Nn Rn	Real Numeric data that requires a decimal point for fractional values. The decimal point is optional for integers. A sign (+ or -) is optional for positive numbers. Positive is assumed if a sign is not present. The length includes the decimal point and sign if they are present. Scientific notation with exponent and mantissa formatting is used. You should use this data when mapping data elements defined as data type N in UN/EDIFACT standards. Examples: 23.949, +23.949, -23949, -39846.7, 50E+4.
Rn	AN ID Nn Rn	Real Signed or unsigned numeric data with a minimum for n significant decimal places. The length includes the decimal point and sign. Any combination of 0-9 with a sign (+ or -).
TM		Time A string of four digits in the form `hhmm` or six digits in the form `hhmmss`, expressed in the 24-hour clock format, where the hour is specified as 00 to 23 for X12 and 00 to 24 for EDIFACT. This data type does not apply to record IDs.

Table 22. Data types for data formats (continued)

Data format	EDI standard data types	Data Format description
ZD	AN ID Nn Rn DT TM	Zoned decimal Any combination of 0-9 with a sign (+ or -). The length defines the number of characters used to represent the value in the external format. The external length requires an extra position for the sign. Storage format: The zoned decimal equivalent in the low-order 4 bits of a byte and 1111 in the high-order 4 bits. The sign displays in the high-order 4 bits of the low-order byte and is either 1100 for a positive value or 1101 for a negative value. The length of the field determines the number of bytes used to store the value. Example: The value +123 is stored as 1111 0001 1111 0010 1100 0011 (X'F1F2F3'). The value -123 is stored as 1111 0001 1111 0010 1101 0011 (X'F1F2D3').
Zn	AN ID Nn Rn DT TM	Zoned decimal Zoned decimal data with n implied decimal places and a trailing sign. Any combination of 0-9 with a sign (+ or -).

Data types for data formats

Chapter 23. Extensible Markup Language (XML)

Extensible Markup Language (XML) is a language that is becoming a popular way to represent structured documents and data. XML defines the syntax used to represent the data, including information such as how to tell an element name from an element value. However, it does not define the semantics or structure of the data. For example, it does not specify where a purchase order number should appear, or even whether it is part of any particular document.

The structure of an XML document is defined by a Document Type Definition (DTD). The syntax of the DTD is defined as part of the XML language. The DTD provides a list of all components included in the XML document and their relationship to each other. For example, a DTD may state that the Header element of the document contains a PONum element. The meaning of the PONum element may be described either in the comments within the DTD, a separate document, or both.

Unlike EDI standards where there are a small number of dominant EDI standard formats that define the document structure, there are numerous different XML DTDs. Also, since XML is "extensible", users are free to create their own DTDs if they choose. Instead of restricting users to a fixed subset of DTDs, WebSphere Data Interchange allows you to import the DTDs you use. You may obtain your DTDs from various sources, such as industry groups, standards bodies, vendors, trading partners, or you may create them yourself. When the DTDs are imported into WebSphere Data Interchange, you can map the documents described by those DTDs.

Note: WebSphere Data Interchange does not support XML Schemas. XML Schemas are a newer way of describing the structure of an XML document.

Imported DTDs are used only for data transformation maps; they cannot be used for send and receive maps.

WebSphere Data Interchange uses XML Dictionaries as way to logically group a set of related DTDs. A DTD name must be unique within its XML dictionary, but does not have to be unique across all XML dictionaries. A DTD may use external references to refer to other DTDs within the same dictionary. Use the XML dictionary editor to create and maintain an XML dictionary.

The XML list window provides access to the XML dictionaries and DTDs. The window displayed when you press the XML button on the navigator bar contains two tabs, as follows:

- The XML Dictionary tab provides access to the XML dictionary list window and XML dictionary editor window.
- The DTDs tab provides access to the DTDs list window and the DTDs editor window.

Accessing XML editors

You access the component editors in essentially the same way. Both the editors display on tabs in the XML list window, as follows.

Note: You cannot change the DTD itself, only the WebSphere Data Interchange properties associated with the DTD. To change a DTD, edit the original DTD

outside of WebSphere Data Interchange using a text editor or a DTD editor, then re-import the DTD into WebSphere Data Interchange.

To access an XML editor:

1. Click XML on the WebSphere Data Interchange Client Navigator bar.

   The XML list window, which contains tabs for the components, displays.

2. Click the tab of the XML component you wish to work with.

   The list window for that component displays.

   This window displays a list of existing XML components. A list of XML dictionaries is shown above. Each row contains information about a component; each column contains data stored in that component. Information in the columns displays in fields in the editor window.

3. To view an item or to add or change information in an item, double-click the row of the item you wish to work with.

   The editor window displays. You add information or make changes to the component through its tabs, as described in the following sections.

Following are detailed procedures for creating XML dictionaries and importing DTDs. For information on viewing, copying, editing, renaming, deleting, and printing components, see "Performing common file management tasks" on page 45. For information on exporting components, see "Exporting" on page 68

The XML component editors are described in the following sections.

Using the XML dictionary editor

An XML dictionary is a named group of related DTDs. The relationship between the DTDs in a dictionary is not fixed. For example, you can group DTDs that refer to one another or DTDs created and maintained by one organization. You can put all DTDs from a particular XML format in one dictionary.

Note: The XML dictionary in WebSphere Data Interchange 3.2 is not related to the XML dictionary created by the DTD conversion utility in WebSphere Data Interchange 3.1.

Creating an XML dictionary

Create a new dictionary when you want to create a logical group of DTDs.

1. At the XML dictionary list window, click New on the tool bar.

   The XML dictionary editor window displays with the General tab in front and the fields blank.

2. Type a name in the Dictionary Name field.

   The name displays in all capital letters. You cannot type spaces within the name.

3. Enter a more complete description of the XML dictionary in the Description field.

4. Click Save on the tool bar to save the dictionary.

Importing a DTD file

After you have created an XML dictionary, DTDs can be imported into WebSphere Data Interchange. When you have obtained a DTD file you want to load into WebSphere Data Interchange, see "Importing a DTD file" on page 71.

Using the DTD editor

The DTD editor is used to maintain properties of the DTD and to view the DTD. It can not be used to change the DTD itself. Change the original DTD using a text editor or a DTD editor.

To open the DTD Editor and view a DTD:

1. Click XML on the WebSphere Data Interchange Client Navigator bar.

   The XML list window, which contains tabs for the components, displays.

2. Click the DTDs tab.

   The list window for DTDs displays.

3. Double-click the row of the DTD you wish to work with.

   The General tab in the DTD editor window displays.

4. Complete any of the optional fields in the editor.

   Click ❓

   for field descriptions.

5. Click Save on the tool bar to save the DTD.

6. Click the View tab on the DTD View Window.

   The View tab page displays, showing the contents of the DTD.
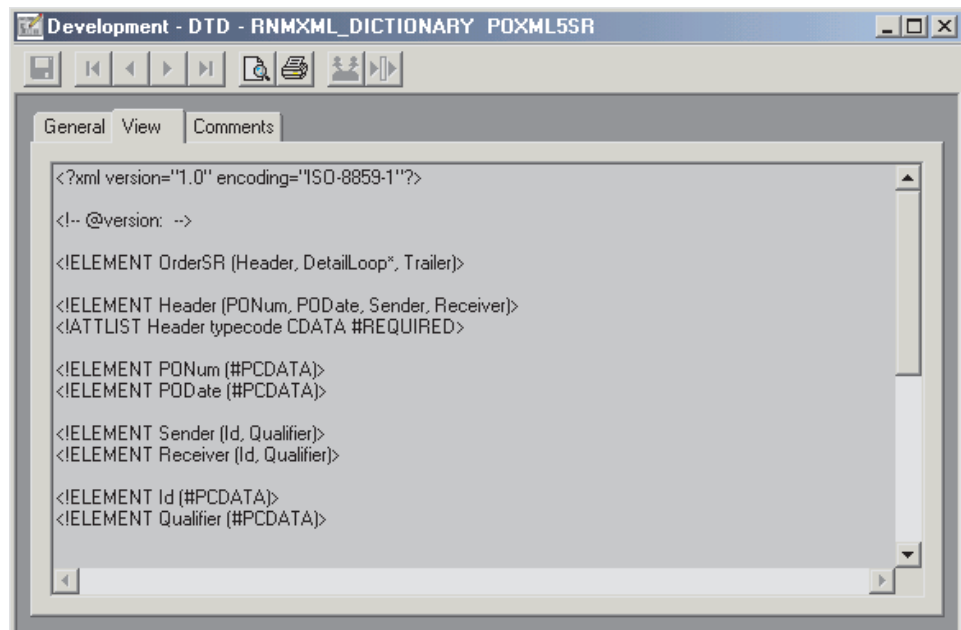
Using the DTD editor

Chapter 24. EDI standards

EDI standards provide a common document layout that trading partners use to exchange data between their computer applications. In essence, EDI standards provide the building blocks for electronic versions of common business documents.

WebSphere Data Interchange translates data from one document layout to another. Commonly, the source document layout describes a document from a business application, which is translated into an EDI standard transaction for transmission to a trading partner. Conversely, WebSphere Data Interchange commonly translates data received in an EDI standard transaction into a format used by a business application. When you install WebSphere Data Interchange, you receive copies of EDI standards currently approved by the primary EDI standards organizations. You can also download EDI standards from the WebSphere Data Interchange Web site at `http://www.ibm.com/websphere/datainterchange` For instructions on how to install EDI standards, see "Importing EDI standards" on page 21.

To begin exchanging documents with a trading partner using an EDI standard transaction, you must select an EDI standard transaction that corresponds to the information you want to send or receive. Ideally, you and your trading partner can agree on an EDI standard transaction that requires no customization to meet your needs, but this is not always possible. Imagine, for example, that you and your trading partner need to exchange specific information that is not included in existing EDI standards. WebSphere Data Interchange Client allows you to customize currently approved EDI standards so that they will fit your needs.

Attention: When altering EDI standards, you should work in close partnership with your trading partners. If you customize EDI standards without informing your trading partners of the changes, they may not be able to process the transactions you send.

> **WebSphere Data Interchange terminology note**
> The terms "transaction set" in ANSI ASC X12 and "message" in UN/EDIFACT are equivalent to "transaction" in WebSphere Data Interchange.

About standards

An EDI standard structures data into two basic categories: envelopes and transactions.

Envelopes

Envelopes are made up of the control structures that "wrap" data for communications to trading partners. UN/EDIFACT refers to envelope standards as service segments, and ASC X12 refers to them as envelope interchange control segments. Envelope standards also specify the default delimiters used in the EDI standard data, such as the data element delimiter, subelement delimiter, and segment delimiter. See "Accessing an EDI standard editor" on page 187

Transactions

Transactions correspond to business documents such as purchase orders or invoices. An EDI standard contains one definition for each unique segment and data

element that occur in a transaction. These segments and data elements are then used in as many transaction sets as necessary.

A transaction consists of the following components:

- EDI standard dictionary

  An EDI standard dictionary contains information about all of the transaction sets, segments, and data elements that comprise the specific version and release of an EDI standard. For detailed information about a particular EDI standard, consult the appropriate EDI standards manuals.

  When you install WebSphere Data Interchange, you receive a copy of EDI standard dictionaries currently approved by the primary EDI standards organizations. For information on how to create dictionaries, see "Using the EDI standard dictionary editor" on page 188.

- Transaction set

  Transaction sets represent business documents such as invoices or purchase orders. Transaction sets are called messages in UN/EDIFACT and transactions in WebSphere Data Interchange. A transaction set contains segments. These segments are sometimes grouped into tables representing heading information (such as billing address), detail information (such as line items from a purchase order), and trailing information (such as totals).For more information on how to create or edit transactions, see "Using the EDI standard transaction editor" on page 190.

- Segment

  A transaction set is composed of segments. In essence, each line of a business document corresponds to a segment in the EDI transaction set. Segments begin with a segment identifier assigned by the EDI standard. They are either mandatory, conditional, optional, or floating (floating segments may display anywhere in the transaction). All segments except for floating segments display in a fixed sequence for a given transaction.

  Segments may repeat within a transaction up to the number of times specified by the EDI standard. Groups of segments, such as the group which makes up a name and address, may form a loop. Loops are identified by a loop ID. Entire loops may be repeated in succession up to the number of times specified by the EDI standard. For information on how to create or edit segments, see "Using the EDI standard segment editor" on page 191.

- Data element

  A segment is composed of data elements and composite data elements, which represent the individual units of data found in business documents, such as quantity ordered or unit price. Data elements display in a sequence specified by the EDI standard and are separated by a delimiting character, such as an asterisk. They have a minimum and maximum length, and are either mandatory, conditional, or optional.

  WebSphere Data Interchange Client also supports composite data elements. Composite data elements are composed of a group of logically related simple data elements. A Composite Unit of Measure, for example, is a combination of Unit of Basis for Measurement, Component, and Multiplier. Composite data elements are defined in the EDI standards.

  All data elements must be of a data type prescribed by the EDI standard, such as date, time, and alpha-numeric, and identifiers, such as data type ID. Data elements may have to contain one of a specific set of codes if prescribed by the EDI standard. The EDI standard specifies the list of acceptable codes, which you can customize. For information on how to create or edit data elements, see "Using the EDI standard data element editor" on page 192.

- Code lists

    A code list is a list of acceptable values for segments or data elements that can only contain certain values. If you include a segment or data element that can only contain certain values in the transaction set you are creating, you should enter all acceptable values into a code list.

    When WebSphere Data Interchange processes the transaction, it references the code list and checks the value of a field against it. If the field contains a value that does not display in the code list, WebSphere Data Interchange returns a processing error. For information on how to create code lists, see "Using the code list editor" on page 194.

    > **WebSphere Data Interchange terminology note**
    > Code lists are called "Validation Tables" on the WebSphere Data Interchange Host.

Using the EDI standard editors

Use the EDI standard editors to create or maintain the various components that make up an EDI standard. Although you can use the WebSphere Data Interchange Client standard editors to create a completely EDI new standard, most users are not likely to do that.

The EDI standard editors are most often used to modify existing EDI standards to meet a company's needs. This section provides a generic description of the procedures for using the EDI standards editors.

Use the EDI standard list window to gain access to the EDI standards component editors. Each component editor corresponds to a tab on the EDI standard list window, as follows:

- The EDI Standard Dictionary tab provides access to the EDI standard dictionary list window and dictionary editor window.
- The Transactions tab provides access to the transactions list window and the EDI standard transaction editor window.
- The Segments tab provides access to the segments list window and EDI standard segment editor window.
- The Data Elements tab provides access to the data elements list window and EDI standard data element editor window.
- The Code Lists tab provides access to the code lists list window and code list editor window.

Accessing an EDI standard editor

1. Click EDI Standards on the WebSphere Data Interchange Client Navigator bar.

    The EDI Standards List window, which contains tabs for the EDI standards components, displays.

2. Click the tab of the EDI standard component you wish to work with.

    The list window for that component displays.

    This window displays a list of existing items in an EDI standards component. A list of EDI Standard Dictionaries is shown above. Each row contains information about an item; each column contains data stored in that item. Information in the columns displays in fields in the editor window. The list window, however, also contains the date, time, and user ID of the last update.

To display additional columns, click the scroll bar on the bottom of the screen to scroll to the right or left. To alter the columns that display on the screen, or to change which query is executed to produce the list, click Modify Window Properties (...). To create new queries, refer to "Creating a query" on page 268

3. To view an item or to add or change information in an item, double-click the row of the item you wish to work with.

The editor window displays. You add information or make changes to the EDI standard item using its editor, as described in the following sections.

Following are detailed procedures for creating EDI standards components. For information on viewing, copying, editing, renaming, deleting, and printing components, see "Performing common file management tasks" on page 45. For information on exporting components, see "Exporting" on page 68. (EDI Standard dictionaries, EDI standard transactions, and code lists are the only EDI standards components that can be exported.) For information on using the grid editors that display in some EDI standards editors, see "Using editor window grids" on page 47.

The EDI standard component editors are described in the following sections in the order in which you use them when creating an EDI standard from scratch.

Using the EDI standard dictionary editor

An EDI standard dictionary is a named group of other related components.

Detailed procedures for creating a new dictionary follow. For information on viewing, copying, editing, renaming, deleting, and printing dictionaries, see "Performing common file management tasks" on page 45. For information on exporting dictionaries, see "Exporting" on page 68.

Creating an EDI standard dictionary

Create a new dictionary when you want to create your own customized EDI standard.

1. At the EDI standard dictionary list window, click New on the tool bar.

The EDI standard dictionary editor window displays with the General tab in front and the fields blank.

2. Type a name in the Dictionary Name field.

   The name displays in all capital letters. You cannot type spaces within the name.

3. Select an envelope type from the Envelope Type drop-down list. The available choices are:

   E   EDIFACT delimiter and envelope definitions

   F   Reserved - denotes an EDI standard created as part of the Fixed-to-Fixed mapping process

   I   ICS delimiter and envelope definitions

   L   Reserved - indicates the EDI standard was created using the WebSphere Data Interchange DTD conversion utility

   T   UNTDI delimiter and envelope definitions

   U   UCS delimiter and envelope definitions

   X   X12 delimiter and envelope definitions

   Note: When you save your dictionary, the Transactions, Segments, Composite Elements, and Simple Elements buttons in the List Of box become available. Click those buttons to display list windows that contain the components associated with this dictionary. When you first create a dictionary, the lists are empty.

4. Enter a more complete description of the EDI standard dictionary in the Description field.

5. Enter the version and release of the dictionary.

6. Select an Industry Code from the drop-down list provided; if a list is not available, you can type a name in the list. The available codes are:

   RAIL Association of American Railroads

   UCS Uniform Communication Standard

EDI standard dictionary

> **VICS**   Voluntary Inter-Industry Communications Standard

7. Select an Agency Code from the drop-down list provided; if a list is not available, you can type a name in the list. The available codes are:

> **T**   TDCC, ODETTE
>
> **UN**   UN/EDIFACT
>
> **X**   ASC X12, RAIL, UCS, VICS

8. Click Save on the tool bar to save the dictionary.

To view lists of EDI standard components in the current dictionary:

1. Click the button in the List Of group box corresponding to the component you wish to view.

   A list window displaying EDI standard components associated with this dictionary displays:

   - The Transactions button displays the EDI Standard Transactions List window.
   - The Segments button displays the EDI Standard Segments List window.
   - The Composite Elements button displays the EDI Standard Data Elements List window showing only composite data elements.
   - The Simple Elements button displays the EDI Standard Data Elements List window showing only simple data elements (data elements that are not composite data elements).

2. You can open any item in the list by double-clicking on it.

Using the EDI standard transaction editor

The EDI standard transaction editor allows you to define and structure the components that make up a transaction.

The EDI standard transaction editor window contains three tabs: General, Details, and Comments. Use the:

- General tab to enter and change transaction properties, such as name and description.
- Details tab to add or change the usage of segments associated with the selected transaction.
- Comments tab to type any comments you wish about the selected transaction.

Following are detailed procedures for creating a new transaction. For information on viewing, copying, editing, renaming, deleting, and printing transactions, see "Performing common file management tasks" on page 45.

Creating a transaction

Create a new transaction when the transactions shipped as part of the EDI standards do not meet your business needs. This editor also modifies existing transactions.

1. At the EDI Standards Transactions List window, click New on the tool bar.

   The EDI Standard Transaction Editor window displays with the General tab in front and the fields blank.

2. Type a name in the Transaction field.

   The name displays in numbers and capital letters. You cannot type spaces within the name.

You can enter a more complete description of the transaction in the Description field, and a brief summary of the transaction's purpose in the Purpose field.

3. Select the dictionary in which you want the transaction to occur through the Dictionary drop-down list. This is a required field, as indicated by the blue asterisk.

4. Enter a Functional Group, such as IN for invoice.

5. Click the Details tab to enter information on the segments and data elements contained in this transaction.

6. Complete the optional fields you need.

   Click ? for field descriptions.

7. Use the Details tab to add and delete the transaction's associations with segments. The Details tab is also used to maintain the properties of each of those associations.

   For instructions on working with the grid editor, "Using editor window grids" on page 47

8. When you have completed entering information, click Save on the tool bar to save the transaction.

Using the EDI standard segment editor

Use the EDI standard segment editor window to enter new segments into an EDI standard or to edit existing segments. From the segments editor, you can add or edit the usage of data elements in segments.

The segments editor window contains three tabs: General, Details, and Comments. Use the:

- General tab to name the segment and select its dictionary.
- Details tab to add or change data elements associated with the selected segment.
- Comments tab to type any comments you wish about the selected segment.

Following are instructions for creating a new segment. For information on viewing, copying, editing, renaming, deleting, and printing segments, see "Performing common file management tasks" on page 45.

Creating a segment

Create a new segment when business needs require one.

1. At the EDI Standard Segments List window, click New on the tool bar.

   The EDI Standard Segment Editor window displays with the General tab in front and the fields blank.

2. Type a name in the Segment field.

   The name displays in all capital letters. You cannot type spaces within the name.

   If you wish, you may enter a more complete description of the segment in the Description field, and a brief summary of the segment's purpose in the Purpose field.

3. Select the dictionary in which you want the segment to display through the Dictionary drop-down list. This is a required field.

EDI standard segment

Click the Details tab to enter information on the data elements contained in this segment.

4. Complete the optional fields you need.

Click ![?] for field descriptions.

For instructions on working with the grid editor, "Using editor window grids" on page 47

5. Use the Details tab to add and delete the segment's associations with data elements and composite data elements. The Details tab is also used to maintain the properties of each of those associations.

For instructions on working with the grid editor, see "Using editor window grids" on page 47.

6. When you have completed entering information, click Save on the tool bar to save the segment.

Using the EDI standard data element editor

Use the EDI standard data element editor window to enter new data elements into a standard or to edit existing data elements.

The data elements editor window contains three tabs: General, Details, and Comments. Use the:

- General tab to name the data element and select its dictionary.
- Details tab to add or change the component data elements associated with a composite data element.
- Comments tab to type any comments you wish about the selected data element.

Following are detailed procedures for creating a new data element. For information on viewing, copying, editing, renaming, deleting, and printing data elements, see "Performing common file management tasks" on page 45.

Creating a data element

Create a new data element when business needs require one.

1. At the EDI Standard Data Elements List window, click New on the tool bar.

   The EDI Standard Data Element Editor window displays with the General tab in front and the fields blank.

2. Type a name in the Data Element field.

   The name displays in all capital letters. You cannot type spaces within the name.

   If you wish, you may enter a more complete description of the data element in the Description field, and a brief summary of the data element's purpose in the Purpose field.

3. Select the dictionary in which you want the data element to display through the Dictionary drop-down list. This is a required field, as indicated by a blue asterisk.

4. Select the data element's data type from the drop-down list in the data type field. This is a required field. The data types are explained in Table 23 on page 193.

5. Enter the minimum and maximum length of the data element in the Min Length and Max Length fields. These are required fields.

6. If the data element you are creating can only contain certain values, select the code list which identifies acceptable values for the data element you are creating from the drop-down list that displays in the Code List field.

   If none of the existing code lists specify the acceptable values for your data element, you may create a new code list. See "Using the code list editor" on page 194.

7. Complete the optional fields you need.

   Click ![?] for field descriptions.

8. The Details tab is only available when the data type is CD (composite data element). If you are creating a composite data element, select CD in the data type drop-down list. Then click on the Details tab to identify the component data elements that will be associated with this composite data element and the properties of those associations.

   For instructions on working with the grid editor, see "Using editor window grids" on page 47.

9. When you have completed entering information, click Save on the tool bar to save the data element.

Table 23. Data types for EDI standard data elements

Data Type	Name	Description
A	Alphabetic	Alphabetic characters up to the length of the field.
AN	Alphanumeric	You can use any combination of characters in the ALPHANUM code list, up to the length of the field.
CD	Composite data element	A data element with data type CD and data element ID beginning with a C by convention for EDI standard composite data elements and S for envelope composite data elements. The component data elements are defined using the Details tab in the EDI Standard Data Element Editor.
CH	Character	Any combination of characters up to the length of the field.
DT	Date	Date format yyyymmdd, where yyyy is the year, mm is the month (01-12), and dd is the day.
ID	Identifier	A data element which usually has a code list for the valid values for the data element. For example, data element UM, unit of measure, has data type ID, and the valid values for this data element are listed in the UMCODES code list. The table name is the same as or starts with the data element ID.
IV	Incrementing value	A data element, such as a message reference number, that starts at 1 and increases by 1 for each usage.
N	Numeric	Any combination of 0-9 and an optional sign (+ or -). The length includes the sign. When mapping data elements defined as data type N in UN/EDIFACT EDI standards, replace it with data type R (because data type N in the ASC X12 EDI standards is the same as data type R in UN/EDIFACT EDI standards).

EDI standard data element

Table 23. Data types for EDI standard data elements (continued)

Data Type	Name	Description
Nn	Numeric	Numeric data with N places to the right of an implied decimal point. The only acceptable characters are the digits 0 through 9. N is the same as N0. For example, if the data type is N2, the value 123 means 1.23. A sign (+ or -) is optional. Positive is assumed if no sign is present. The length does not include the sign. The data type should be used when defining data elements defined as data type N in UN/EDIFACT EDI standards.
PW	Password	A password used in the interchange or functional group header.
R	Real	Numeric data that requires a decimal point for fractional values. The decimal point is optional for integers. A sign (+ or -) is optional. Positive is assumed if no sign is present. The length does not include the decimal point and sign. This data type should be used when defining data elements defined as data type N in UN/EDIFACT EDI standards.
Rn	Real	Signed or unsigned numeric data with a minimum of n significant decimal places. On sending, at least n decimal places are generated. On receiving, the decimal places are the same as data type R. A sign (+ or-) is optional. Positive is assumed if no sign is present. The length does not include the decimal point and sign.
TM	Time	Time format is `hhmm` or `hhmmss`, depending on the length of the data element, where `hh` is the hour, `mm` is the minutes, and `ss` is the seconds. The time format uses a 24-hour clock, where the hour is specified as 01 to 24 for EDIFACT and 00 to 23 for X12.

Using the code list editor

A code list is a list of acceptable values for data elements which can only contain certain values.

Following are detailed procedures for creating a new code list. For information on viewing, copying, editing, renaming, deleting, and printing profiles, "Performing common file management tasks" on page 45 For information on exporting profiles, "Exporting" on page 68

Note: Code lists can also be specified in data format fields (when used with send and receive maps) and can be used in all maps.

Creating a code list

Create a new code list when you create a data element which can only contain certain values. A code list can also be created if you wish to restrict the values of any data item during translation. In this case, the code list is specified during mapping.

1. At the list window for Code Lists, click New on the tool bar.

   The Code List Editor window displays with the General tab in front and the fields blank.

2. Type a name in the Code List field.

The name displays in numbers and capital letters. You cannot type spaces within the name.

If you wish, you may enter a more complete description of the code list in the Description field.

3. Choose a Data Type from the drop-down list that displays for the Data Type field. Your choices are:

   CH   Characters

   R   Real numbers

4. Choose the length of entries from the drop-down list that displays for the Length of Entries field. Entries may consist of one to thirty-five characters or numbers.

5. In the grid at the bottom of the Code List General tab, enter the acceptable entries that will be part of this code list in the Entry column, then enter a description of what this entry signifies in the Description column. For instructions on working with the grid editor, see "Using editor window grids" on page 47.

6. Click Save on the tool bar to save the code list.

Using the envelope standard editor

An envelope standard is a name under which components of an EDI envelope standard are grouped.

Following are detailed procedures for editing an envelope standard. For information on viewing, copying, editing, renaming, deleting, and printing profiles, see "Performing common file management tasks" on page 45.

Envelope standards and control strings from the database are not used with Data Transformation maps. Instead the WebSphere Data Interchange server uses plug-in enveloper and deenveloper modules for the supported standards.

Note: An envelope standard cannot be created - it must be imported. Obtain envelope standards from the WebSphere Data Interchange Web site at `http://www.ibm.com/websphere/datainterchange`.

Editing an envelope standard

Edit existing envelope standards when you want to change the default values of envelope fields for envelopes being created during translation.

1. Select the envelope standard you want to edit from the list displayed on the Envelope Standards List window.

   The Envelope Standard Editor window displays with the General tab in front and the Dictionary field filled in with the name of the selected dictionary.

2. Edit the description of the envelope standard in the Description field.

   Note: Click the Segments, Composite Elements and Simple Elements buttons to display list windows that contain the components associated with this envelope standard.

3. Enter the version and release of the envelope standard.

4. Select an Industry Code from the drop-down list provided; if a list is not available, you can type a name in the list. The available codes are:

   RAIL   Association of American Railroads

   UCS   Uniform Communication Standard

EDI envelope standard

> **VICS**   Voluntary Inter-Industry Communications Standard

5. Select an Agency Code from the drop-down list provided; if a list is not available, you can type a name in the list. The available codes are:

> **T**   TDCC, ODETTE

> **UN**   UN/EDIFACT

> **X**   ASC X12, RAIL, UCS, VICS

6. Click the Delimiters tab to add the envelope specific information.
7. Complete the fields on the Delimiters tab.

   Click for field descriptions.

8. Click Save on the tool bar to save the changes to the envelope standard.

To view lists of EDI standard components in the current envelope standard:

1. Click the button in the List Of group box corresponding to the component you wish to view.

   A list window displaying EDI standard components associated with this envelope standard displays:

   - The Segments button displays the EDI Standard Segments List window.
   - The Composite Elements button displays the EDI Standard Data Elements List window showing all composite data elements in this envelope standard.
   - The Simple Elements button displays the EDI Standard Data Elements List window showing all simple data elements (those that are not composite data elements) in this envelope standard.

2. You may open any of the components by double-clicking them.

Envelope control strings

An envelope control string helps improve performance of the translator when envelopes are prepared. Envelope control strings must be compiled after they are installed and after any change is made. If stand-alone mode is being used on the client, the envelope control string must be exported from the Client and imported into the Host. In client/server mode, the envelope control string will be automatically stored on the Host database during the compile.

The Envelope Control String List window displays a list of compiled envelope standards and shows the compilation information about each.

To compile an envelope control string:

1. Select the envelope control string to be compiled from the list displayed on the Envelope Control Strings List window. Alternately, select the envelope standard on the Envelope Standards List window.
2. Click Compile Control String(s) on the tool bar of the list window.

   While compiling, WebSphere Data Interchange Client checks for errors in the changes you made. Error messages are written to the event log.

Chapter 25. Creating a map

Mapping is the process by which you relate input and output documents. Through mapping, you specify the relationships between the internal documents used by your applications and the external documents that you exchange with trading partners. Maps are then compiled into control strings which allow WebSphere Data Interchange Translation Servers to transform documents between the internal and external formats.

WebSphere Data Interchange Client is designed to make mapping easy. You can choose to select elements in your source document, drag them onto elements in the target document, and drop them. Or you can choose to select elements in your target document, drag them onto elements in the source document, and drop them.You can then apply WebSphere Data Interchange's specialized mapping functions as required.

Getting started

Follow the process below to create a map. The same basic steps work for maps that you create to translate data that will be sent to trading partners and for maps that you create to translate data that will be received from trading partners. In both cases, you are mapping data from one document format to another.

1. Obtain the layout of the source document to be used in the translation.

   This might be a data format, an EDI standard transaction, or an XML DTD. You might need to obtain the layout from your trading partner. When you have the source document layout (called the source document definition), put that layout into WebSphere Data Interchange.

   If the source document is data in a proprietary format, create a data format in WebSphere Data Interchange. For information on creating a data format, see "Creating a data format" on page 151.

   If the source document is an EDI standard transaction, study the layout of the transaction. EDI standards are broad. You can use a number of methods to map onto an EDI standard transaction. Any number of organizations are likely to use different methods to map the same data to the same transaction. When you are comfortable with the transaction, make sure the transaction is defined in WebSphere Data Interchange. For instructions on how to install EDI standards, see "Importing EDI standards" on page 21.

   If the source Document is in XML format, obtain the corresponding DTD and import it into WebSphere Data Interchange. This is done using import. For information about importing a DTD file, see "Importing a DTD file" on page 71.

   Attention: If you change the metadata for a document, it affects each map that uses that document as either a source or target document.

2. Obtain the layout of the target document to be used in the translation.

   This might be a data format, EDI standard transaction, or an XML DTD. You might need to obtain the layout from your trading partner. When you have the target document layout (called the target document definition), put that layout into WebSphere Data Interchange. Refer to step 1 for additional information.

3. Decide how source data is handled.

   When you are creating a map to send a document to a trading partner, study the target document definition and decide how you want to use it to pass your source data.

Creating a map

When you are creating a map that will receive a document from a trading partner, study your trading partner's document definition to see what data your partner is sending and how you will handle it.

4. Compare your source document definition to the target document definition.

   When you map, you are associating elements in one document definition with elements in another document definition.

   Study the source document definition for which you are creating a map. Make note of which elements in the source document definition correspond to which elements in the target document definition. Then decide how you are going to map the two document definitions. For example, you might use a loop on an EDI standard transaction to handle repeating records in a data format.

5. Map the data elements in each segment.

   Use the Map Editor to drag elements between the source and target document definitions. You might notice that some elements in target document definition do not occur in the source document definition. If they are required, you need to fill them with data on the outbound side. WebSphere Data Interchange can fill many such elements using special handling options, literals, mapping commands, and accumulators.

   Special handling options and mapping commands allow you to perform such functions as translating date formats, converting values supplied by a trading partner to values you require, and validating the contents of data elements.

   Accumulators and variables allow you to perform such actions as counting segments in a transaction for later placement in a transaction's trailer record.

   Literals and mapping commands are a means of supplying data to data elements or fields, such as dates and times.

6. Specify how loops and repeating segments are handled.

   Document definitions can include repeating elements. Examples of repeating elements are loops in a data format, records, and structures. Repeating elements in data formats include loops, records, and structures. You must specify how WebSphere Data Interchange should handle the repeating elements.

7. Associate the map with a trading partner using a usage or map rule.

   After you have created a map, you must associate it with trading partners. Because you can use the same map for multiple trading partners, each usage or rule identifies one or more trading partners that use the map. The usage or rule can supply values specific to that trading partner or group of trading partners, such as:

   - Validation and error levels
   - Unique values for enveloping that override the default values
   - The logical name of the file to which output data should be written

   For more information, see "Understanding processes and rules" on page 132 and "Understanding minimal trading partners" on page 134.

8. Compile a map.

   The WebSphere Data Interchange translator uses a control string in its processing, not the map itself. When you complete your mapping work, you must compile the map to create a control string. The compiler identifies any errors when it creates a control string.

Choosing the right map

WebSphere Data Interchange supports several types of maps that you can use to transform data that you exchange with other business and trading partners:

- Data transformation maps

  Using data transformation maps, you can define any-to-any transformations that allow you to create complex mapping operations. These support the wide range of mapping that you might need to handle data formats, EDI, and XML messages.

  For information about creating a data transformation map, see Chapter 26, "Data transformation mapping" on page 201.

- Functional acknowledgement maps

  A set of standard functional acknowledgment maps is sent with WebSphere Data Interchange. You can customize these maps, or create your own if you have special requirements for your functional acknowledgments.

  For information about creating a functional acknowledgement map, see "Creating a functional acknowledgment map" on page 259.

- Validation maps

  Validation maps allow you to specify extended checking of the EDI input data, beyond the normal standard validation performed by WebSphere Data Interchange. Using a validation map, you can check whether the data conforms to an implementation guide or other user-defined requirements.

  For information about creating a validation map, see "Creating a Validation map" on page 252.

- Send and receive maps

  Send and receive maps are supported in WebSphere Data Interchange 3.2 for migration and compatibility. If you have created send and receive maps in previous releases, you can continue to use these. When you are familiar with the other map types and their editors, you can migrate your send and receive maps and begin to take advantage of the enhanced capability that they offer.

  You are strongly recommended to use the data transformation, validation, or functional acknowledgement map editor for all new maps, and use the send and receive map editors for maintenance of existing send and receive maps only.

  If you are a new WebSphere Data Interchange user, do not create send and receive maps. Use the Data Transformation Map editor, the validation map editor, or the functional acknowledgement map editor for all maps to take advantage of the enhanced function that they provide. You can therefore skip the chapters on creating send and receive maps and usages. Send usages and receive usages are the send and receive mapping equivalents of data transformation map rules.

  For information about creating send and receive maps, see Chapter 27, "Send and receive mapping" on page 219.

Choosing the right map

Chapter 26. Data transformation mapping

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.

The map editor

WebSphere Data Interchange Client's map editor features a visual means for associating elements from a source document definition with elements in a target document definition, or from a target document definition with elements in a source document defintion. The map editor uses a split screen that permits you to create map associations.

The source document definition appears in the upper left corner of the map editor and displays the layout of the document that will be used as the source of the translation. The target document definition displays the layout of the document that will be created by the translation. It is displayed in the upper right hand corner of the Map Editor. The map editor displays a mapping commands window pane in its lower left hand corner. The mapping commands window pane displays a representation of either the source document definition or the target document definition, depending on whether the map is source-based or target-based. The mapping commands and comments inserted into it. There is also a variables window pane in the map editor. It is divided into three sub-panes that list all special variables, global variables, and local variables.

Starting the map editor

The Map Editor displays when you select a map from the Mapping List window, as follows.

1. Click Mapping on the WebSphere Data Interchange Client Navigator bar.

    The Mapping Functional Area displays.

    This window is used to access the list windows for each component type that occurs in the Mapping Functional Area. Select the Data Transformation Maps tab to view the Data Transformation Map List window. The list window shows all of the Data Transformation Maps selected by the current query. Each row on a list window displays information about a different Data Transformation Map.

    To display additional columns, click the scroll bar on the bottom of the screen to scroll to the right or left. To alter the columns that display on the screen, or to change which query is executed to produce the list, click Modify Window Properties . To create new queries, refer to "Creating a query" on page 268

2. To view a map or to add or change its information, double-click the row of the map you want to work with.

    The Data Transformation Map editor window displays, with the Details tab in front. You add information or make changes to maps through its tabs and related dialog boxes, as described in the following sections.

Using the Data Transformation Map editor

The Data Transformation Map editor allows you to show how data is to be translated from the source document to the target document. The editor works by displaying the source document definition on one side of the screen and target

document definition on the other. The Data Transformation Map editor allows you to associate components of your source document with components of the target document by dragging source components and dropping them into the correct locations in the target document definition. Components include simple elements, such as fields in a data format or data elements in an EDI standard transaction, and compound elements, such as loops and record in a data format and segments in an EDI standard transaction.

You can map the simple elements in the source document definition to the simple elements in the target document definition in any of these patterns:

- One simple element to one simple element
- Several simple elements to one simple element
- One simple element to several simple elements

The Data Transformation Map editor window contains three tabs: General, Details, and Comments. Use the:

- General tab to enter and change map properties.
- Details tab to create and maintain the mapping commands.
- Comments tab to type any comments you wish about the selected map.

Following are detailed procedures for creating and editing maps. For information on viewing, copying, editing, renaming, deleting, and printing maps, see "Performing common file management tasks" on page 45. For information on exporting maps, see "Exporting" on page 68.

Creating a new Data Transformation map

Create a new map when you want to translate a document from one format to another.

In the most basic sense, creating a new map consists of associating simple elements in a source document definition with simple elements in a target document definition. The following procedure shows how WebSphere Data Interchange Client makes basic associations between simple elements.

1. In the Mapping List window, click the Data Transformation Maps tab so that any existing Data Transformap maps are displayed.
2. Click the Create New Document icon on the tool bar.

    The Create a Data Transformation Map — Map Name wizard page is displayed.
3. Type a map name in the Map Name field.

    You can use both letters and numbers to identify your map. Letters display in capitals. You cannot type spaces within the name.

    You can enter a more complete description of the map in the Description field.
4. Click Next to continue. The Source or Target wizard page displays.

    Select the appropriate radio button according to whether you want to create a source-based Data Transformation map, or a target-based Data Transformation map.
5. Click Next to continue.

    The Source Syntax Type wizard page displays.
6. Select the appropriate radio button that indicates the syntax type of your source document definition, and click Next to continue.

    The Source Dictionary wizard page displays with a list of the dictionaries for your specified syntax type.

7. Select your source dictionary and click Next.

   A wizard page displays with a list of the document definitions in your source dictionary. The wizard page is titled Source Data Format, Source EDI Standard Transaction, or Source XML DTD, depending on the Source Syntax Type previously selected.

8. Select the data format, the EDI Standard transaction, or the XML DTD to be used as the source document definition in your map, and click Next.

   The Target Syntax Type dialog box displays.This page allows you to indicate the syntax type of your target document definition.

9. Select the syntax type of your target document, and click Next.

   The Target Dictionary wizard page displays with a list displays of the dictionaries for your specified syntax type.

10. Select your target dictionary, and click Next.

    A wizard page displays with a list of the document definitions in your target dictionary. The wizard page is titled Target Data Format, Target EDI Standard Transaction, or Target XML DTD, depending on the Target Syntax Type previously selected.

11. Select the data format, the EDI standard transaction, or the XML DTD to be used as the target document definition in your map, and click Next.

    The Confirmation window displays.

12. Confirm the selections displayed. If they are correct, click Finish to save the information.

    After finishing the map information, the Map Editor displays with the Details tab opened.

    a. Click the General tab.

       The General tab displays.

       The Map Editor's General tab page contains general information about the map. It includes the map name, a brief description of the map and it identifies the source and target document definitions.

       The name of the map is set when the map is created. It displays on the General tab page for informational purposes only. If you want to change it, you must use the rename action.

       Use the Description field to provide a brief description of the map. The description can be up to 50 characters long.

General editing procedures

The Details tab in the Map Editor allows you to perform drag and drop mapping on your documents.

General editing procedures
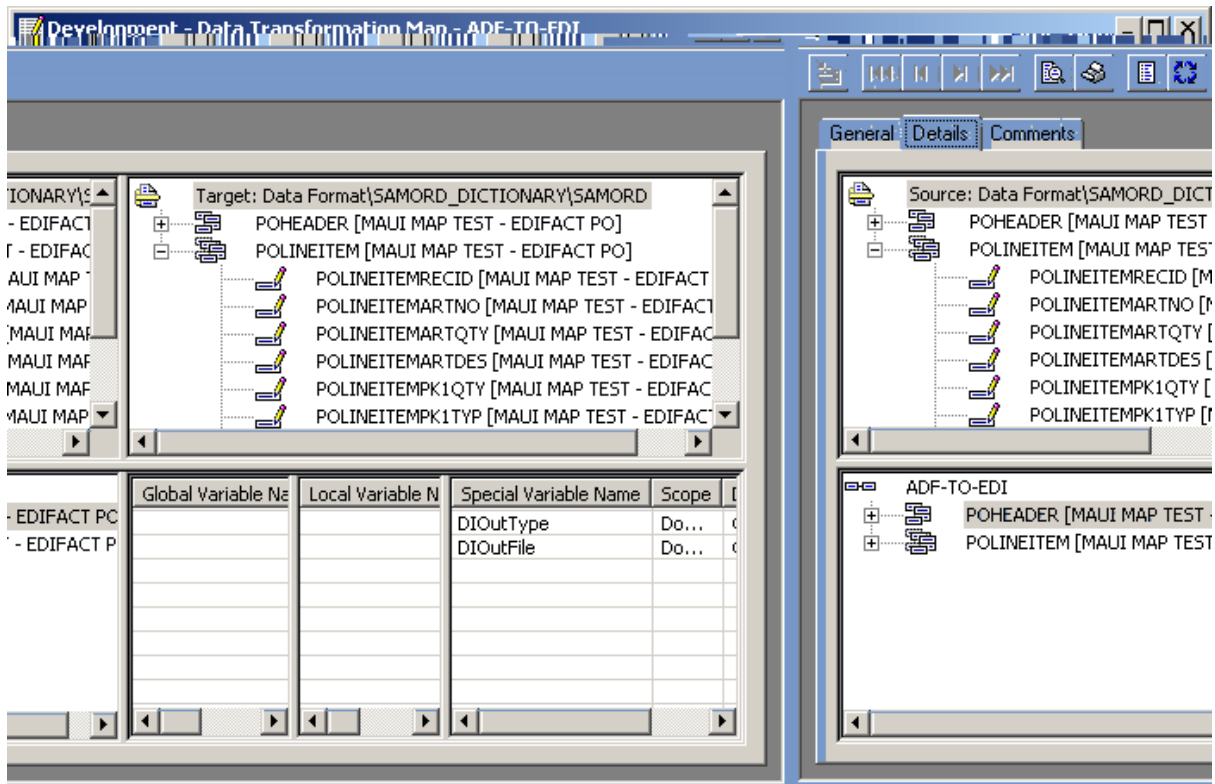
Figure 8. Data Transformation Map Editor

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.

For more information about the Map Command window pane, see "Using the Map Command window pane" on page 207.

For a list of mapping components and their associated graphics, see Table 24 on page 206.

1. Click the plus (+) sign next to a compound element, such as a loop, segment, or record in the target document definition. Simple elements, such as fields and data elements, will not have a plus (+) sign next to them.

   The compound element expands to show the compound and simple elements that comprise the compound element.

   If you wish to close any expanded compound element, click the minus (-) sign.

2. Click the element you want to map on the top left side of the screen. 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 screen over the element with which you want to associate it, that component becomes highlighted. Release the mouse button.

   You can drag all simple elements and repeating compound elements. When you are dragging an element, holding it over a compound element will cause the compound element to expand after a few moments, if it is not already expanded. Dragging an element to any edge of the window pane will cause the

window pane to automatically scroll up, down, left or right, depending on which edge you are near and whether the window pane can be scrolled in that direction.

Note: If the element does not become highlighted when you move the cursor on top of it, that means it is not a valid place to perform a drop. For example, you cannot drop a data format record onto a data element in an EDI transaction.
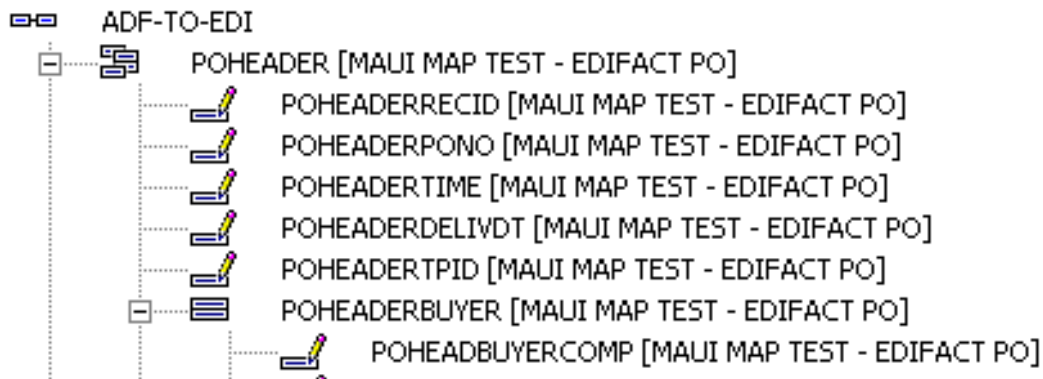

Figure 9. Data Transformation Mapping

After you have mapped an element, a command is created in the mapping commands window pane and inserted into an appropriate position. A check mark displays next to the mapped element in the mapping command window. A check mark also displays next to its parent elements so that you know the parent element contains mapped components.

Note: This procedure presents a simple mapping situation in which you associate elements in the source document definition with elements in the target document definition. For anything but the most basic mapping associations, you must use WebSphere Data Interchange's advanced mapping capabilities. For more information, "Using the Map Command window pane" on page 207

3. Continue dragging and dropping elements in the source document definition onto elements in the target document definition until you have mapped all of the information required for this map.

4. Enter any general comments on the map in the Comments tab.

5. When you have completed mapping, click Save on the tool bar to save the map.

Editing a map

Edit a map when you have modified the associated target or source document definition, or when you need to add, delete, or change mapping commands within the map. You might also need to edit a map to meet specific requirements of a new trading partner.

1. In the Mapping List window, double-click the map you wish to edit.

   The map displays in the Map Editor window with the Details tab in front.

   Changes you made earlier to the map or its associated items display on the screen, as well as changes you made to the source or target document definition.

2. Create new associations between the source and target elements, or change existing ones.

General editing procedures

- To make an association between a new element in the source document definition and an element in the target document definition, drag the source element and drop it on the proper target element.
- To edit an existing association, double-click the mapping command in the Mapping Command window pane. The Mapping Command Editor displays.
- To delete an association, select the mapping command you want to delete and press the Delete key.

3. Change information as required in the General tab.
4. Click Save on the tool bar to save the map.

Table 24. Symbols used in Data Transformation maps

Symbol	Mapping	Data Format	EDI Standard	XML DTD
			EDI Standard Transaction	
			Table	
		Record	Segment	
		Repeating Record	Repeating Segment	
		Loop	Loop	
		Repeating Loop	Repeating Loop	
		Structure	Composite Data Element	Compound Element
		Repeating Structure	Repeating Composite Data Element	Repeating Compound Element
		Field**	Simple Data Element**	Simple Element**
			Repeating Simple Data Element**	
	Mapping Indicator			
	Mapping Command			
	Command Group			
	Comment			
	Comment Group			
	If, ElseIf, Else, EndIf			
	Qualify			

Table 24. Symbols used in Data Transformation maps (continued)

Symbol	Mapping	Data Format	EDI Standard	XML DTD
	Qualify (multi-occurrence)			

** Simple Elements - all others are Compound Elements

Using the Map Command window pane

The Map Command window pane allows you to apply WebSphere Data Interchange Client's advanced mapping capabilities.

1. Right-click an element in the Mapping Command window pane and follow the cascading menus. You can choose to insert a command, a command group, a comment, or a comment group. You can choose to insert your selection before, after, or within the element you have selected.

2. If you want to insert a command, select where the command should be inserted, select Command, and then select the command to be inserted.

   The Mapping Command Editor displays with a prototype of the command.

3. Edit the prototype to create the command you need.

4. Click Repeat if you want to add the command, and create another similar to it.

5. Click OK when you have your last command ready.

Specifying qualification

Terminology

You must understand the following terminology when you use qualification.

- Simple element - a single value
- Compound element - a grouping of elements
- Element - refers to simple or compound element

In data formats, simple elements are fields within the data format. Compound elements include structures, records, and loops.

In EDI standards, simple elements are data elements or sub-elements. Compound elements include loops, segments, and composite data elements.

An XML element is a compound element. An XML attribute and an XML value are considered simple elements.

Within your document, certain elements and groups of elements can repeat. When you map an element that repeats within the source document (whether compound or simple element), you must tell WebSphere Data Interchange which occurrence of the segment or loop you are using. This is called "qualifying" the segment or loop.

WebSphere Data Interchange supports four types of qualification for data transformation maps. You can qualify by:
- Occurrence
- Multi-occurrence
- Value
- Expression

Qualification

When you qualify by occurrence, your mapping is related to the position of data in a repeating sequence. When you qualify by value, your mapping is related to the value that you receive in a simple element or in a variable. When you qualify by expression, your mapping is based upon a condition. For multi-occurrence qualification, you are indicating what mapping instructions should be executed for iterations of the repeating element that are not handled in a previous qualification. If there are no qualifications in place for a repeating element, then it is handled as a multi-occurrence qualification by default.

Qualifying repeating simple and compound elements

WebSphere Data Interchange allows you to qualify compound elements by single occurrence, multi-occurrence, value, and expression.

Qualify by occurrence when the order in which repeating data occurs in either the source or target document is important. For more information, "Qualifying an element by occurrence"

Qualify by multi-occurrence when you want to create multiple elements in the target to correspond to repeating elements in the source using the same mapping instructions. All iterations of the repeating element that are not otherwise qualified will be handled under the multi-occurrence qualification. There can only be one multi-occurrence qualification for a repeating element. For more information, see "Qualifying an element by multi-occurrence" on page 209

Qualify by value when you want a set of mapping instructions to be executed for a repeating element when a specific value occurs in a source element or in a variable. For more information, see "Qualifying an element by value" on page 210

Qualify by expression when you need a more complex qualification. For more information, see "Qualifying an element by expression" on page 211

Qualifying an element by occurrence

Qualify an element by occurrence when a specific instance of a repeating element requires mapping instructions specific to that occurrence of the repeating element. For example, say that you are working with a source document that has an address compound element. The first occurrence of the address compound element has the send-to address, the second occurrence has the ship-to address. Map each occurrence to the appropriate target element using occurrence qualification.

Note: Whether you are doing qualification in a source or target-based map, you are always qualifying on something in the source. `Qualify(Occurrence() = 1)` means an the first Occurrence of something in the source, not an Occurrence of something in the target.

1. For a source based map, right-click the repeating source element in the Mapping Command window pane. For a target based map right-click the ″For Each″ command underneath the repeating target element. If one does not exist, then first right-click the repeating target element and select ″For Each″ to create one.

2. Select By Occurrence.

   The Qualify by Occurrence Number dialog box displays.

3. Type in a number in the Enter the Occurrence Number field that corresponds to the occurrence number of the element you are mapping.

   If you are mapping the second occurrence of an element, type 2 in the Enter the Occurrence Number field.

4. If you need to map additional occurrences, click Repeat.

   A mapping command is created that qualifies the element by occurrence and then redisplays the Qualify by Occurrence Number dialog box.

5. When you have specified the desired number of occurrences, click OK.

   A mapping command is created that qualifies the element by occurrence, and the Qualify by Occurrence Number dialog box is closed.

You can qualify an element by more than one occurrence in a single expression. For example the expression:

```
Qualify ((Occurrence() EQ 1) OR (Occurrence() EQ 2))
```

performs the same set of mapping commands for occurrence one and occurrence two of the repeating element. This command can be created by using the ″Qualify by expression″ command or by creating a qualify by occurrence and then editing the qualify by occurrence command after it has been created.

Qualifying an element by multi-occurrence

Qualify an element by multi-occurrence when you need WebSphere Data Interchange to handle all occurrences of a repeating element, that are not otherwise qualified, in the same way. For example, say you are working on an EDI standard transaction to data format map and you find that the PO1 segment in a purchase order repeats to handle multiple purchase-order line items. You need WebSphere Data Interchange to create a separate record for each instance of the PO1 segment and when each occurrence of the PO1 segment is to execute the same mapping instructions. Only one multi-occurrence qualification is allowed per repeating element. It is always the last qualification. When there are no qualifications under a repeating element, a multi-occurrence qualification is assumed.

Consequently, you would qualify the PO1 segment by multi-occurrence. That way, WebSphere Data Interchange will create as many line-item records in your application data as there are occurrences of PO1 in your trading partner's transaction.

WebSphere Data Interchange Client's mapping function automatically qualifies repeating elements. When you create mapping commands under a repeating element, and no qualifications exist for the repeating element, the repeating element is treated as if it is multi-occurrence qualified. The mapping commands under the repeating element will be executed for each occurrence of the repeating element.

The multi-occurrence qualification commands for source-based maps and target-based maps are different. Source-based maps use `Mapto()` and the Default qualification commands, but target-based maps use the For Each and Default commands. In addition, you can only have one multi-occurrence qualification (Default) on a source repeating element, but you can have multiple multi-occurrence qualifications (For each) on a target repeating element. Although they have different names and usage rules, they accomplish the same thing.

For example, if you want to map two repeating source elements (S1 and S2) to a single repeating target element (T1), then you should either create two Default qualifications in a source-based map (one on S1 and one on S2), or two For Each qualifications in a target based map (both on T1).

The main difference between multi-occurrence qualification in source-based and target-based maps is that because you can map more than one repeating source element to a repeating target element, the tree display has an extra level of

hierarchy. In a source based map the qualifications appear directly under the repeating source element. In a target based map, each repeating source element associated with the repeating target element results in the creation of a For Each command under the repeating target element. The qualifications related to that source element appear under the For Each command.

In source-based maps, the multi-occurrence Default command only displays if there are other existing qualifications for the same repeating element. When you are using multiple qualifications, multi-occurrence qualification always displays last when it is present. This indicates that mappings under the multi-occurrence qualification are performed only when the specific instance of the repeating source element is not resolved to one of the other qualifications. This is why it is known as the default qualification.

In target-based maps, the multi-occurrence qualification For Each command displays whenever there is any type of qualification on the element. Default only displays when there are other existing qualifications for the same repeating source element. When multiple qualifications exists and the multi-occurrence qualification is present, it always displays last. This indicates that mappings under the multi-occurrence qualification are performed only when the specific instance of the repeating source element is not resolved to one of the other existing qualifications. This is why it is known as the default qualification.

Drag a compound element onto a repeating element in the source document definition:

- If there are no existing qualifications for the element, a MapTo command will be inserted under the corresponding repeating element in the Mapping Command window pane.For source based maps, if there are no existing qualifications for the element, a Mapto command will be inserted under the corresponding repeating source element in the Mapping Command window pane. For target based maps, If there are no existing qualifications for the element, a For Each command will be inserted under the corresponding repeating target element in the Mapping Command window pane.

- If there is an existing qualification for the repeating element, the Default command is inserted as the last qualification for the repeating element in the Mapping Command window pane.

The multi-occurrance command ″Default″ is automatically created when you first qualify a repeating element that already has mapping commands under it. The existing mapping commands be moved to under the ″Default″ command.

Qualifying an element by value
Qualify an element by value when you want the value of data in a simple element or variable to drive WebSphere Data Interchange's translation of a repeating element.

For example, say you want to qualify the N1 loop with the value of BY in Element 98, which is the "Entity Identifier Code," received in a purchase order to create a buyer record. Further, say that you want the buyer's name to be mapped into the buyer record depending on the value in Element 98 of the N1 loop.

To handle that case, you would qualify the element by value. That way, WebSphere Data Interchange will put specific information into the buyer records it creates from the purchase order depending on the name in each order's Entity Identifier Code.

1. For a source based map, right-click the repeating source element in the Mapping Command window pane. For a target based map, right-click the ″For

Each″ command underneath the repeating target element. If one does not exist, right-click the repeating target element and select ″For Each″ to create one.

2. Right-click to select Qualify.

3. Click By Value.

   The Mapping Command Editor displays.

4. Drag the element you wish to qualify from the source document window and drop it onto the word **path** on the Mapping Command Editor.

5. Highlight the word **value** in the Qualify field in the Mapping Command Editor.

6. Type the value.

7. If you are creating multiple qualifications by value, click Repeat. Then repeat the previous steps.

   Clicking Repeat creates the qualification in the Mapping Command window pane and redisplays the Mapping Command Editor.

8. When you have created the desired number of qualifications of this element, click OK.

   Clicking OK creates the qualification in the Mapping Command window pane and closes the Mapping Command Editor.

You can qualify an element by several different values in a single expression. For example the expression:

```
Qualify ((StrComp(\Table 1\310 0 N1 Loop\310 0 N1\1 M 98\\, "BY") EQ 0) OR
(StrComp(\Table 1\310 0 N1 Loop\310 0 N1\1 M 98\\, "BG") EQ 0))
```

performs the same set of mapping commands if the value in element 98 is ″BY″ or ″BG″

Qualifying an element by expression

1. For a source based map, right-click the repeating source element in the Mapping Command window pane. For a target based map, right-click the ″For Each″ command underneath the repeating target element. If one does not exist, right-click the repeating target element and select ″For Each″ to create one.

2. Right-click to select Qualify.

3. Click By Expression.

   The Mapping Command Editor displays.

4. Enter a valid expression into the Mapping Command Editor.

5. If you are creating multiple qualifications by expression, click Repeat. Then repeat the previous steps.

   Clicking Repeat creates the qualification in the Mapping Command window pane and redisplays the Mapping Command Editor.

6. When you have mapped the desired number of qualifications of this element, click OK.

   Clicking OK creates the qualification in the Mapping Command window pane and closes the Mapping Command Editor.

Editing an element qualification

Edit a qualified element when you want to change the qualification.

If the element is qualified by occurrence, value, or expression, and you wish to change occurrence, value, or expression qualifications:

1. Double-click the qualify command in the Mapping Command window pane that is to be changed.

Qualification

The Mapping Command Editor displays.

2. Edit the qualification.

3. Click OK.

Changing multi-occurrence qualification on a qualified element:

1. Create a multi-occurrence qualification if one does not already exist.

2. Check that the multi-occurrence qualification command (Default) is selected.

3. Drag repeating element from the source document definition onto a repeating element in the target document definition.

   A MapTo command is created in the Mapping Command window pane under the multi-occurrence qualification command (Default).

4. If the new MapTo command is to replace an existing MapTo command, delete the existing MapTo command by right-clicking on it and selecting Delete.

Specifying Hierarchical Loop Levels

Hierarchical loops are a way for EDI transactions to dynamically define the looping structure. The standard defines all segments which might be a part of a hierarchical loop but it is the application/standard data being received which adds structure to those segments. For example an HL loop would be defined to contain segments A, B, C, D, E; but it is the application/standard data which defines that segments A,B,C form one level 1 inner loop, segments D,E form another level 1 inner loop and further that segments D,E might form a level 2 inner loop within the A,B,C loop. Theoretically there is no limit to the nesting levels.

HL loops may be qualified and mapped using the usual qualification and mapping methods described in this book. However, WebSphere Data Interchange's HL support allows you to define Hierarchical levels and specify unique mapping instructions for each identifiable group of structures in a hierarchical level (loop).

Creating an HL Loop Level

To create an HL loop level as a base level within an EDI source or target message:

1. Right-click the HL Loop in the Mapping Command window pane and select Add HL Qualification. The HL Qualification dialog box displays.

2. Select the HL level code from the drop down list. This will result in an HLLevel() command.

The HLLevel command has the following format:

```
HLLevel(levelcode)
```

where:

levelcode

Indicates the level code for this level.

You must select this value from the drop-down list of level codes that are valid for the code list for HL03 (element 735).

This value identifies the context of a series of segments following the current HL up to the next occurrence of an HL, or the end of the loop.

To insert an HL loop level as a child of the current HL loop level:

1. Right-click the HLLevel Command in the Mapping Command window pane, select HL Qualification, and select AddChild HL Qualification. The HL Qualification dialog box displays.

2. Select the HL level code from the drop down list.

To insert an HL loop level as a peer (sibling) of the current HL loop level:

1. Right-click the HLLevel Command in the Mapping Command window pane, select HL Qualification, and select AddPeer HL Qualification. The HL Qualification dialog box displays.

2. Select the HL level code from the drop down list.

With each HL loop level created, the HL loop is copied to its hierarchical location (as a child or peer) with an HLLevel command and displayed in the command window as if the hierarchy were explicitly defined in the standard.

For more information about HL loops, see Appendix D, "Hierarchical loops" on page 377.

Advanced mapping techniques

WebSphere Data Interchange techniques for using literal keywords and other advanced mapping techniques are documented in Appendix B, "Data Transformation mapping commands and functions" on page 301.

Translation tables

┌─ **WebSphere Data Interchange Terminology Note** ─────────────────┐
│ The terms "EDI standard value" or "trading partner value" used in translation
│ tables refer to the value in the target document. The term "local value" refers
│ to the value in the source document.
└──┘

When WebSphere Data Interchange translates data from your source document to a target document, it can also substitute one value for another. WebSphere Data Interchange substitutes values through translation tables. Use translation tables to handle:

- Differences between your data and your trading partners' data. For example, say your trading partner uses its own numbers for parts you sell. You can set up a translation table to convert the part numbers your trading partner uses to those you use. An example of such a translation table is illustrated in Table 25.

Table 25. Translation table, differences in data

Local Value	Trading Partner Value
GLF8088	FR0100
GLF8588	FR0600
GLF8788	FR0800

- Conflicts between data in your source document and data in your target document are different. For example, say the source document uses an abbreviated unit of measurement that is not the same as in the target document. You can create a translation table to substitute a different code in your target document than is in your source document. An example of such a translation

table is illustrated in Table 26.

Table 26. Translation table, conflicts with standards

Local Value	Standards and Trading Partner Value
Boxes	BX
Cases	CS
Doz	DZ
Each	EA

You associate translation tables with maps through the use of the Translate function. See Appendix B, "Data Transformation mapping commands and functions" on page 301 for more information.

Following are detailed procedures for creating new Translation Tables. For information on viewing, copying, editing, renaming, deleting, and printing translation tables, see "Performing common file management tasks" on page 45. For information on exporting translation tables, see "Exporting" on page 68.

Creating the tables

WebSphere Data Interchange provides forward translation tables in data transformation mapping.

Set up a forward translation table when you want to translate values from the source document into values required by the target document. The translation table contains values with a one-to-one relationship or many local values to one standard or trading partner value. The local value side of the table definition must be unique, as illustrated in Table 27.

Table 27. Sample forward translation table values

Local Value	Standard Value
01	AA
02	BB
03	CC
04	CC

Creating a new translation table

Create a new translation table when you need to substitute values in the target document with values supplied in the source document.

1. In the Mapping Functional Area, click either the Forward Translation tab.
2. Click New on the tool bar.

    The General tab displays.
3. Type in a name for the translation table. This is a required field.

    You can add a more complete description in the Description field if you wish.
4. Select—from the Data Type drop-down list in the Local Variable group box—the type of data in the source document.

    - Select CH if the data is character data.
    - Select R if the data is numeric data.

5. Select—from the Max Length drop-down list in the Local Variable group box—the maximum length of the data in your source document's fields. The maximum supported length is 35 characters, numbered 001 through 035.

6. Select—from the Data Type drop-down list in the Standard or Trading Partner Variable group box—the type of data in the standard or in your trading partner's format.

   • Select CH if the data is character data.

   • Select R if the data is numeric data.

7. Select—from the Max Length drop-down list in the Standard or Trading Partner Variable group box—the maximum length of the data in the standard or in your trading partner's fields. The maximum supported length is 63 characters, numbered 001 through 063.

   Note: The combined lengths for the local variable and the standard or trading partner variable cannot exceed 68 characters.

8. Type the translation table in the grid at the bottom of the tab.

   a. Type the value in your source document in the Local Value column, then press the Tab key.

   b. Type the value in the target document in the Standards or Trading Partner column, then press the Tab key.

9. When you have finished entering all values required in the translation table, click Save on the tool bar to save the translation table.

Specifying map rules

When you have completed a map, you must associate it with a trading partner or trading partners. WebSphere Data Interchange calls those associations rules, map rules, or data transformation map rules.

Applying the minimal trading partners concept

The concept of minimal trading partners attempts to reduce the amount of time spent on administrative functions. The traditional WebSphere Data Interchange was based on the idea that each trading partner would be identified to the product through a trading partner profile and a map rule or usage. Thus, a WebSphere Data Interchange installation with tens of thousands of trading partners would require an equal number of profiles and map rules or usages, even though the options might be identical. The WebSphere Data Interchange concept of generic rules and usages reduces the administrative impact of this model, but does not completely meet all its needs. Some installations do not need a setup for a trading partner because they keep that information in their application and pass it to WebSphere Data Interchange at transformation time. WebSphere Data Interchange uses a combination of techniques and terminology to accommodate this minimal administrative model.

The following procedures can be carried out from the trading partner list windows as well as from the Mapping List window.

Viewing map rules

To view a map rule:

1. In the Data Transformation Maps List window, select the map for which you want to view map rules, or go to that map's editor window.

2. Click View Usages on the tool bar.

WebSphere Data Interchange Client runs a query that displays a window, which contains the Data Transformation Map Rule list window. A list of map rules associated with the map displays.

Creating a new map rule

Create a new map rule after you create a new data transformation map and need to associate an existing trading partner with the map. You can also create map rules to associate trading partners with existing maps.

1. In the Data Transformation Maps List window, select the map for which you want to create map rules, or open the editor for the map.
2. Click View Usages on the tool bar.

   WebSphere Data Interchange Client runs a query that displays a list of map rules associated with that map. If there are no map rules the list window displays empty.
3. Click New on the tool bar.

   The Data Transformation Map Rule Editor displays with the General tab in front.
4. Indicate whether the map rule will be trading partner based or process based.

   Note: WebSphere Data Interchange allows you to categorize trading partners by a "process". You can create a rule between a process (or category) and a map, as well as between a trading partner and a map, to reduce the number of rules needed. For more information, see "Understanding processes and rules" on page 132
5. Enter the Process ID or the sending and receiving trading partners, as required.
6. Complete any desired optional fields.

   Click [?] for optional field names and descriptions.
7. When you have finished entering all the values you require in the map rule, click Save on the tool bar to save the map rule.

Editing map rules

Edit a map rule when you need to change translation specifications.

1. In the Data Transformation Maps List window, click the map for which you want to edit a map rule.
2. Click View Usages on the tool bar.

   A window is displayed containing the Data Transformation Map Rule list window. If you have created map rules for this map, the existing map rules display in the list window.
3. Double-click the map rule you need to edit.

   The general tab displays.
4. Add, change, or delete entries as required.
5. When you have finished entering all values required in the map rule, click Save on the tool bar to save the map rule.

Copying map rules

The copy function allows you to duplicate a map rule within the WebSphere Data Interchange system in which you are working. If you want to base a new map rule on an existing one, for example, copy the existing map rule under a new name and edit it to the new specifications. You can also copy a rule to a different map or to a new trading partner.

1. In the Data Transformation Maps List window, click the map for which you want to create a map rule.
2. Click View Usages on the tool bar.

   A window is displayed containing the Data Transformation Map Rule list window. If you have created map rules for this map, the existing map rules display in the list window.
3. Select the map rule you wish to copy.
4. Select Copy from the Actions menu.

   The Copy Data Transformation Map Rule dialog box displays.
5. Select a map from the Map Name drop-down list provided.
6. Complete any fields you need in the Associated With section.
7. Click OK.

   WebSphere Data Interchange Client copies the map rule under the new name.

For information on editing and deleting map rules, see "Performing common file management tasks" on page 45

Compiling control strings

After you complete a map and associate it with trading partners, you must compile a control string before WebSphere Data Interchange can use the map. WebSphere Data Interchange Client uses the data you created during the mapping process as input to a program that compiles the map to create a control string. The WebSphere Data Interchange Server uses the control string in its translation processing.

While compiling, WebSphere Data Interchange Client checks for errors in the map you created. Error messages are displayed in the Execution Status window. Serious errors are also logged to the Message log.

Compiling a control string is the last thing you do after adding or updating a map. Any time you change a map, you must compile a new control string.

To compile a control string:
1. In the Data Transformation Maps list window, select the map for which you want to compile a control string.
2. Click Compile Control String.

   An Execution Status window displays. Any errors are noted in the window.

   Note: If you are not using WebSphere Data Interchange Client in stand alone mode, export the control string into the WebSphere Data Interchange Server. For more information on exporting and importing, see Chapter 6, "Export and Import" on page 67.

Viewing compiled control strings

Compiled control strings display in the Control Strings list window. Click "Mapping" on the Navigator bar to view the control strings list window. Then click on the Control Strings tab to view the Control Strings list window. Table 28 describes the fields that display in the control strings list window.

Table 28. Control string list window field descriptions

This field. . .	Displays:
Map Name	The name of the map that this control string compiled.

Table 28. Control string list window field descriptions (continued)

This field. . .	Displays:
Gen Date	The date the control string was complied.
Gen Time	The time the control string was compiled.

Defining Global Variables

Global variables are similar to local variables, but they are not unique to any map; they are shared across maps. You can define and maintain a global variable from a map, or from the Mapping Functional Area window, using the Global Variables tab. During a translation session, a global variable will retain its value across document translations. It is not reset unless a mapping command specifically changes the value in the global variable.

To define a global variable:

1. Click the Global Variables tab on the Mapping Functional Area window. The Global Variable list window displays.

2. Complete the required field (the name) and any optional fields on the editor. Required fields are preceded by a blue asterisk.

   Click [?] for field descriptions.

3. Select Save from the File menu to save the variable.

Migrating a map to a different source or target document

The source or target document definition in a map can be changed at any time. This is most commonly done to change from one version of a document definition to another. This occurs commonly with EDI standards. To change the source or target document definition:

1. On the General tab of the Data Transformation Map editor, select a different source or target document from the drop-down menus. When changing the version or release of an EDI standard transaction, usually only the dictionary name must be changed. For more information on using the Data Transformation Map editor, see "Using the Data Transformation Map editor" on page 201.

2. Click Save.

   The Data Transformation Map editor tries to validate existing mapping commands. If the base document definition was changed, the Data Transformation Map editor attempts to locate the appropriate place in the Mapping Commands window pane to migrate the existing mapping commands.

   Any mapping commands that could not be migrated display with a red exclamation mark. You can edit those mappings or drag them to the correct place in the map. Any elements in the source document definition that the map could not identify appear with a red exclamation mark. Any commands under these can be moved or deleted as needed. When you are sure you no longer need the unknown elements that have a red exclamation mark by them, you can delete them by right-clicking and selecting Delete.

Chapter 27. Send and receive mapping

Send maps and receive maps are two of the supported map types in WebSphere Data Interchange. They are primarily used for migration and compatibility with maps from earlier versions of WebSphere Data Interchange. Send maps are a set of mapping instructions that describe how to translate a proprietary application data document into an EDI standard transaction. Receive maps are a set of mapping instructions that describe how to translate an EDI standard transaction into a proprietary application data document.

The map editor

WebSphere Data Interchange Client's Map Editor features a visual means for associating your data fields and records with the EDI standard's elements and segments. The Map Editor uses a split screen that permits you to create map associations.

The data format you created for your application displays on the left side of the screen, and the EDI standard transaction displays on the right. The transaction displays exactly as published, unless you have modified the EDI standard loaded when you set up WebSphere Data Interchange Client.

Starting the map editor

The Map Editor displays when you select a map from the Map List window, as follows.

1. Click Mapping on the WebSphere Data Interchange Client Navigator bar.

   The Map List window displays.

   This window displays a list of existing maps. All the different types of map are listed. Each row contains information about a component; each column contains data stored in that component. Information in the columns displays in fields in the editor window. The list window, however, also contains the date, time, and user ID of the last update.

   To display additional columns, click the scroll bar on the bottom of the screen to scroll to the right or left. To alter the columns that display on the screen, or to change which query is executed to produce the list, click Modify Window Properties (...). To create new queries, refer to "Creating a query" on page 268

2. To view a map or to add or change its information, double-click the row of the map you want to work with.

   The appropriate Map Editor window displays, with the Details tab in front. You add information or make changes to maps through its tabs and related dialog boxes, as described in the following sections.

Using the map editor

The Map Editor allows you to associate your application data with an EDI standard transaction. The editor works by displaying the data format you created on one side of the screen and the EDI standard transaction on the other. The Map Editor allows you to associate components of your data format with the transaction by dragging data format components and dropping them into the correct locations in the EDI standard transaction.

You can map the data format fields to EDI standard data in any of these patterns:

- One field to one data element

219

- Several fields to one data element
- One field to several data elements

The Map Editor window contains three tabs: General, Details, and Comments. Use the:

- General tab to enter and change global map information.
- Details tab to associate components of a data format with components of the EDI standard transaction.
- Comments tab to type any comments you wish about the selected map.

Following are detailed procedures for creating and editing maps. For information on viewing, copying, editing, renaming, deleting, and printing maps, "Performing common file management tasks" on page 45 For information on exporting maps, "Exporting" on page 68

Creating a send or receive map

Create a new map after you have created a data format. In some cases, you might need to create a new map to meet the data requirements of a particular trading partner. You can also create a new map from an existing data format to meet new requirements. The EDI standard transaction you are going to work with must also be loaded into WebSphere Data Interchange Client.

In the most basic sense, creating a new map consists of associating fields in a data format with data elements in a transaction. The following procedure shows how WebSphere Data Interchange Client makes basic associations between fields and data elements.

1. In the Map List window, click New Document on the tool bar.

   The Create Map wizard displays with the Map Name wizard page.

2. Type a map name in the Map Name field.

   You can use both letters and numbers to identify your map. Letters display in capitals. You cannot type spaces within the name.

   You can enter a more complete description of the map in the Description field.

3. Click Next to continue.

   The Map Type wizard page displays. This page allows you to choose the type of map you are creating, from the available WebSphere Data Interchange map types, by selecting the appropriate radio button.

4. Click the appropriate radio button for a send or receive map, and then click Next.

   The appropriate Source Dictionary wizard page displays. This page contains a list of data format dictionaries if you are creating a send map. The page contains a list of EDI standard dictionaries if you are creating a receive map.

5. Select the dictionary for the source document, and then click Next.

   The wizard page listing the available document definitions within the selected dictionary displays. For send maps, this is a list of data formats within the selected data format dictionary. For receive maps, the list contains all transactions within the selected EDI standard dictionary.

6. Select the document definition to be used as the source document definition in your map, and click Next.

   If this is a send map, the Target Dictionary wizard page displays. This page contains a list of EDI standard dictionaries. Proceed to step 8.

If this is a receive map, the Compliance Checking wizard page displays. This page offers you the choice of checking all EDI Standard Data for Compliance to the EDI standard, or only checking mapped data. Proceed to step 7.

7. Indicate which type of compliance checking you want on this map, then click Next.

   The Target Dictionary wizard page displays. This page contains a list of data format dictionaries.

8. Select the dictionary that contains the target document definition you intend to use in your map, and then click Next.

   The wizard page listing the available document definitions within the selected dictionary displays. For receive maps, this page contains a list of data formats within the selected data format dictionary. For send maps, the page contains a list of all transactions within the selected EDI standard dictionary.

9. Select the document definition to be used as the target document definition in your map, and then click Next.

   The Confirmation window displays.

10. Confirm the selections displayed. If correct, click Finish to save the information.

    After the information has been saved, the Create Map wizard closes and the Map Editor displays with the Details tab in front. The Details tab allows you to drag components of your data format and drop them on data elements within the transaction.
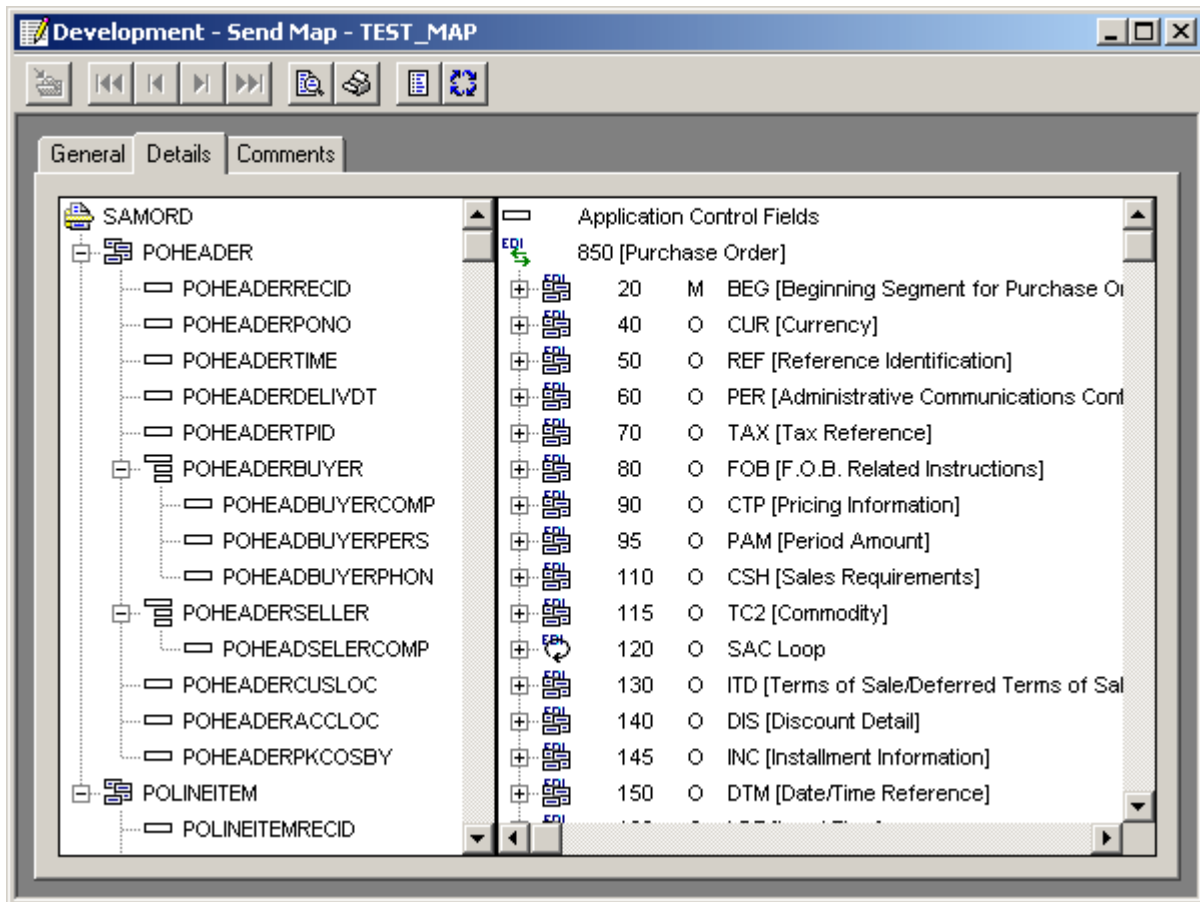

Figure 10. Send and Receive Map Editor

Creating a send or receive map

> **Note:** The Direction group box displays grayed. It is for information purposes only. You cannot change the direction of a map.

a. Click the plus (+) sign next to the component (a loop or segment) in the EDI standard transaction you wish to associate with the data format component. The transaction displays on the right side of your screen.

If you clicked a loop, you see the segments and nested loops that comprise the loop. If you clicked a segment, you see the data elements and composite data elements that comprise the segment, as shown above.

By default, all components of a data format are expanded to show their fields. If you wish to close any portion of the data format in order to see more of it on the screen, click the minus (-) sign.

b. Click the field you wish to map on the left side of the screen. While holding down the mouse button, drag the field to the corresponding data element of the EDI standard transaction on the right.

The cursor changes to an arrow connected to an outline of the component you are dragging.

For a list of mapping components and their associated graphics, see Table 29 on page 223.

When you have dragged the field to the right side of the screen over the data element with which you want to associate it, that component becomes highlighted. Release the mouse button.

> **Note:** If the data element does not become highlighted when you move the cursor on top of it, that means it is not a valid place to drop a field. For example, you cannot drop a data format record onto a data element in an EDI transaction.

The Mapping Data Element Editor might display (see Note below). An element mapping displays below the data element, as illustrated.
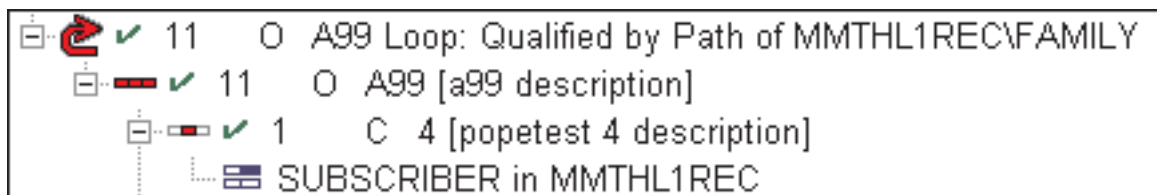
Figure 11. Data element mapping

After you have mapped a data element, a green check mark displays next to the data element to which you have mapped the field. A green check mark also displays next to the segment containing that data element so that you know the segment contains mapped components.

> **Note:** This procedure presents a simple mapping situation, in which you associate data format fields with EDI standard data elements. For anything but the most basic mapping associations, you must use WebSphere Data Interchange's advanced mapping capabilities. For more information on WebSphere Data Interchange's advanced mapping capabilities, "Using the mapping data element editor" on page 224

11. Continue dragging and dropping fields onto data elements until you have mapped all of the information required for this map.

12. Enter any comments on the map in the Comments tab.
13. When you have completed mapping, click Save on the tool bar to save the map.

Editing a map

Edit a map when you have modified its associated data format. You may also need to edit a map to meet specific requirements of a new trading partner.

1. In the Map List window, double-click the map you wish to edit.

   The map displays in the Map Editor window with the Details tab in front.

   Any changes you made earlier to the map or its associated items display on the screen. Any changes you made to the data format display on the left side of the screen. If you made any changes to the data format or EDI standard since the last time you opened the map, you see those changes on the Details tab.

2. Create new associations between the data format and the transaction or change existing ones.
   - To make an association between a new data format component and a EDI standard component, drag the data format component and drop it on the proper EDI standards component.
   - To edit an existing association, double-click the element mapping below the data element. The Mapping Data Element Editor displays.
   - To delete an association, click the element mapping you want to delete and press the Delete key.
   - You can also change the order of multiple element mappings by clicking on the element mapping you wish to move and dragging it to its new location.

3. Change information as required in the General tab.

   You may not change the direction of a map. Create a new map if you need to turn a send map into a receive map or vice versa.

4. Click Save on the tool bar to save the map.

Table 29. Mapping components and their graphics

This icon. . .	Represents a:
	Data Format
	Record
	Field
	Structure
	Transaction
	Loop
	Segment
	Composite Data Element
	Data Element

Using the mapping data element editor

The Mapping Data Element Editor allows you to apply WebSphere Data Interchange Client's advanced mapping capabilities to data elements and fields. You can:

- Use an accumulator to store, count, and add values to a field or data element. Accumulators allow you to perform such actions as counting segments in a transaction for later placement in a transaction's trailer record. For more information on accumulators, see "Using accumulators" on page 233.

- Associate a WebSphere Data Interchange literal or other mapping commands with a field or data element. Literals allow you to perform such actions as providing data to trading partners that your application does not contain. For more information on literals, see "Using literals and mapping commands" on page 235.

- Use any of WebSphere Data Interchange's special handling options on a field or data element. Special handling options allow you to perform such actions as editing dates, verifying data in a field against predefined lists, and converting data from one value to another.

To apply advanced mapping capabilities to a field or data element:

1. Double-click a data element in an EDI transaction that has not been mapped, or if the element has been mapped, then double-click one of the mappings below it.

   If this is a receive map, the Qualified Element Support dialog box displays. Select Qualified Element Support or normal support, as appropriate. If this is a send map, go to step 3.

2. Click Normal.

   The Mapping Data Element Editor displays.

3. Select the mapping capability you want to apply to this field or data element. You may use more than one.

   - To use a literal, type any literal or mapping commands you desire to use on the field or data element in the Literal or Mapping Commands field. See "Using literals and mapping commands" on page 235 for more information.

   - To use an accumulator, select an accumulator from the Accumulators drop-down list. For each accumulator, you must select an action from the Actions drop-down list. You may use up to four accumulators on any data element or field. See "Using accumulators" on page 233 for more information.

   - To use a special handling option on a field or data element, click Special Handling. See "Using the special handling button" on page 225 for more information.

   - To create another element mapping for this data element or field, click Repeat. See "Using the Repeat Button" on page 226 for more information.

4. If you wish, you can view attributes of the data element and field you are mapping.

   - To view the attributes of the data element on which you are working, click Element Attributes.

     The Data Element Attributes dialog box displays. You see the same information about the data element that displays on the General tab of the Data Elements Editor.

   - To view the attributes of the field on which you are working, click Field Attributes.

The Field Attributes dialog box displays. You see the same information about the field that displays on the General tab of the Data Format Field Editor.

5. Type any comments on the mapping in the Comments field.

6. Click OK to save the mapping.

Using the special handling button

Use the Special Handling button when you want to request special handling on data elements and fields for both send and receive maps. You can:

- Change date formats from one format to another.
- Verify field values against predefined lists.
- Translate field values from one predefined value to another.
- Call an external program to perform custom processing on the value contained in a field or data element.
- Specify the length and position of fields when mapping a concatenation for send maps or specify the length and position of fields that are included in a substring for receive maps.

To request special handling options:

1. While working on a data element or field in the Mapping Data Element Editor, click Special Handling.

   The Data Element Special Handling dialog box displays.
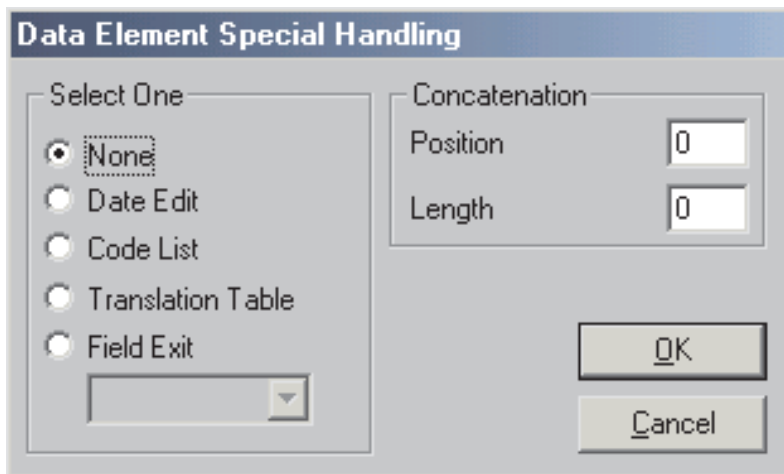
Figure 12. Data Element Special Handling dialog box

2. Click the option corresponding to the Special Handling option you desire.

   Click [?]
   for more information about the options.

3. Select the item you need from the drop-down list.

4. Click OK to save your options.

   Note: You may use more than one Special Handling option on any field by using repeat mapping. Click Repeat to create another instance of the element mapping you are working with. Edit that instance with the Mapping Data Element Editor using the Special Handling button again to select another option.

Using the Repeat Button

Use the Repeat button when you want to duplicate the element mapping you completed in the Mapping Data Element Editor. One reason to duplicate an element mapping is to apply more than one special handling option for the data element or field. For example, if you want to use more than one literal command on a field, you need to repeat the element mapping as you can only use one literal command per element mapping.

You can also use repeat mapping to add data to your application when it does not occur in the EDI standard and vice versa. The following example shows how you can add data to your application that does not occur in the EDI standard through a receive map using repeat mapping.

Say that an EDI transaction contains a purchase order total in a particular data element in a particular segment. A purchase order total less than $100 is an error that you want to report. You can use the Repeat button to repeat the element mapping and enter the mapping commands necessary to accomplish this task, as follows.

To repeat an element mapping:

1. Double-click the data element that contains the purchase order total.

   The Mapping Data Element Editor displays.

2. Enter in the Literal or Mapping Command field the mapping command &SAVE TOTPO to save the data element value in variable TOTPO.

3. Click Repeat.

   The element mapping is saved and another element mapping is created. The Mapping Data Element Editor redisplays with the new element mapping. The saved element mapping appears under the associated data element in the Map Editor.

   You may select any options you desire for the new element mapping.

4. Enter the mapping command in the Literal or Mapping Command field to check if the value saved is less than 100 and to issue a user error.

   &IF (TOTPO < 100) &ERR(1,1,0,"PO is less than $100.00").

5. Click OK and finish editing the map.

   The new element mapping is designated Literal of (the entered mapping command).

Setting an application control key

To make it easier to find a series of documents in WebSphere Data Interchange's Transaction Store database, you can set up keys in your maps using application control fields. The application control field contains an ID used to identify the document in the transaction store, such as the purchase order number in a purchase order document. The Application Control Fields dialog box allows you to select up to eight fields, literals and variables to used to construct the application control field.

For example, say your company has a string of characters that occur in front of a purchase order number. Your trading partners likely send only the number, but you can use the Application Control Fields dialog box to have WebSphere Data Interchange add the characters you desire to their values. You can also use WebSphere Data Interchange variables to modify data in the application control field.

To make documents received from or sent to a particular trading partner easier to find in the Transaction Store, you could add a string of characters representing that company to the application control field. That character string becomes something you can search on to build a list of documents when viewing the Transaction Store.

For more information on viewing the Transaction Store, Chapter 32, "Transaction store" on page 279. For more information on how the Transaction Store works, see the *WebSphere Data Interchange for z/OS Administration Guide*.

Note: Application control fields are optional; the translator does not require them.

To set up application control fields:

1. Click the Details tab of the map for which you want to set up application control fields.

2. On the data format (left) side of the screen, click the field you want to set as an application control field and hold down the left mouse button. Drag the field over the Application Control Fields section of the transaction (right) side of the screen, and release the mouse button.

   The Application Control Fields dialog box displays.

   The Path column contains the name of the data format loop(s), record and (possibly) structures in which the field you dragged occurs. The Field column contains the name of the field. The Length column contains the length of the field.

   You can add as many as eight fields to the dialog box.

3. If you want to add fixed data to the application control fields, click Add &LIT.

   The Application Control Fields - &LIT Support dialog box displays.

   a. Type in the Enter Literal field the data you wish to add to the application control field.

   b. Type in the Enter Length field the length of the data you wish to add to the application control field.

   c. Click OK.

      The data you typed displays in the Field column in a row below the field to which you added it. The data is preceded by &LIT.

4. If you want to include the data from a WebSphere Data Interchange variable, click Add &VAR.

   The Application Control Fields - &VAR Support dialog box displays.

   a. Type in the Enter Variable field the name of the WebSphere Data Interchange variable you wish to use.

   b. Type in the Enter Length field the length of the data you wish to add to the application control field.

   c. Click OK.

5. When you have completed your setup, click OK.

   The Details tab of the Map Editor redisplays.

Specifying qualification

Within the EDI standards, certain segments and groups of segments, called loops, can repeat. Some EDI standards also support data elements or composite data elements that can repeat.

Qualifying

When you map a segment, loop, or data element that repeats within an EDI standard, you must tell WebSphere Data Interchange which occurrence of the segment, loop, or data element you are using. This is called "qualifying" the segment, loop, or data element.

WebSphere Data Interchange supports three types of qualification for send and receive map types. You can qualify by:

- Occurrence
- Path
- Value (loops and repeating segments on receive maps only)

Non-repeating data elements can also be qualified by value on receive maps.

When you qualify by occurrence, your mapping is related to the position of data in a repeating sequence. When you qualify by path, your mapping is related to a specific structure or record in the data format and how it handles repeating segments or data elements in the EDI standard. When you qualify by value, your mapping is related to the value that you receive in a data element.

Note: You can mix types of qualification for the same loop, segment, or data element. Path qualification works with Value qualification on receive. Occurrence works with Path on both send and receive. You cannot mix Occurrence and Value qualification.

Qualifying loops and segments

WebSphere Data Interchange allows you to qualify loops and segments by occurrence, path, and value.

Qualify by occurrence when the order in which repeating data occurs in either the data format or EDI standard transaction is important. For more information, see "Qualifying a loop or segment by occurrence".

Qualify by path when you want to create multiple segments in an EDI standard transaction to correspond to multiple occurrences of a record or structure in a data format and vice versa. For more information, see "Qualifying a loop or segment by path" on page 229.

Qualify by value when you want to specify data received in a standard transaction to trigger the creation or population of fields in your data format. For more information, see "Qualifying a loop or segment by value" on page 230.

Qualifying a loop or segment by occurrence

Qualify a loop or segment by occurrence when a specific instance of the segment or loop must be mapped to a specific component of the data format. For example, say that you are working on a send map and need to send two addresses, the send-to address and the bill-to address. Map the ship-to address to the first occurrence of the N1 segment and the bill-to address to the second occurrence of N1.

Attention: Qualification by occurrence creates records or segments first and then looks for the data to fill them. If you use occurrence qualification on a receive map, then you need to move data to that record because WebSphere Data Interchange will create the record, and it may not contain any data.

WebSphere Data Interchange Client's mapping function automatically qualifies loops and segments that repeat. When you drop a field onto a data element that occurs in a loop or a repeating segment, the loop or segment is automatically qualified by occurrence; the title of the loop or segment changes after the drop to include the words, "Qualified by Occurrence #1". Use the Qualify a Loop or Segment dialog box, as follows.

To qualify a loop or segment by occurrence:

1. Double-click a loop or repeating segment (a segment that has a maximum use value greater than one).

   For receive maps, either the Qualify a Loop or Qualify a Segment dialog box displays. For send maps, proceed to step 3.

   Note: You may also drop a field onto a data element in a loop or repeating segment. The loop or segment is automatically qualified by occurrence. See "Editing a loop or segment qualification" on page 230 to change the occurrence number.

2. Click Occurrence.

   Either the Qualify a Segment by Occurrence dialog box or the similar dialog box for loops displays. The name of the loop or segment displays in the title bar.

3. Type in a number in the Enter the Occurrence Number field that corresponds to the occurrence number of the component you are mapping.

   If you are mapping the second occurrence of a field in your data format to an EDI standard transaction, type 2 in the Enter the Occurrence Number field.

4. If you need to map additional occurrences, click Repeat.

   The qualification is added to the map and displays on the transaction side of the Map Editor. The Qualify a Segment by Occurrence Number dialog box or its corresponding loop dialog box redisplays with the next available number in the Enter the Occurrence Number field.

5. When you have specified the desired number of occurrences, click OK.

   The qualification is added to the map and displays on the transaction side of the Map Editor.

Qualifying a loop or segment by path

Qualify a loop or segment by path when you need WebSphere Data Interchange to create multiple instances of either:

- a record or structure for receive maps
- a segment or loop for send maps

For example, say you are working on a receive map and find that the PO1 segment in a purchase order repeats to handle multiple purchase-order line items. You need WebSphere Data Interchange to create a separate record for each instance of the PO1 segment.

Consequently, you would qualify the PO1 segment by path. That way, WebSphere Data Interchange will create as many line-item records in your application data as there are occurrences of PO1 in your trading partner's transaction.

Note: Qualification by path finds loops or repeating segments and creates records for them on receive and vice versa on send. Use path qualification when the number of records WebSphere Data Interchange may need to create is unknown.

WebSphere Data Interchange Client's mapping function automatically qualifies loops and segments that repeat. When you drop a record or structure onto a loop or repeating segment, the loop or segment is automatically qualified by path; the title of the loop or segment changes after the drop to include the words, "Qualified by Path of (Path Name)," as follows.

To qualify a loop or segment by path:

1. Drag a record or structure onto a loop or repeating segment (a segment that has a maximum use value greater than one).

   The title of the loop or segment changes to include the words, "Qualified by Path of (Path Name)."

Qualifying a loop or segment by value

Qualify a loop or segment by value when you want the value of data received in a data element to drive WebSphere Data Interchange's translation of a whole loop or segment. Qualification by value is not supported on send maps.

For example, say you want to qualify the N1 loop with the value of BY in Element 98 (Entity Identifier Code) received in a purchase order to create a buyer record. Further, say that you want the buyer's name to be mapped into the buyer record depending on the value in Element 98 of the N1 loop.

To handle that case, you would map the segment by value. That way, WebSphere Data Interchange will put specific information into the buyer records it creates from the purchase order depending on the name in each order's Entity Identifier Code.

To qualify a loop or segment by value in receive maps:

1. Double-click a loop or a repeating segment (a segment that has a maximum use value greater than one).

   The Qualify a Loop or Qualify a Segment dialog box displays.

2. Click Value.

   Either the Qualify a Segment by Element dialog box or a similar dialog box for loops displays. The name of the segment or loop displays in the title bar.

3. Select from the Select an Element drop-down list the name of the data element by which you want to qualify the loop or segment. The list includes all data elements in the segment or in the first segment within a loop.

4. In the Enter a Qualifying Value drop-down list, select a value or type in the value you want to qualify the loop or segment on. The list will contain values when a code list is associated with the selected data element.

5. If you are mapping multiple occurrences, click Repeat.

   The qualification is added to the map and displays on the transaction side of the Map Editor. Then either the Qualify a Segment by Element dialog box or the similar dialog box for loops redisplays. Repeat Step 4 and Step 5.

6. When you have mapped the desired number of occurrences of this segment, click OK.

   The qualification is added to the map and displays on the transaction side of the Map Editor.

Editing a loop or segment qualification

Edit a qualified loop or segment when you want to change the default qualification that WebSphere Data Interchange Client places on a segment when you use drag-and-drop mapping of fields, structures, and records.

To edit a qualified loop or segment: If the Loop or Repeating Segment is Path Qualified and you wish to change to or add occurrence or value qualifications:

1. Double-click the qualified loop or repeating segment.

   The Qualify a Segment dialog box displays.

2. Click New or the Replace button, depending on whether you want to add another qualification or replace the existing qualification.

   If this is a receive map and you have no other qualifications for this loop or repeating segment, the Qualify a Loop or Qualify a Segment dialog box displays with the choices Value, Occurrence, and Cancel. Continue to step 3.

   If other qualifications exist for the loop or segment, then additional qualifications must be of the same type. The Qualify a Loop by an Element, Qualify a Segment by Element, Qualify a Loop by Occurrence Number, or Qualify a Segment by Occurrence Number dialog box displays directly in this case.

   For send maps, the Qualify a Loop by Occurrence Number or Qualify a Segment by Occurrence number displays. Skip to step 4.

3. Click Occurrence or Value (if given the choice) depending on how you want to qualify the loop or segment.

4. When you have qualified the desired number of occurrences of the loop or segment, click OK.

Changing path on qualified repeating element:

1. Drag a record or structure onto the loop or repeating segment.

   The Qualify a Segment or Qualify a Loop dialog box displays.

2. Click New or Replace, depending on whether you want to add another qualification or replace the existing qualification.

   Note: For receive maps, you may only have one path qualified mapping for a loop or segment. You will be issued an error message if you attempt to create a second path qualified mapping.

Qualifying data elements

Nonrepeating data elements can be used to qualify by value other nonrepeating data elements. Repeating data elements can be qualified by occurrence number or path. A repeating data element can be qualified by both an occurrence number and a path.

Qualify data elements by value when you receive data elements that have qualifiers in a segment. In such cases, you may need to qualify how WebSphere Data Interchange handles each occurrence of the data element. Data element qualification by value is not supported on send maps.

If data you receive from a trading partner contains many units of measure, (each, case, for example), you might need to qualify each data element. By doing so, WebSphere Data Interchange can translate data in the data elements into a single unit of measure for your application.

When you qualify a repeating data element by occurrence, your mapping is related to the position of data in a repeating sequence. When you qualify a repeating data element by path, your mapping is related to a specific structure in the data format and how it handles repeating data elements in the EDI standard.

Qualifying a data element by value in receive maps: Use the Add an Element Qualification dialog box to qualify a data element by the value of another data element, as follows.

Qualifying

1. Double-click the data element in the segment that will be used to qualify one or more data elements in the segment. Make sure you qualify data elements before you map them.

   The Qualified Element Support dialog box displays.

2. Click Qualified.

   The Add an Element Qualification wizard displays.

   If you click Normal, the Mapping Data Element Editor displays. For more information, see "Using the mapping data element editor" on page 224

3. Select the data element or data elements you want to qualify.

   a. Select the data element or data elements in the Select Elements group box.

   b. Click > to move the selected data element or data elements to the Qualified Elements group box.

      Clicking >> moves all data elements in the list.

      Clicking < or << removes the selected element or all data element(s) from the Qualified Elements group box.

4. Click Next.

   The second page of the Add an Element Qualification wizard displays.

5. Select or enter a qualifying value. Values will appear in the drop-down list if a code list is associated with the qualifying element.

   a. Click the value or values from the Enter a Value drop-down list. You can also type a value in the list.

   b. Click > to move the value to the Qualifying Values group box.

      Clicking >> moves all values in the list.

      Clicking < or << removes the selected value or all values from the Qualifying Values group box.

6. Click Finish.

   An element mapping icon (or icons, if you selected several values) displays below the data element or data elements you qualified and below the qualifying data element. The mapping element is designated Not Mapped - Qualified by Element in Position x with a Value of y where x is the position of the qualifying data element in the segment and y is the value you selected.

Editing qualification of a data element qualified by value:

1. Double-click the qualifying data element if you want to add additional values.

   The Update an Element Qualification dialog box displays.

2. Select or enter a new qualifying value. Values will appear in the drop-down list if a code list is associated with the qualifying element.

   a. Click the value or values from the Enter a Value drop-down list. You can also type a value in the list.

   b. Click > to move the value to the Qualifying Values group box.

      Clicking >> moves all values in the list.

      Clicking < or << removes the selected value or all values from the Qualifying Values group box.

3. If you want to see all information on the qualified element, click Previous Selected Info.

   The Previous Element Qualification Data dialog box displays, displaying all data elements qualified by this element and the values used to qualify those elements.

4. Click OK.

An element mapping icon (or icons, if you specified several values) displays below the data element or data elements you previously qualified. The mapping element is designated Not Mapped - Qualified by element in Position x with a Value of y where x is the position of the qualifying element in the segment and y is the value you specified.

Qualifying a repeating data element or composite data element by occurrence: Qualify a repeating data element or composite data element by occurrence when a specific instance of the data element must be mapped to a specific component area of the data format.

1. Double-click the repeating data element or composite data element (maximum use value greater than one).

   The Qualify an Element by Occurrence dialog box displays.

2. Type in a number in the Enter the Occurrence Number field that corresponds to the occurrence number of the data element you are mapping.

   If you are mapping the second occurrence of the data element, type 2 in the Enter the Occurrence Number field.

3. If you need to map additional occurrences, click Repeat.

   The qualification is added to the map and displays on the transaction side of the Map Editor. The Qualify an Element by Occurrence Number dialog box dialog box redisplays with the next available number in the Enter the Occurrence Number field increased by one.

4. When you have specified the desired number of occurrences, click OK.

   The qualification is added to the map and displays on the transaction side of the Map Editor.

To qualify a repeating data element or composite data element by path:

1. Drag a structure onto a repeating data element (a data element that has a maximum use value greater than one).

   The title of the data element or composite data element changes to include the words Qualified by Path of (Path Name).

To change path qualification on a qualified repeating data element:

1. Drag a structure onto the loop or repeating segment.

   For send maps, the Qualify an Element dialog box displays. Move to step2.

   For receive maps, you may only have one path qualified mapping for a repeating data element. Dragging a structure onto an existing path qualified repeating data element results in the qualification being replaced by the new path qualification.

2. When the Qualify an Element dialog box displays, click New or Replace, depending on whether you want to add another qualification or replace the existing qualification.

Using accumulators

Accumulators are special fields that keep running totals or accumulate numeric data. Accordingly, WebSphere Data Interchange's accumulators can add data to an EDI transaction that does not occur in an application database and vice versa. WebSphere Data Interchange Client supports global and transaction accumulators. The scope of a global accumulator is an entire translation session. The scope of a transaction accumulator is a single transaction.

Accumulators

You can use accumulators to count the occurrences of a repeated event, and you can use them to total field values for control purposes. For example, an accumulator can count the detail line items in a payment order to provide a hash total required by the EDI transaction. Or you could use an accumulator to total a field named QUANTITY to cross check the amount of invoices paid.

Accumulators can apply to individual transactions or to all transactions in a translation session. To map both an accumulator and a received value for the same element, use the Repeat action to create another occurrence of the element mapping. Then map one occurrence from the data element to a field and the other occurrence from the accumulator to a field. Each element mapping can support up to four accumulators.

Adding an accumulator to a map

You set up an accumulator through the Mapping Data Element Editor, as follows.

1. Double-click an element mapping in an EDI transaction that has been mapped to a field from a data format.
2. Double-click a data element in an EDI transaction that has been mapped to a field from a data format or on a field that has been mapped onto a data element.

   The Mapping Data Element Editor displays.
3. Select an accumulator from the Accumulators drop-down list.

   Accumulator values are described in Table 30.
4. Select an action for the accumulator from the Actions drop-down list.

   Only actions that are valid for this data element or field display in the list. Actions are described in Table 31 on page 235.
5. Complete any other mapping you need from the Mapping Data Element Editor and click OK.

For outgoing data, accumulator actions are not processed unless:
- Data is generated for the data element or segment, or
- The accumulator is mapped.

For incoming data, the accumulator actions are not processed unless:
- The data element associated with the accumulator is received, or
- The accumulator is mapped and at least the segment containing the data element is received.

Accumulator types

WebSphere Data Interchange supports transaction accumulators and global accumulators, as follows.

Table 30. WebSphere Data Interchange accumulator types

Name	Type	Description
T0-T9	Transaction Accumulator	Applies only to one transaction and are reset at the beginning of each transaction. You can use a maximum of 10 per transaction. Each accumulator holds a maximum of 31 binary digits.
G0-G9	Global Accumulator	Applies to entire translation session and are reset at the beginning of each translation session. You can use a maximum of 10 per session. Each accumulator holds a maximum of 31 binary digits.

Accumulator actions

WebSphere Data Interchange supports the following accumulator actions.

Table 31. Accumulator actions

This action. . .	Does this:
Increment the accumulator	Adds 1 to the value stored in the accumulator.
Map the accumulator	Maps the value stored in the accumulator to the data element for send maps and to the application field for receive maps.
Zero the accumulator	Sets the accumulator value to 0.
Map the accumulator and then increment it	Maps the value stored in the accumulator to the data element for send maps and to the application field for receive maps, and then adds 1 to that value. Only one accumulator may be mapped on any given mapping.
Increment the accumulator and then map it	Adds 1 to the value stored in the accumulator, and then maps the value stored in the accumulator to the data element for send maps and to the application field for receive maps.
Add to the accumulator and then map it	Adds the value of an element or field, and then maps the value stored in the accumulator to the data element for send maps and to the application field for receive maps.
Map the accumulator and then add to it	Maps the value stored in the accumulator to the standard data element for send maps and to the application field for receive maps, and then adds the value of a data element or field to it.

Using literals and mapping commands

WebSphere Data Interchange literals and mapping commands let you add data to a transaction that is not contained in your business application for send maps, or put data into your application data that is not contained in the transaction. When your application does not have specific information required by the transaction, or when you need to pass specific information, you can map a literal to the data element. Conversely, when your application requires data that is not specified in the transaction, you can supply the information by mapping a literal.

A literal is a value that you specify for the field or data element. The value can be a constant or it can be calculated by any of several WebSphere Data Interchange expressions.

A mapping command is a WebSphere Data Interchange command that begins with an ampersand (&). For a list of mapping commands, search for the key "mapping commands" in WebSphere Data Interchange Client Help.

Adding a literal or mapping command to a map

You can add literals or mapping commands to maps through the Mapping Data Element Editor, as follows.

1. Double-click an element mapping that has been mapped to a field from a data format, or double-click on a data element that has not been mapped. If this is a receive mapping, the application data field is required for a literal.

   The Mapping Data Element Editor displays.

2. Type the name of the literal and any required expressions. Note that literals are case-sensitive. Mapping commands and variable names are not.

3. Complete any other mapping you require from the Mapping Data Element Editor, and click OK.

Literals and data types

The following apply to literals used in both send and receive maps:

- For data types BN, Bn, HX, IT, In, Ln, PD, Pn, Zd, and Zn, the translator converts the literal before placing it in the outgoing EDI standard data or the incoming application data. In send maps, it converts the literal to character data. In receive maps, it converts the literal to the application data type. For example, if the data type is BN, the translator converts the literal to binary and then moves it to the application field.

- Do not type a decimal point when one is implied. To use a default value of 9.99 for a field defined as P2 (packed number with two implied decimal positions), enter the literal as 999.

- Literal values for hexadecimal fields are hexadecimal strings. For example, if the application field is defined as a one-byte hexadecimal field and you want to use a default value of X'FF', enter FF as the value of the literal. For incoming data, the translator converts each two bytes of literal value to a single byte of application data.

    Attention: Type hexadecimal numbers using their EBCDIC values, not their ASCII values.

- Specify a literal value of zero to move this value into a data element. WebSphere Data Interchange generally removes leading zeros from application data so that a field containing all blanks or all zeros will result in no value for the data element. A value of zero is treated the same as all other literal values when determining if a segment should be created. The &ZEROSIG special literal can also be used to indicate that zeros within the application field are significant.

- In translation and validation tables, enter numeric values left-justified and formatted according to the data format. For example, if the data format defines a field as R2, enter the value 7 as 7.00 or the value 7.1 as 7.10.

    Note: Before creating a new map for a specific trading partner, check to see if you can use WebSphere Data Interchange's advanced mapping functions to meet a trading partner's special processing requirements. You can set up a map so that WebSphere Data Interchange meets a trading partner's custom data and processing requirements. For more information, "Using the mapping data element editor" on page 224

Advanced mapping techniques

WebSphere Data Interchange techniques for using literal keywords and other advanced mapping techniques are documented in Appendix C, "Advanced send and receive mapping" on page 337.

To make mapping easier, WebSphere Data Interchange Client Help contains syntax for literals, mapping commands, and operators.

Click ? for more information about mapping.

Table 32. WebSphere Data Interchange mapping variables for Set Command

In this field. . .	Type:
DIAPPFILE	This variable may be used to change the name of the file to which the application data will be written during a Receive Translate. It will override any value that was used in the receive usage or in the data format definition. It provides the capability for data that is being received to influence the final destination for the data. For example, the statement: &SET DIAPPFILE 'SPECIAL' would force the current application data to be written to the application file identified by the DDNAME SPECIAL.
DIAPPTYPE	This variable sets the application file type that corresponds with the file name provided by DIAPPFILE.
DIAUTOCC	This variable may be used to allow automatic century manipulation for both send and receive translation. Century can be automatically added to or removed from the date using the length of the data element or application field. For example, the statement: &SET DIAUTOCC 1 would allow the translator to add or remove the century to the date depending on the length of the data element or application field.
DICCCTRL	This variable may be used to set the century control year. The default WebSphere Data Interchange uses is 10. This means, if year is greater than 10, then century is assumed to be 19, otherwise century is assumed to be 20. For example, the statement: &SET DICCCTRL 95 would force WebSphere Data Interchange to interpret year as follows, if year is greater than 95, then century is assumed to be 19, otherwise century is assumed to be 20. This variable overrides the Application Defaults profile's Century Control Year field.
DICUSERDATA	This variable is a reserved variable. When this variable occurs in a map, the mapped value will be moved to the associated field in the TRCB. This variable is valid only on receive maps.
DIERRFILTER	This variable can be used to control which errors are actually meaningful to you at a point in time during a translation. A description of the error filter can be found in the WebSphere Data Interchange Programmers Reference under the section "Error Filtering" and under the description for "ERRFILTER".
DIEXPTRACE	This variable, when given a nonzero value (&SET DIEXPTRACE 1), causes WebSphere Data Interchange to create a TRACE of the results of all expression evaluations. When tracing is active, WebSphere Data Interchange will write out message TR0411 to the PRTFILE for each expression. The message will show the expression being evaluated and the result of the evaluation. Tracing will remain active until the DIEXPTRACE is given a zero value (&SET DIEXPTRACE 0).

Advanced mapping

Table 32. WebSphere Data Interchange mapping variables for Set Command (continued)

In this field. . .	Type:
DIMAPCHAIN	This variable can be used when an inbound transaction must be translated into more than one output document. It provides a way for more than one map to be executed for the specified transaction. The last value given to DIMAPCHAIN in a map will establish the application sender ID value that will be used to locate the next map to execute. For example, if MAPABC had this coded: &SET DIMAPCHAIN APPLICATIONB the inbound transaction would be translated using map MAPABC, and then it would be translated using the map that is associated with application sender ID APPLICATIONB. The DIMAPCHAIN command will complete translation under the current map before retranslating using the next map indicated by DIMAPCHAIN, whereas the DIMAPSWITCH command will stop translating the map that has the DIMAPSWITCH variable in it, and literally switch to the new map indicated in the command.
DIMAPSWITCH	This variable can be used when data being received needs to be inspected before it can be determined exactly what map should be used against the transaction. It allows you to switch the map that is being executed dynamically based on the data that is being received. A map could be created to initially look at the data being received. Only those data elements necessary to make a map decision would be mapped. WebSphere Data Interchange would determine the real map to be used by interpreting values resulting from conditional logic expression. For example, a map would contain conditional logic expression: &IF(X > Y) &SET DIMAPSWITCH APPLICATIONA Here, if X is greater than Y, the map identified with an application sender ID value of APPLICATIONA would be used to translate the transaction.
DIPROLOG	This variable allows you to override the default XML prolog if you are translating from data format to XML using a send map. Note that data transformation maps are the preferred way to translate to XML. However, send maps are still supported so that XML maps created with WebSphere Data Interchange 3.1 can continue to be used until they are migrated to data transformation maps.
DISAPSEQ	This variable can be used to allow saving of the SAP IDOC record sequence number on the first error encountered during outbound processing. The sequence number may be provided through the application or using the WebSphere Data Interchange accumulators. Variable DISAPSEQ is captured in the SAP status record to indicate first record in error. For more information, see the *WebSphere Data Interchange Programmer's Reference*.
DIVALLEVEL	This variable can be used to control the level of validation done. It can have the same values as the validation level specified in a usage record, which are: 0 (no validation), 1 (validation tables activated), and 2 (validation tables plus type checking). Any other value other than 0, 1, or 2 will be treated as a 0.

Table 32. WebSphere Data Interchange mapping variables for Set Command (continued)

In this field. . .	Type:
DIVALTYPE	This variable can be used to control the data types for which data type checking is done (validation level of 2). The types that may be specified are DT, TM, N, R, CH, AN, A, and HX. They must be specified in uppercase and separated by a comma. Any value specified that is not in the list above will be ignored. For example, to activate DT, TM and HX validation, the following could be done: &SET DIVALTYPE DT,TM,HX
DIVARTRACE	This variable, when given a nonzero value (&SET DIVARTRACE 1), causes WebSphere Data Interchange to create a TRACE of all accesses to variables. When tracing is active, WebSphere Data Interchange will write out message TR0410 to the PRTFILE for each variable access. The message will indicate the variable being accessed and its current value. Tracing will remain active until the DIVARTRACE is given a zero value (&SET DIVARTRACE 0).

Translation tables

When WebSphere Data Interchange translates data from your data format to an EDI transaction or from an EDI transaction to your data format, it can also substitute one value for another. WebSphere Data Interchange substitutes values through translation tables. Use translation tables to handle:

- Differences between your data and your trading partners' data. For example, say your trading partner uses its own numbers for parts you sell. You can set up a translation table to convert the part numbers your trading partner uses to those you use. An example of such a translation table is illustrated in Table 33.

Table 33. Translation table, differences in data

Local Value	Trading Partner Value
GLF8088	FR0100
GLF8588	FR0600
GLF8788	FR0800

- Conflicts between application data and EDI standards. For example, your application uses a code for a unit of measure that does not occur in the EDI standard. You can create a translation table to substitute an EDI standard code for your code on the send side and your code for the EDI standard code on the receive side. An example of such a translation table is illustrated in Table 34.

Table 34. Translation table, conflicts with standards

Local Value	Trading Partner Value
Boxes	BX
Cases	CS
Doz	DZ
Each	EA

You associate translation tables with maps through the Special Handling button on the Mapping Data Element Editor, as described on 224. This section describes how to set up translation tables.

Following are detailed procedures for creating new Translation Tables. For information on viewing, copying, editing, renaming, deleting, and printing translation tables, see "Performing common file management tasks" on page 45. For information on exporting translation tables, see "Exporting" on page 68.

Creating the tables

WebSphere Data Interchange provides two types of translation tables:

- Forward translation tables
- Reverse translation tables

Forward translation tables

The most commonly used is the forward translation table. Set up a forward translation table when you want to translate values from your application into values required by your trading partner when sending data, or translate values from your trading partner data into values required by your application when receiving data. This type of translation table contains values with a one-to-one relationship or many application values to one EDI standard value. The local value side of the table definition must be unique, as illustrated in Table 35.

Table 35. Sample forward translation, table values

Application Value	Standard Value
01	AA
02	BB
03	CC
04	CC

This type of translation table may be used on either send or receive maps, when both application and EDI standard values are unique. It can be used in send maps when only the application values are unique.

Reverse translation tables

Set up a reverse translation table when you want to translate one or more values from your trading partner to a single value in your application. This type of translation contains values with a one-to-one relationship or many EDI standard values to one application value. The EDI standard value side of the table definition must be unique, as illustrated in Table 36.

Table 36. Sample reverse translation, table values

Application Value	Standard Value
01	AA
01	BB
01	CC
22	DD
22	EE

The procedures for creating Forward Translation tables and Reverse Translation tables are exactly the same.

Creating a new translation table

Create a new translation table when you need to substitute values supported by your application for values supported by the trading partner's. You may also need to create a translation table to substitute values supported by your application and EDI standard transactions.

1. In the Mapping List window, click either the Forward Translation tab or the Reverse Translation tab.

2. Click New on the tool bar.

   The General tab displays.

3. Type in a name for the translation table. This is a required field.

   You may add a more complete description in the Description field if you wish.

4. If you are creating a Forward Translation Table, select—from the Data Type drop-down list in the Local Variable group box—the type of data in your data format.

   If you are creating a Reverse Translation Table, select—from the Data Type drop-down list in the Standard or Trading Partner Variable group box—the type of data in the EDI standard or in your trading partner's format. This is a required field.

   • Select CH if the data is character data.

   • Select R if the data is numeric data.

5. If you are creating a Forward Translation Table, select—from the Max Length drop-down list in the Local Variable group box—the maximum length of the data in your data format's fields. The maximum supported length is 35 characters, numbered 001 through 035.

   If you are creating a Reverse Translation Table, select—from the Max Length drop-down list in the Standard or Trading Partner Variable group box—the maximum length of the data in the EDI standard's or your trading partner's fields. The maximum supported length is 35 characters, numbered 001 through 035. This is a required field.

6. If you are creating a Forward Translation Table, select—from the Data Type drop-down list in the Standard or Trading Partner Variable group box—the type of data in the EDI standard or in your trading partner's format.

   If you are creating a Reverse Translation Table, select—from the Data Type drop-down list in the Local Variable group box—the type of data in your data format. This is a required field.

   • Select CH if the data is character data.

   • Select R if the data is numeric data.

7. If you are creating a Forward Translation Table, select—from the Max Length drop-down list in the Standard or Trading Partner Variable group box—the maximum length of the data in the EDI standard or in your trading partner's fields. The maximum supported length is 63 characters, numbered 001 through 063.

   If you are creating a Reverse Translation Table, select—from the Max Length drop-down list in the Local Variable group box—the maximum length of the data in your data format's fields. The maximum supported length is 35 characters, numbered 001 through 035. This is a required field.

   Note: The combined lengths for the local variable and the EDI standard or trading partner variable cannot exceed 68 characters.

8. Type the translation table in the grid at the bottom of the tab.

    a. If you are creating a Forward Translation Table, type the value in your data format in the Local Value column, then press the Tab key.

        If you are creating a Reverse Translation Table, type the value in the EDI standard or the value your trading partner wants to receive in the Standards or Trading Partner column, then press the Tab key.

    b. If you are creating a Forward Translation Table, type the value in the EDI standard or the value your trading partner wants to receive in the Standards or Trading Partner column, then press the Tab key.

        If you are creating a Reverse Translation Table, type the value in your data format in the Local Value column, then press the Tab key.

        WebSphere Data Interchange Client inserts another row and the display shifts back to the Local Value column in Forward Translation Tables and the Standards or Trading Partner Value column in Reverse Translation Tables. For more information on how the grid editor works, see "Using editor window grids" on page 47.

9. When you have finished entering all values required in the translation table, click Save on the tool bar to save the translation table.

Specifying send and receive usages

When you have completed a map, you must associate it with a trading partner or trading partners. WebSphere Data Interchange calls those associations send usages or receive usages, or jointly usages. Send usages are also referred to as send map usages. Receive usages are also referred to as receive map usages.

Applying the minimal trading partners concept

The concept of minimal trading partners attempts to reduce the amount of time spent on administrative functions of EDI. The traditional WebSphere Data Interchange was based on the idea that each trading partner would be identified to the product through a trading partner profile and a send usage or receive usage. Thus, a WebSphere Data Interchange installation with tens of thousands of trading partners would require an equal number of profiles and usages, even though the options were identical. The WebSphere Data Interchange concept of generic usages reduces the administrative impact of this model, but does not completely meet all its needs. Some installations do not need a setup for a trading partner, relying on post-translator processes to validate EDI transactions. WebSphere Data Interchange uses a combination of techniques and terminology to accommodate this minimal administrative model.

Applying the minimal trading partners concept, usages allow you to specify the same transaction mapping for several trading partners and provide specific overrides for each trading partner. This gives you the capability to use a single map for several different trading partners. Each send or receive usage instructs WebSphere Data Interchange Client which components of the map should be used and which should be overridden for that particular trading partner. See "Specifying usages and rules" on page 140 for more information.

The following procedures can be carried out from the trading partner windows as well as the mapping windows.

Viewing usages

1. In the Mapping List window, click the map for which you want to view usages, or go to that map's editor window.

2. Click View Usages on the tool bar.

   WebSphere Data Interchange Client runs a query that displays the Usages List window, which contains one tab. The Send Map Usages tab displays if the map is a send map. It displays a list of Send Usages associated with the map. The Receive Map Usages tab displays if the map is a receive map. It displays a list of Receive Usages associated with the map.

Creating a send or receive usage

Create a new send or receive usage after you create a new send or receive map and need to associate an existing trading partner with the map. You may also create usages to associate trading partners with existing maps.

1. In the Mapping List window, click the map for which you want to create usages, or go to that map's editor window.

2. Click View Usages on the tool bar.

   WebSphere Data Interchange Client runs a query that displays a list of send or receive usages associated with that map. Either the Send Map Usages tab or the Receive Map Usages tab displays listing the results of the query. If you created usages for this map previously, the existing usages display in the list window.

3. Click New on the tool bar.

   The Send Map Usage Editor or the Receive Map Usage Editor displays with the General tab in front.

Send and receive usages

Figure 13. Send Map Usage Editor

4. For both send and receive usages, you must identify the sending and receiving trading partners.

   For send maps, the sending trading partner identifies the internal trading partner who is the sender of the document. The receiving trading partner identifies the external trading partner who is the receiver of the document. Only trading partner profiles defined with a trading partner type of application trading partner or both will be included in the sending trading partner TP nickname drop-down list. Trading partner profiles are defined with a trading partner type of EDI trading partner or both.

   For receive maps, the sending trading partner identifies the external trading partner who is the sender of the document. The receiving trading partner identifies the internal trading partner who is the receiver of the document. Only trading partner profiles defined with a trading partner type of EDI trading partner or both will be included in the sending trading partner TP nickname drop-down list. Trading partner profiles are defined with a trading partner type of application trading partner or both.

For information on defining generic send and receive usages, see "Defining generic send usages" on page 246 and "Defining generic receive usages" on page 247.

Notes:

a. In send usages, the primary key is Map ID + Sending Trading Partner + Receiving Trading Partner + Internal Trading Partner ID. It is not possible to have two usages on the same map where all the primary key fields are the same. For example, it is not possible to have two usages on the same map where the only difference is that one usage is a test usage while the other one is production. If you need to send both test and production data to the same trading partner using the same map, then you can create a production usage, and in the Application Defaults profile set the check box Production Usage - Test Message. If that will not work in your environment, and if both the test and production usage must be active, then you must use a different internal trading partner ID for the test and production usages. If it is not necessary that they both be active at the same time, then you could copy the map and associate the test usage with one map, and the production usage with the other.

b. In receive usages, the primary key is Map ID + Sending Trading Partner + Receiving Trading Partner + Application Sender ID + Application Receiver ID. It is not possible to have two usages on the same map where all the primary key fields are the same. For example, it is not possible to have two usages on the same map where the only difference is that one usage is a test usage while the other one is production. If you need to send both test and production data to the same trading partner using the same map, then you can create a production usage, and in the Application Defaults profile set the check box Production Usage - Test Message. If that will not work in your environment, and it is not necessary that they both be active at the same time, then you could copy the map and associate the test usage with one map, and the production usage with the other.

You must create a trading partner profile before a trading partner's nickname displays in the list. For information on creating trading partner profiles, see Chapter 21, "Trading partners" on page 129.

5. Type values for any desired optional fields in the proper fields.

   Click

   ?

   for optional field names and descriptions.

6. When you have finished entering all the values you require in the send or receive usage, click Save on the tool bar to save the send or receive usage.

Editing send and receive usages

Edit a send or receive usage when you need to change translation specifications.

1. In the Mapping List window, click the map for which you want to edit a send or receive usage.

2. Click View Usages on the tool bar.

   Either the Send Map Usages tab or the Receive Map Usages tab displays. If you have created usages for this map, the existing usages display in the list window.

3. Double-click the send or receive usage you need to edit.

   The general tab displays.

4. Add, change, or delete entries as required.
5. When you have finished entering all values required in the send or receive usage, click Save on the tool bar to save the send or receive usage.

Copying send or receive usages

The copy function allows you to duplicate a send or receive usage within the WebSphere Data Interchange system in which you are working. If you want to base a new send or receive usage on an existing one, for example, copy the existing trading send or receive under a new name and edit it to the new specifications. You can also copy a usage to a different map or to a new trading partner.

1. In the Mapping List window, select the map that is associated with the send or receive usage you wish to copy.
2. Click View Usages on the tool bar.

   Either the Send Map Usages tab or the Receive Map Usages tab displays. If you have created usages for this map, the existing usages display in the list window.
3. Select the send or receive usage you wish to copy.
4. Select Copy from the Actions menu.

   The Copy Send Usage or Copy Receive Usage dialog box displays.
5. Select or enter new values into the field provided on the dialog box.
6. Click OK.

   WebSphere Data Interchange Client copies the send or receive usage using the information specified on the copy dialog box.

WebSphere Data Interchange Client copies the send or receive usage using the information specified on the copy dialog box. For information on deleting send or receive usages, see "Performing common file management tasks" on page 45.

Defining generic send usages

Where multiple trading partners can use the same send usage definition and map, a generic send usage can be defined to WebSphere Data Interchange. When combined with a generic routing code supplied by the application, it provides the capability to define one or more generic usages, each of which can handle multiple trading partners.

The generic routing code is an optional, three-character code provided by the application to select the correct generic usage when no specific usage has been defined for the trading partner. The Generic Routing Code can be provided in one of three ways:

1. The Translator Control Block (TRCB) includes a three-character field for the generic routing code. The API calls can provide the routing code in the TRCB.
2. For C and D processing, the C record includes a field for the three-character generic routing code.
3. For raw data processing, an application field that contains the generic routing code can be specified when defining the data format. The application field should be from one to three characters. If less than three characters, the value is filled with blanks. If greater than three characters, it is truncated to three.

Defining send usages

When defining send usages, an Internal Trading Partner ID that begins with an ampersand (&) indicates it is a generic usage. For a specific usage by routing code,

specify an ampersand (&) followed by the three-character generic routing code that the application provides to select this usage.

If the application does not want to provide a routing code or does provide a routing code but wants to select a default generic usage, the Internal Trading Partner ID can be defined as a single (&) followed by blanks. This type of generic definition is the default definition and is selected when no routing code is provided, or when the routing code is provided but no specific usage is found, or when the routing code contains blanks.

The purpose of the generic routing code is to allow a file containing one data format to generate more than one type of transaction (such as purchase orders and purchase-order changes). The generic routing code is required when the user wants to process transactions using different maps that reference the same data format.

For example, if the user wants to process purchase orders and purchase-order change documents that are defined using the same data format, the application could provide the following: a routing code of POR for a purchase order and WebSphere Data Interchange would select the usage with the Internal Trading Partner ID of &POR, which would reference a purchase order map; and a routing code of POC for a purchase-order change and WebSphere Data Interchange would select usage &POC, which would reference a purchase-order change map. An additional benefit when using this type of definition is that multiple documents can be included in the same raw data file if they all use the same data format.

Defining generic receive usages

When transactions received from multiple trading partners can use the same mapping, a generic receive usage can be defined to WebSphere Data Interchange to handle multiple trading partners with a single usage and map.

When defining the generic receive usages, a trading partner nickname with only an ampersand (&) is a special form indicating that it is a generic usage. The generic usage is selected when the normal selection process using the trading partner nickname does not find a receive usage.

Many generic receive usages can be defined as long as one of the listed match criteria is different (such as Application Sender, Application Receiver, Agency, Version, or Release, production/test).

Control strings

Compiling control strings

After you complete a map, you must compile a control string from the map before WebSphere Data Interchange can use the map. WebSphere Data Interchange Client uses the data you created during the mapping process as input to a program that compiles the map to create a control string. The WebSphere Data Interchange translator uses the control string in its translation processing.

While compiling, WebSphere Data Interchange Client checks for errors in the map you created. Error messages are displayed in the Execution Status window. Serious errors are also logged to the Message log.

Compiling a control string is the last thing you do after adding or updating a map. Any time you change a map, you must compile a new control string.

To compile a control string:

1. In the Mapping List window, click the map for which you want to compile a control string.
2. Click Compile Control String.

   An Execution Status window displays. Any errors are noted in the window.

   Note: If you are using WebSphere Data Interchange Client in client-server mode, skip Step three and four, as the client communicates directly with the WebSphere Data Interchange Host database. For more information on exporting and importing, see Chapter 6, "Export and Import" on page 67.

3. Export the control string from WebSphere Data Interchange Client.
4. Import the control string into WebSphere Data Interchange Host.

Viewing compiled control strings

Compiled control strings display in the Control Strings List window. Click Mapping on the Navigator bar to view the Control Strings List window. Table 37 describes the fields that display in the Control Strings List window.

Table 37. Control String List window field descriptions

This field. . .	Displays:
Map Name	The name of the map that this control string compiled.
Gen Date	The date the control string was complied.
Gen Time	The time the control string was compiled.

Mapping hierarchical loops

A hierarchical loop is similar to an organization chart. Just as an organization chart shows you the various groups of people and their relationships to the whole, a hierarchical loop shows you each group of data and its relationship to the whole.

Hierarchical loops define different levels of data, which can be used in any sequence and skipped when appropriate. This allows you to place the loop anywhere in your data.

Note: WebSphere Data Interchange includes support for Hierarchical Loops. For detailed explanations of HL loops and how WebSphere Data Interchange handles them, refer to Appendix D, "Hierarchical loops" on page 377

Mapping the HL Segment

WebSphere Data Interchange provides special handling for Hierarchical Loops. This section shows how to map the HL segment using the WebSphere Data Interchange Client interface.

1. Double-click an HL Loop.

   The Hierarchical Loop Support dialog box displays.
2. Click Special HL Support.

   The Qualify a Hierarchical Loop dialog box displays.
3. Type the ID of the node you are mapping in the Node Number field.

   Click

   for field descriptions.

4. Select a hierarchical level code from the Hierarchical Level Code (HL03) drop-down list.

   Click ? for field descriptions.

5. If you need to repeat the segment mapping, click Repeat.

   The Hierarchical Loop Support dialog box redisplays.

6. When you have completed all repeat mappings, click OK.

   The words Qualified by HL Logic. . . display next to the segment name in the Mapping Editor.

Creating fixed-to-fixed maps

Fixed-to-fixed translation is a method of translating application data from one format to application data of another format. This requires a send map to direct the movement of data between data formats.

Note: It is recommended that a data transformation map be used to create a map from a data format to a data format instead of using a send map. Using a data transformation map allows you to specify the original data format definitions in the source and target documents, avoiding the need to convert a data format to an EDI standard.

To create a fixed-to-fixed map

1. The target data format must be converted to an EDI standard. Begin by selecting the target data format from the Data Format List window.

2. Select Create Standard from Data Format from the Action menu to convert the target data format into an EDI standard.

   Note: The name of the generated EDI standard is taken from the Application File/Queue name field on the General tab of the Data Format Editor. This conversion only needs to be done once when the target data format is defined. If the target data format changes, the Create Standard from data format can be repeated to delete the old EDI standard and create a new one.

   An EDI standard is created. You can view this using the EDI standards related list windows and editors.

3. Create a send map using the source data format and the target EDI standard just created. Refer to "Creating a send or receive map" on page 220.

4. Create send usages as needed. Refer to "Specifying usages and rules" on page 140.

5. Compile the control string for the map. Refer to "Compiling control strings" on page 247.

Migrating a map to a new standard

Although migration is usually from one version of an EDI standard transaction to a later version of the same EDI standard transaction, sometimes it is necessary to migrate a map from one EDI standard to another.

Migrating a map to a new standard

Client migration option

1. Ensure that the new version of the EDI standard is loaded.
2. Migrate the map to the new EDI standard. On the General Tab of the Map Editor, select the new EDI standard and transaction using either the Source Document Definition or Target Document Definition fields, depending on whether this is a send or receive map. For more information on using the Map Editor, see "Using the map editor" on page 219.

Chapter 28. Validation mapping

WebSphere Data Interchange V3.2 supports Validation maps. A validation map is used to handle extended validation requirements, and to call an extended error function to report the information needed to create a functional acknowledgment. A validation map can also copy to local variables, and use most other mapping functions. A Validation map in WebSphere Data Interchange cannot produce an output.

This chapter contains an introduction to the Validation map editor, and how to create Validation maps. For detailed information on mapping techniques see Chapter 26, "Data transformation mapping" on page 201, and Appendix B, "Data Transformation mapping commands and functions" on page 301.

The Validation map editor

WebSphere Data Interchange Client's Validation map editor features a visual means for defining your extended validation criteria.

The Validation map editor displays a mapping commands window pane on its left side. The mapping commands window pane displays the layout of the document to be validated with mapping commands and comments inserted into it. These commands are used to check the source document for various user-defined conditions, and set appropriate errors and/or functional acknowledgement information if the conditions are not met. There is also a variables window pane in the Validation map editor. It is divided into three sub-panes that list all special variables, global variables, and local variables.

Starting the Validation map editor

The Validation map editor displays when you select a map from the Mapping List window, as follows.

1. Click Mapping on the WebSphere Data Interchange Client tool bar.

   The Mapping Functional Area displays.

   This window is used to access the list windows for each component type that occurs in the Mapping Functional Area. Select the Validation Maps tab to view the Validation Map List window. The list window shows all of the Validation Maps selected by the current query. Each row in a list window displays information about a different Validation Map.

   To display additional columns, click the scroll bar on the bottom of the screen to scroll to the right or left. To alter the columns that display on the screen, or to change which query is executed to produce the list, click the Modify Window Properties icon on the WebSphere Data Interchange Client Navigator bar. To create new queries, refer to "Creating a query" on page 268.

2. To view a map or to add or change its information, double-click the row of the map you want to work with.

   The Validation map editor window displays, with the Details tab in front. You add information or make changes to maps through its tabs and related dialog boxes, as described in the following sections.

Using the Validation Map editor

The Validation map editor window contains three tabs: General, Details, and Comments. Use the:

The Validation map editor

- General tab to enter and change map properties.
- Details tab to create and maintain the mapping commands.
- Comments tab to type any comments you wish about the selected map.

Following are detailed procedures for creating and editing Validation maps. For information on viewing, copying, editing, renaming, deleting, and printing maps, see "Performing common file management tasks" on page 45. For information on exporting maps, see "Exporting" on page 68.

Creating a Validation map

Create a new map when you want to handle extended validation requirements, or you want to call an extended error function to report the information needed to create a functional acknowledgment.

A validation map usually includes a number of **If** commands to check whether the document meets various user-defined conditions. If the conditions are not met, then the **Error** or **FAError** command is used to write a message to the log and set an error code for the transaction. The **FAError** command will set functional acknowledgement information, in addition to writing the log message and setting the error code. The following procedure shows how to create a validation map.

1. In the Mapping List window, click the Validation Maps tab so that any existing Validation maps are displayed.
2. Click the ″Create New Document″ icon on the tool bar.

   The Create a Validation Map — Map Name wizard page is displayed.
3. Type a map name in the Map Name field.

   You can use both letters and numbers to identify your map. Letters display in capitals. You cannot type spaces within the name. A Validation map must have a unique map name. Click Show Existing Map Names to view a list of map names already in use. A dialogue appears, showing existing map names, and types.

   You can enter a more detailed description of the map in the Description field.

   Click Next to continue.
4. The Source Dictionary wizard page displays. Select the EDI standard dictionary that contains the transaction that will be used as the source document definition in the validation map you are creating.

   Click Next to continue.
5. The Source EDI Standard Transaction wizard page displays. From the list that appears, select the EDI standard transaction you want to use as the source document definition for your map.

   Click Next to continue.
6. The Confirmation window displays. Confirm the selections displayed. If they are correct, click Finish to save the information.

   After finishing the map information, the Validation Map editor displays with the Details tab opened.

   a. Click the General tab.

   The General tab displays.

   The Validation map editor's General tab page contains general information about the map. It includes the map name, a brief description of the map and it identifies the source document definition.

The name of the map is set when the map is created. It displays on the General tab page for informational purposes only. You can change the name of the map by using the Rename function.

Use the Description field to provide a brief description of the map. The description can be up to 50 characters long.

General editing procedures

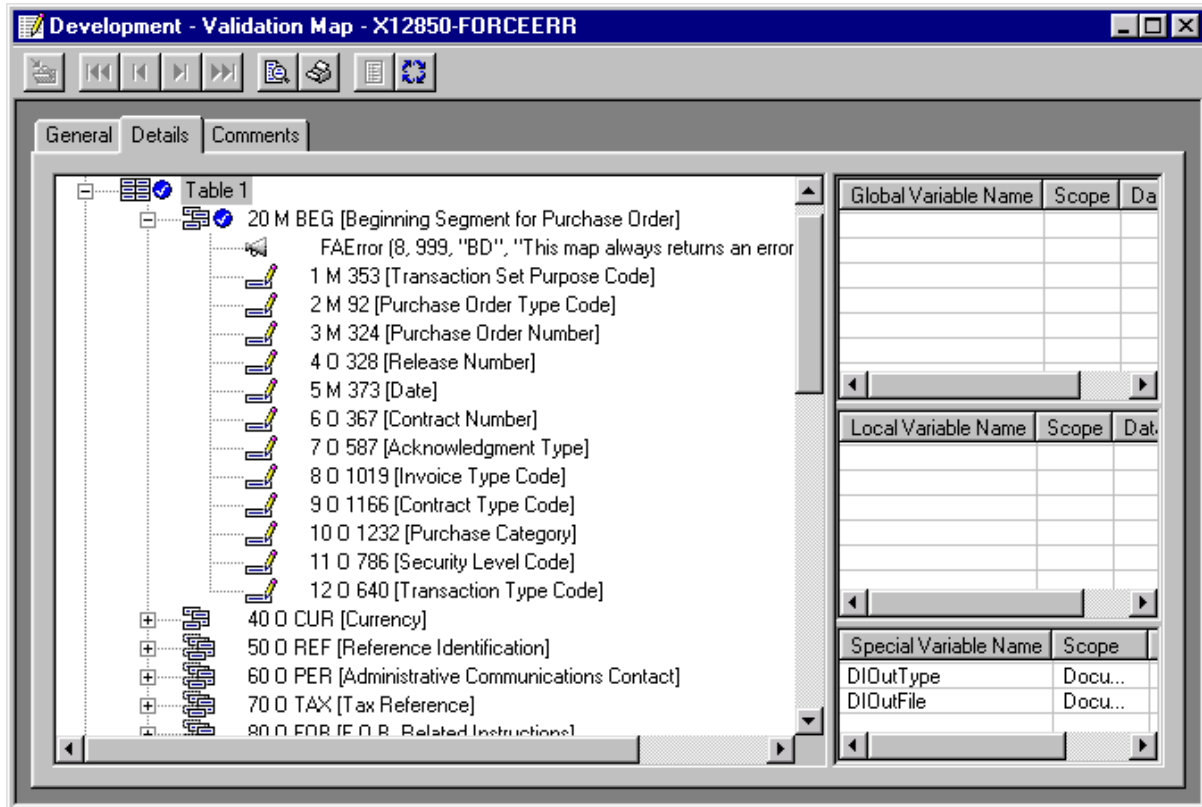The Details tab in the Validation map editor allows you to perform drag and drop mapping on your documents.

Figure 14. Validation map editor

The left pane in the window displays the details for the Validation Map, together with mapping commands and comments for the map.

The right right pane is the variables window pane, which includes the lists for Global, Local, and Special Variables defined for the map.

Editing a map

Edit a map when you need to add, delete, or change mapping commands within the map. You might also need to edit a map to meet specific requirements of a new trading partner.

1. In the Mapping List window, double-click the map you wish to edit.

   The map displays in the Validation map editor window with the Details tab in front.

   Changes you made earlier to the map or its associated items display on the screen.

General editing procedures

2. Create new map commands or comments for the map by right-clicking the map name in the left pane. You can insert a Command from the available list, a Command Group, a Comment, or a Comment Group. A navigation tree is created, containing the elements you have created.

   When you have created Map Commands or comments, you can move them by clicking and dragging the elements to another part of the tree.

   To insert a new command or comment into the tree, right-click the existing comand or comment you want to insert your new command or comment before or after, and select "Insert Before" or "Insert After" from the pop-up list.

   To open an element, double-click the element, or right-click the element and select "Open...".

   To delete an element, right-click the element and select "Delete".

3. To create new variables, right-click in the relevant variable pane and select "New". Complete the fields in the window that appears, and click "Save". To edit an existing variable, double-click the line that contains the variable.

4. Change information as required in the General tab.

5. Click Save on the tool bar to save the map.

Using the Map Command window pane

The Map Command window pane allows you to apply WebSphere Data Interchange Client's advanced mapping capabilities.

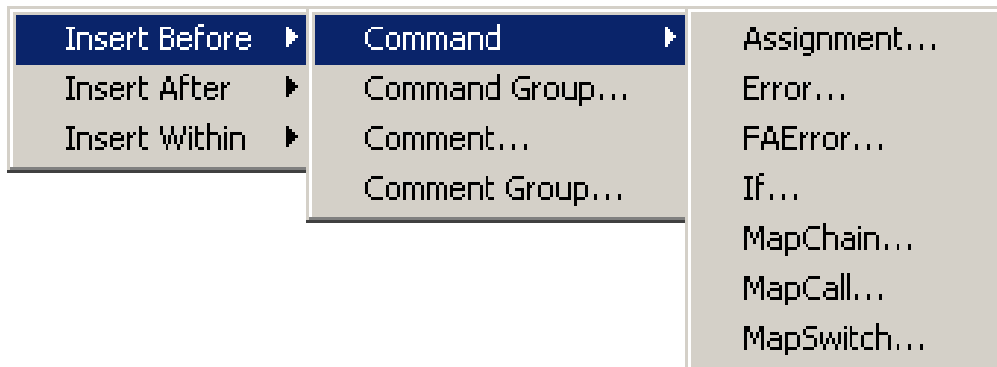1. Right-click an element in the Mapping Command window pane and follow the cascading menus.

Figure 15. Mapping command menus

2. If you want to insert a command, select where the command should be inserted, select Command, and then select the command to be inserted.

   The Mapping Command Editor displays with a prototype of the command.

3. Edit the prototype to create the command you need.

4. Click Repeat if you want to add the command, and then create another similar to it.

5. Click OK when you have your last command ready.

Using Validation Maps

You can work with and use Validation maps in much the same way as you can with Data Transformation Maps. Many of the concepts explained for Data Transformation maps in Chapter 26, "Data transformation mapping" on page 201 also apply to Validation Maps. You should note the following:

- Validation maps are source-based maps, in which the commands are based on the order of the source document. Commands that are only valid for target-based maps are not permitted in a validation map.
- You can use the FAError command in Validation Maps. For information on FAError, see Appendix B, "Data Transformation mapping commands and functions" on page 301.
- Validation maps do not have a target document. Commands and functions that specify a target element, such as **MapTo**, are not permitted in a validation map.

For more information on Validation Mapping techniques and concepts, see the following sections in Chapter 26, "Data transformation mapping" on page 201:

- "Specifying qualification" on page 207
- "Specifying Hierarchical Loop Levels" on page 212
- "Translation tables" on page 213
- "Specifying map rules" on page 215
- "Compiling control strings" on page 217
- "Defining Global Variables" on page 218

WebSphere Data Interchange techniques for using literal keywords and other advanced mapping techniques are documented in Appendix B, "Data Transformation mapping commands and functions" on page 301.

General editing procedures

Chapter 29. Functional Acknowledgement mapping

WebSphere Data Interchange V3.2 supports Functional Acknowledgement maps. A functional acknowledgement map is a data transformation map that allows you to create a functional acknowledgment to be returned to your trading partner. After the functional acknowledgment map has been created, it can be selected on the trading partner rules setup as the map to be used during functional acknowledgment generation. Both source and target based mapping are available for mapping functional acknowledgments.

This chapter contains an introduction to the Functional Acknowledgement map editor, and how to create Functional Acknowledgement maps. For detailed information on mapping techniques see Chapter 26, "Data transformation mapping" on page 201, and Appendix B, "Data Transformation mapping commands and functions" on page 301

Functional Acknowledgement maps provided with WebSphere Data Interchange

Several Functional Acknowledgment maps are provided with WebSphere Data Interchange Version 3 Release 2:

- $DT_FA997V2R4 - For X12 997 Version 2 Release 4 and lower
- $DT_FA997V3R5 - For X12 997 Version 3 Release 5 and lower
- $DT_FA997V3R7 - For X12 997 Version 3 Release 7 and lower
- $DT_FA999V3R3 - For UCS 999 Version 3 Release 3 and lower
- $DT_FACONTRL - For UN/EDIFACT earlier than Version 94B
- $DT_FACONTRL94B - For UN/EDIFACT version 94B(Version 2, Release 1) and later

The source document definition for Functional Acknowledgment maps always uses Dictionary $FUNC_ACK_METADATA_DICTIONARY and Document $FUNC_ACK_META. This is the data format definition for the functional acknowledgment records created during syntax validation of EDI data. The target document definition is a selection of EDI standard transaction definitions and are provided as follows:

- $DT99724 - For X12 997 Version 2 Release 4 and lower
- $DT99735 - For X12 997 Version 3 Release 5 and lower
- $DT99737 - For X12 997 Version 3 Release 7 and lower
- $DT99933 - For UCS 999 Version 3 Release 3 and lower
- $DTCTL - For UN/EDIFACT earlier than Version 94B
- $DTCTL21 - For UN/EDIFACT version 94B (Version 2, Release 1) and later

The Functional Acknowledgement map editor

WebSphere Data Interchange Client's Functional Acknowledgement map editor features a visual means for associating elements from a source document format with elements in a target document format. The Functional Acknowledgement map editor uses a split screen that permits you to create map associations.

The source document definition appears in the upper left corner of the Functional Acknowledgement map editor and displays the layout of the document that will be used as the source of the translation. The target document definition displays the

The Functional Acknowledgement map editor

layout of the document that will be created by the translation. It is displayed in the upper right hand corner of the Functional Acknowledgement map editor. The Functional Acknowledgement map editor displays a mapping commands window pane in its lower left hand corner. The mapping commands window pane displays the source document layout with mapping commands and comments inserted into it. There is also a variables window pane in the Functional Acknowledgement map editor. It is divided into three sub-panes that list all special variables, global variables, and local variables.

Starting the Functional Acknowledgement map editor

The Functional Acknowledgement map editor displays when you select a map from the Mapping List window, as follows.

1. Click Mapping on the WebSphere Data Interchange Client Navigator bar.

   The Mapping Functional area displays, showing lists of existing maps.

   This window is used to access the list windows for each component type that occurs in the Mapping Functional Area.

2. Select the Functional Acknowledgement Maps tab to view the Functional Acknowledgement Map List window. The list window shows all of the Functional Acknowledgement Maps selected by the current query. Each row in a list window displays information about a different Functional Acknowledgement Map.

   To display additional columns, click the scroll bar on the bottom of the screen to scroll to the right or left. To alter the columns that display on the screen, or to change which query is executed to produce the list, click the Modify Window Properties icon on the WebSphere Data Interchange Client Navigator bar. To create new queries, refer to "Creating a query" on page 268

3. To view a map or to add or change its information, double-click the row of the map you want to work with.

   The Functional Acknowledgement map editor window displays, with the Details tab in front. You add information or make changes to maps through its tabs and related dialog boxes, as described in the following sections.

Using the Functional Acknowledgement map editor

The Functional Acknowledgement map editor allows you to show how data is to be translated from the source document to the target document. The editor works by displaying the source document definition on one side of the screen and target document definition on the other. The Functional Acknowledgement map editor allows you to associate components of your source document with components of the target document by dragging source components and dropping them into the correct locations in the target document definition. Components include simple elements, such as fields in a data format or data elements in an EDI standard transaction, and compound elements, such as loops and record in a data format and segments in an EDI standard transaction.

You can map the simple elements in the source document definition to the simple elements in the target document definition in any of these patterns:

- One simple element to one simple element
- Several simple elements to one simple element
- One simple element to several simple elements

The Functional Acknowledgement map editor window contains three tabs: General, Details, and Comments. Use the:

- General tab to enter and change map properties.
- Details tab to create and maintain the mapping commands.
- Comments tab to type any comments you wish about the selected map.

Following are detailed procedures for creating and editing maps. For information on viewing, copying, editing, renaming, deleting, and printing maps, see "Performing common file management tasks" on page 45. For information on exporting maps, see "Exporting" on page 68.

Creating a functional acknowledgment map

Create a new map when you want a functional acknowledgement to be returned to your trading partner but do not want to use one of the functional acknowledgement maps that was shipped with WebSphere Data Interchange.

Note: To create a functional acknowledgment map, you need to have a functional acknowledgment definition. Functional acknowledgment document definitions are:

- 997 and 999 in the X12 EDI Standard
- CONTRL transaction in the EDIFACT EDI standard

1. In the Mapping List window, click the Functional Acknowledgement Maps tab so that any existing Functional Acknowledgement Maps are displayed.
2. Click the ″Create New Document″ icon on the tool bar.

   The Create a Functional Acknowledgement Map — Map Name wizard page is displayed.
3. Type a map name in the Map Name field.

   You can use both letters and numbers to identify your map. Letters display in capitals. You cannot type spaces within the name. A Functional Acknowledgement Map must have a unique map name. Click Show Existing Map Names to view a list of map names already in use. A dialogue appears, showing existing map names, and types.

   You can enter a more detailed description of the map in the Description field.

   Click Next to continue.
4. The ″Source or Target″ wizard page displays. Use this page to define whether your map will be source based, or target based.

   Select Source or Target by selecting the appropriate radio button.

   Click Next to continue.
5. The ″Select Target″ wizard page displays. Available target document definitions are displayed in the page's list box.

   Choose the functional acknowledgement to be used as the target document definition in your map.

   Click Next to continue.
6. The Confirmation window displays. Confirm the selections displayed. If they are correct, click Finish to save the information.

   After finishing the map information, the Functional Acknowledgement Map editor displays with the Details tab opened.

   a. Click the General tab.

      The General tab displays.

Creating a functional acknowledgement map

The Functional Acknowledgement map editor's General tab page contains general information about the map. It includes the map name, a brief description of the map and it identifies the source and target document definitions.

The name of the map is set when the map is created. It displays on the General tab page for informational purposes only. You can change the name of the map by using the Rename function.

Use the Description field to provide a brief description of the map. The description can be up to 50 characters long.

General editing procedures

The Details tab in the Functional Acknowledgement map editor allows you to perform drag and drop mapping on your documents.
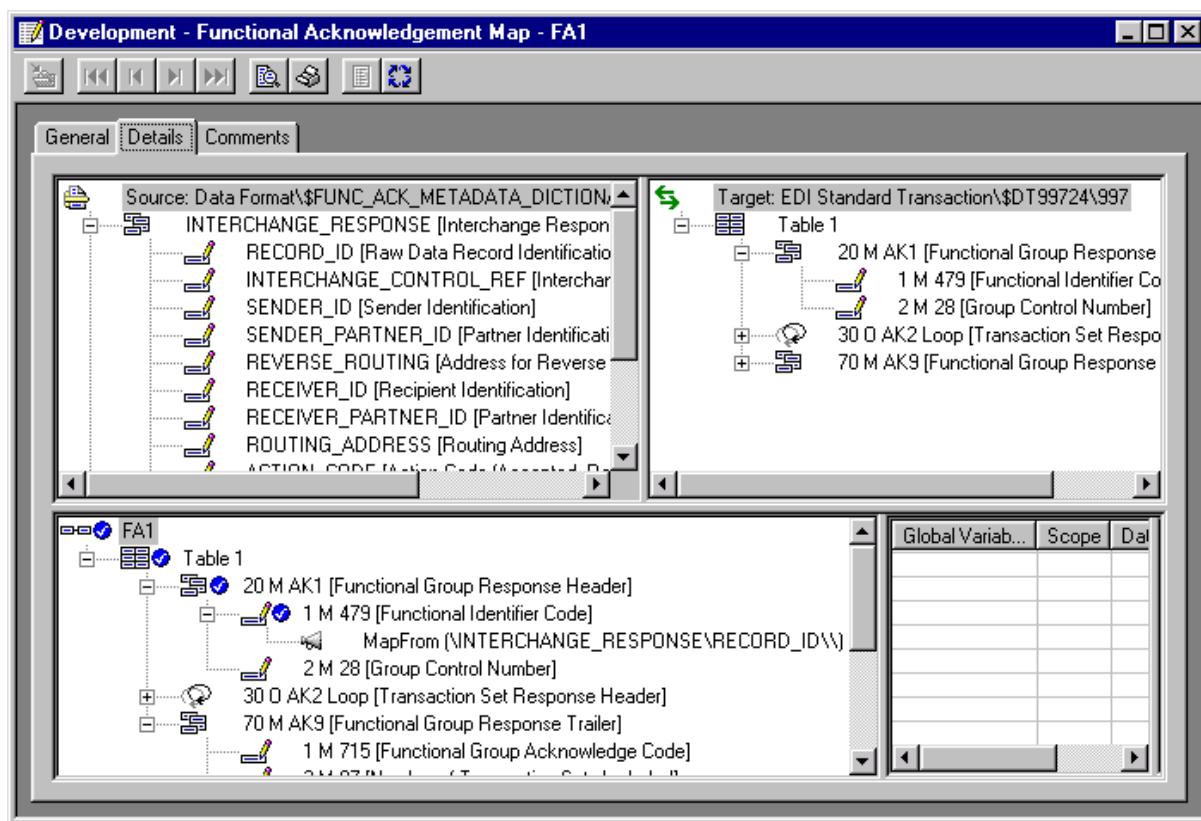

Figure 16. Functional Acknowledgement map editor

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.

For more information about the Map Command window pane, see "Using the Map Command window pane" on page 207.

For a list of mapping components and their associated graphics, see Table 24 on page 206.

1. Click the plus (+) sign next to a compound element, such as a loop, segment, or record in the target document definition. Simple elements, such as fields and data elements, will not have a plus (+) sign next to them.

   The compound element expands to show the compound and simple elements that comprise the compound element.

   If you wish to close any expanded compound element, click the minus (-) sign.

2. Click the element you wish to map on the top left side of the screen. 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 screen over the element with which you want to associate it, that component becomes highlighted. Release the mouse button.

   You can drag all simple elements and repeating compound elements. When you are dragging an element, holding it over a compound element will cause the compound element to expand after a few moments, if it is not already expanded. Dragging an element to any edge of the window pane will cause the window pane to automatically scroll up, down, left or right, depending on which edge you are near and whether the window pane can be scrolled in that direction.

   Note: If the element does not become highlighted when you move the cursor on top of it, that means it is not a valid place to perform a drop. For example, you cannot drop a data format record onto a data element in an EDI transaction.

   After you have mapped an element, a command is created in the mapping commands window pane and inserted into an appropriate position. A green check mark displays next to the mapped element in the mapping command window. A green check mark also displays next to its parent elements so that you know the parent element contains mapped components.

   Note: This procedure presents a simple mapping situation in which you associate elements in the source document definition with elements in the target document definition. For anything but the most basic mapping associations, you must use WebSphere Data Interchange's advanced mapping capabilities. For more information, see "Using the Map Command window pane" on page 207.

3. Continue dragging and dropping elements in the source document definition onto elements in the target document definition until you have mapped all of the information required for this map.

4. Enter any general comments on the map in the Comments tab.

5. When you have completed mapping, click Save on the tool bar to save the map.

Editing a map

Edit a map when you have modified the associated target or source document definition, or when you need to add, delete, or change mapping commands within the map. You may also need to edit a map to meet specific requirements of a new trading partner.

1. In the Mapping List window, double-click the map you wish to edit.

   The map displays in the Functional Acknowledgement map editor window with the Details tab in front.

General editing procedures

Changes you made earlier to the map or its associated items display on the screen, as well as changes you made to the source or target document definition.

2. Create new associations between the source and target elements, or change existing ones.

   • To make an association between a new element in the source document definition and an element in the target document definition, drag the source element and drop it on the proper target element.

   • To edit an existing association, double-click the mapping command in the Mapping Command window pane. The Mapping Command Editor displays.

   • To delete an association, select the mapping command you want to delete and press the Delete key.

3. Change information as required in the General tab.

4. Click Save on the tool bar to save the map.

** Simple Elements - all others are Compound Elements

Using the Map Command window pane

The Map Command window pane allows you to apply WebSphere Data Interchange Client's advanced mapping capabilities.

1. Right-click an element in the Map Command window pane and follow the cascading menus.
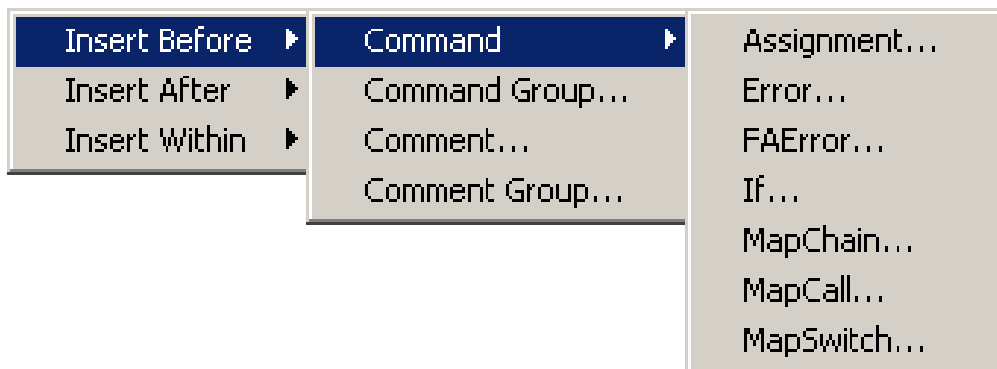

Figure 17. Cascading menus in the Map Command window pane

2. If you want to insert a command, select where the command should be inserted, select Command, and then select the command to be inserted.

   The Mapping Command Editor displays with a prototype of the command.

Figure 18. Mapping Command Editor

3. Edit the prototype to create the command you need.

4. Click Repeat if you want to add the command, and then create another similar to it.

5. Click OK when you have your last command ready.

Source document definition record layout

The Source document definition record layout is:

```
Interchange response Record - ID = INTHREC
Functional Group Response Loop:
Functional Group Response Record - ID = GRPHREC
Message  Response Loop:
Message response Record - ID = MSGHREC
Segment Data Loop:
Data Segment Note Record - ID = SEGREC
Element Data Loop:
Data Element Note  Record - ID = ELEREC
Message Response Trailer Record - ID MSGTREC
Functional Group Response Trailer Record - ID = GRPTREC

Source document definition record definitions $FUNC_ACK_META

Interchange response Record - ID = INTHREC
Interchange control reference
Sender Identification
Partner Identification code
Address for reverse routing
Recipient Identification
Partner Identification code
Routing address
Action, coded
Syntax error, coded
Segment tag, coded
Erroneous data element position
Erroneous component data element

Functional Group Response Record - ID = GRPHREC
Functional Identifier Code
Group Control Number
Application sender's identification
Partner Identification code
Application recipient's identification
Partner Identification code
Action, coded
Syntax error, coded
Segment tag, coded
Erroneous data element position
Erroneous component data element

Message response Record - ID = MSGHREC
Message reference number
Message type identifier
Common Access Reference
Message type version number
Message type release number
Controlling agency
Association assigned code
Action, coded
Syntax error, coded
Segment tag, coded
Erroneous data element position
Erroneous component data element

Data Segment Note Record - ID = SEGREC
Segment ID Code
Segment Position in Transaction Set
```

General editing procedures

```
            Loop Identifier Code
            Segment Syntax Error Code

            Data Element Note  Record - ID = ELEREC
            Element Position in Segment
            Component Data Element Position in Composite
            Data Element Reference Number
            Data Element Syntax Error Code
            Copy of Bad Data Element

            Message Response Trailer Record - ID MSGTREC
            Message Acknowledgment Code
            Message Syntax Error Code
            Message Syntax Error Code
            Message Syntax Error Code
            Message Syntax Error Code
            Message Syntax Error Code

            Functional Group Response Trailer Record - ID = GRPTREC
            Functional Group Acknowledge Code
            Number of Messages Included
            Number of Received Messages
            Number of Accepted Messages
            Functional Group Syntax Error Code
            Functional Group Syntax Error Code
            Functional Group Syntax Error Code
            Functional Group Syntax Error Code
            Functional Group Syntax Error Code
```

Using Functional Acknowledegment Maps

You can work with and use Functional Acknowledgement maps in much the same way as you can with Data Transformation Maps. Many of the concepts explained for Data Transformation maps in Chapter 26, "Data transformation mapping" on page 201 also apply to Functional Acknowledgement maps. You should note the following:

- Functional Acknowledgment maps always use the same data format for the source document. The dictionary name is `$FUNC_ACK_METADATA_DICTIONARY`, and the data format name is `$FUNC_ACK_META`.

For more information on Functional Acknowledgement Mapping techniques and concepts, see the following sections in Chapter 26, "Data transformation mapping" on page 201:

- "Specifying qualification" on page 207
- "Translation tables" on page 213
- "Specifying map rules" on page 215
- "Compiling control strings" on page 217
- "Defining Global Variables" on page 218

WebSphere Data Interchange techniques for using literal keywords and other advanced mapping techniques are documented in Appendix B, "Data Transformation mapping commands and functions" on page 301.

Part 5. Administration

Chapter 30. Queries

WebSphere Data Interchange Client's Query function gives you complete control over the information that displays in any WebSphere Data Interchange Client list window. Queries control both the fields that display in the list window and which items display in the list window.

Queries are particularly useful after you have been using WebSphere Data Interchange Client for a while and the number of items in your list windows is large. You can write queries that filter items to display only the set you wish to view. This ability will be particularly useful in working with the Transaction Store in order to monitor the status of your activity.

WebSphere Data Interchange Client comes with preset queries for each of the list windows you can access in the application. You can use these queries as a template to create and modify your own.

For information on printing reports of query results, see Chapter 31, "Reports" on page 273.

About queries

A query is, in simplest terms, a request for specific information from the database. A query allows you to determine what information you want to see and the order in which you want to see it.

Purpose

Queries have three main uses.
1. They provide general information about what is currently stored in the database
2. They let you monitor various aspects of your translation
3. They improve system performance by limiting the items that display in WebSphere Data Interchange Client windows

You run a query each time you open any of the list windows in WebSphere Data Interchange Client. You see the results of that query displayed in the list portion of the window. For example, when you open the Trading Partners List window, you generally run the Trading Partners "All" query, which finds all Trading Partner profiles and displays the resulting list with many trading partner fields shown in the window. This list contains all Trading Partner profiles and sorts them by their Trading Partner Nickname.

When you open a list window, the last query that was run is executed. To modify the query or to select an alternate query, use the Properties button, as described in "Modifying list window information" on page 36.

The second purpose of queries is to allow you to monitor the status of your translation activity. This will likely be most useful in the Transaction Store. For example, you can track translation activity with a specific trading partner, or any other type of information you would like to track.

Setup overview

You work with queries starting in the Query List window, which you access by selecting the Open Query List command from the File menu.

The Functional Area drop-down list at the top left of the window allows you to select a functional area within WebSphere Data Interchange Client. The Query Type list box allows you to select the type of item you wish to work with within the functional area.

For example, selecting the Trading Partner functional area displays queries available for the Trading Partner List window. You can then choose to work with queries for either Trading Partner profiles or Contact profiles, which are the two profiles that are available in the Trading Partner functional area.

The Query List box in the middle of the window displays all queries that have been defined for a specific item within a functional area. The buttons on the right side of the window allow you to create a new query; edit, delete, copy or run an existing query; and, finally, close the window when you are finished with it.

Note: Queries provided with WebSphere Data Interchange Client are read-only. Therefore, you cannot edit or delete them. You can, however, copy them and then make changes to or subsequently delete the copy. Understand that in copying a preset query, you are actually creating an identical query with a new name.

To create an identical query, edit an existing query, or copy an existing query, you work in the Query Editor window. By clicking the appropriate tab, you can select the columns, or fields, that will display in your query, decide how you want WebSphere Data Interchange Client to sort information in the columns, and apply criteria that filter, or limit, the information that displays in query results.

Working with queries

When you have decided what information you want to receive from a query, you can create a new query, or copy or edit an existing query. WebSphere Data Interchange Client also allows you to run queries at any time so you can view specific information without having to go to a particular functional area. When you are finished with a query, you can also delete it, except for the preset queri-332.9(beena8d0)-332.9(y

5. In the Columns tab, use the Query Name field to type in the query name. Then use the Available Columns and Selected Columns list boxes, and the arrow selection buttons to determine which columns will be included in the query results.

   The > button moves the highlighted item from the Available Columns to the Selected Columns list box.

   The < button moves the highlighted item from the Selected Columns to the Available Columns list box.

   The >> button moves all items from the Available Columns to the Selected Columns list box.

   The << button moves all items from the Selected Columns to the Available Columns list box.

6. If you want to sort information that displays in the list, click the Sorting tab.

   The Query Editor window displays changes. Sorting lets you specify the way in which that data will be organized in the results of your query.

   a. Use the Columns Available For Sorting list box to choose which columns the query results will be sorted on. The < and > buttons allow you to move column selections back and forth as desired.

   b. For each column, choose whether you want to sort Ascending (0 to 9, A to Z) or Descending (Z to A, 9 to 0).

7. If you want to narrow the list based on the value of a particular field, set up a filter, as described below.

8. When you are finished working with the various tabs in the Query Editor window, click OK.

   WebSphere Data Interchange Client returns to the Query List window.

9. At this point, you can begin again in another Functional Area or with a different query type. If you are finished, click Close to close the Query List window.

Using filters in queries

Filters allow you to narrow the list of items that display in a list window based on values in specific fields or columns. Only information that meets the filter specifications is displayed in the query results. If no filters are defined, then all information available for each selected column displays.

You can set up two types of filter queries: static and dynamic. Static filters allow you to set up the query to filter for the same value in a column every time you run the query. Dynamic filters prompt you to type in a value each time you run the query so that you can specify the particular value that meets your needs at that time.

To set a filter in a query:

1. Create a new query following the procedure in "Creating a query" on page 268 or edit an existing query following the procedure in "Editing a query" on page 270.

2. Click the Filters tab.

   The Query Editor window displays changes.

3. If you are creating a new filter condition, click New to set up filters for this query. If you are copying and modifying existing filter conditions, you can highlight an existing filter and then click Edit.

   The Filter Condition dialog box displays.

4. In the Filter Column list box, click the column you want to filter.

5. In the Relation Operator list box, click the desired operator for the selected column. The function of most of the operators are self-evident. Explanations of "Like," "Between," "Is Empty," and "Is Not Empty," however, display in Table 38.

6. The Value area allows you to specify whether you want this specific filter to always be the same, and what the value of the filter is, or whether you want to be prompted each time you run the query so that you can specify the particular value that meets your needs at that time. Note that if you click the Prompt option, the prompt field below is automatically completed with the selected column and relation operator. You can change the prompt text as desired. Changing the prompt text will not change the filter.

   Note: Many database management systems are case sensitive. Use the correct case when entering a value.

   If you always want the query to return the same Mailbox ID, click Fixed Value option and then type in the specific Mailbox ID you want the query to return.

   If, however, you may want to see different Mailbox IDs at different times, click the Prompt option. When you run the query, you will be prompted for a specific Mailbox ID.

   Note: Not all operators work in all columns. A Date column, for example, will not accept a "Like" operator. You will receive database errors if you run such a query.

7. When you are finished setting filter conditions, click OK.

   You return to the Filters tab of the Query Editor window.

8. When you have finished working with the query, click OK.

   WebSphere Data Interchange Client returns to the Query List window.

9. Click Close to close the Query List window.

Table 38. Selected Relation Operators

This operator. . .	Displays:
Like	Items similar to the string of characters typed in the field. You can also use the % wildcard with the Like operator. The % stands for any string of characters. If you typed in RPT%, for example, you would receive a list of all items that begin with RPT. **Note:** Results of queries using Like may vary depending on the client-server middleware your company uses. It is recommended that you always use the % operator at the end of strings when using Like.
Between	Items between the two values you enter in the Value list box. Whether the endpoints are included when you use the Between operator depends on the middleware you use.
Is Empty	All fields that contain Null as a value. Use this operator when you want to find an error, as null fields only exist when there are errors.
Is Not Empty	All fields besides those containing Null as a value. Use this operator when you want to show all fields, including those that include blanks.

Editing a query

WebSphere Data Interchange Client allows you to make changes to existing queries when the information you wish to view changes.

1. Choose the Open Query List command from the File menu.

   The Query List window displays.

2. Use the Function Type drop-down list to select a specific functional area within WebSphere Data Interchange Client.

3. In the Query Type list box, click the type of query you want to work with.

4. Make a selection in the Query List box, and then click Edit.

   The Query Editor window displays.

   At this point, you can follow Steps 5 on page 269 through 9 on page 269 of the procedure for creating a new query. See "Creating a query" on page 268.

5. When you are finished working with the various tabs in the Query Editor window, click OK.

   WebSphere Data Interchange Client returns to the Query List window.

   At this point, you can choose another query to edit. If you are finished, click Close to close the Query Editor window.

Copying a query

You may want to create a new query that is similar to an existing one. For example, one of your queries displays all purchase orders received on a given day and you need one to display all purchase order acknowledgments received. Rather than creating an entirely new query, copy the first query and make the appropriate changes.

1. Highlight the query you want to copy.

2. Click Copy on the Query List window.

   The Query Editor window displays.

3. Type a new name for the query in the Query Name field.

4. Modify the query in any way you want, then click OK.

   The Query List window redisplays, and the query displays in the Query List.

Deleting a query

If you have not used a particular query in some time and do not anticipate using it again, you may want to delete that query.

1. Highlight the query you want to delete.

2. Click Delete on the Query List window.

   You see a confirmation message.

3. If you are sure you want to delete the query, click Yes.

   WebSphere Data Interchange Client deletes the query.

Running a query

After you have created a query, WebSphere Data Interchange Client allows you to run it so you can view the information you have selected in a tabular format. You can run any existing query at any time.

1. Choose the Open Query List command from the File menu.

   The Query List window displays.

2. Use the Functional Area drop-down list to select a functional area within WebSphere Data Interchange Client.

3. In the Query Type list box, click the name of the item you want to work with.

4. Click the specific query you wish to execute and then click Run.

   WebSphere Data Interchange Client immediately executes the query and displays the results.

Queries: running

If the query filter calls for a prompt, the Specify Filter Values dialog box displays on top of what will be the results display.

Steps 5 and 6 apply only if a query filter has been specified in the query.

5. Press Tab to move to the Value field and enter a value appropriate to the Parameter Prompt.

   Note: Many database management systems are case sensitive. Use the correct case when entering your values.

6. Click OK to display the results of the query.

   The results of the query display in the list window.

Chapter 31. Reports

The WebSphere Data Interchange Client Report function lets you preview and print the results of a query. In essence, a report displays information for printing on paper or previewing on screen. Printing a report allows you to create a permanent copy of any information you choose.

Note: WebSphere Data Interchange Client includes a run-time version of Crystal Reports** Version 5 for using the layouts that come with WebSphere Data Interchange Client. To create your own report layouts, you must purchase the full Crystal Reports product.

For information on creating queries, see Chapter 30, "Queries" on page 267.

About reports

A report is a combination of a query and a report layout that is printed on paper rather than displayed in a WebSphere Data Interchange Client window.

The main purpose of a report is to create a paper document that can make it easier for you to share or retain certain key information stored in the database. You can create management reports for circulation, or permanent records for filing, as needed.

Setup overview

When you create a report, you associate a query you have already created with a report layout that has also already been created, and name it.

You begin all report procedures in the Report List window, which you access by selecting the Open Report List command from the File menu.

The Functional Area drop-down list at the top left of the window allows you to select a functional area within the WebSphere Data Interchange Client. The Report Type list box allows you to select the type of item within the functional area.

The Report List list box in the middle of the window displays all reports that have been defined for the selected list window. The buttons on the right side of the window allow you to print or preview an existing report, create a new report, edit or delete an existing report, and, finally, close the window when you finish working with it.

To create a new report or edit an existing report, you work in the Report Editor window. This window allows you to name reports, select the query to associate with a report as well as select a specific layout to use for the report.
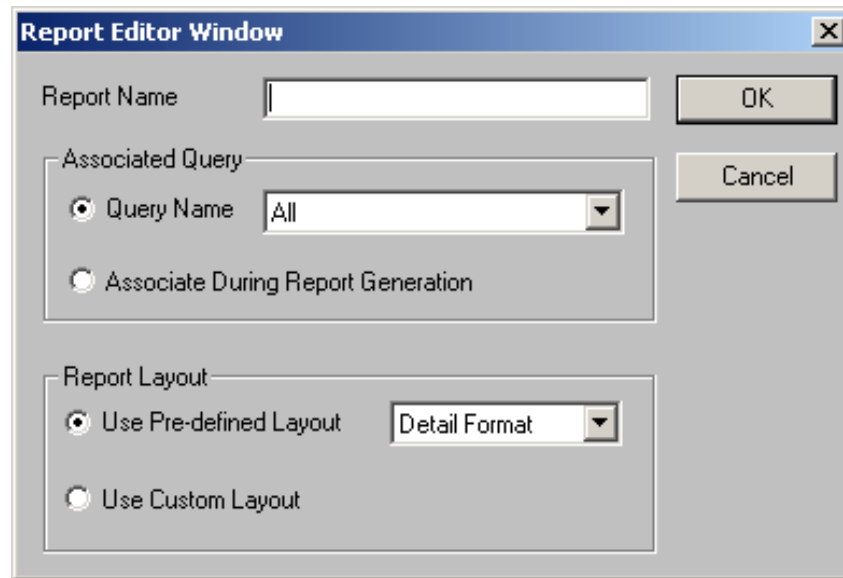
Working with reports

When you have decided what information you want to include in your report (and verified that there is a query to provide that information), you can create a report. You can also print or preview any report, edit it, and, when you are finished with a report, you can delete it.

Creating a report

Creating a report is a fairly straightforward process in which you name the report, pick a query to associate with the report and specify a layout for the report.

1. Choose the Open Report List command from the File menu.

   The Report List window displays.

2. Use the Functional Area drop-down list to select a functional area within WebSphere Data Interchange Client.

3. In the Report Type list box, click the specific list window you want to work with.

4. Click New.

   The Report Editor window displays.

5. Type in a name for the report.

6. In the Associated Query section, you have two choices: You can select the name of an existing query to permanently associate with this report, or you can choose to be prompted for the name of an existing query at the time you go to print the report.

7. In the Report Layout section, you also have two choices: You can choose one of the predefined report layouts that come with WebSphere Data Interchange Client, or you can choose a custom report layout you have created using Crystal Reports.

   Note: WebSphere Data Interchange Client includes a runtime version of Crystal Reports for using the layouts that come with WebSphere Data Interchange Client. To create your own report layouts, you must purchase the full Crystal Reports product. After you create a report format, copy the resulting *.RPT file into the /CRW sub-directory located with the WebSphere Data Interchange Client install directory.

8. When you have completed the specifications for the new report, click OK. You are returned to the Report List window.

   At this point, you can create additional reports, print or preview the report you've just created, or click Close to close the Report List window.

Editing a report

You can change the information displayed in a report by changing the query on which the report is based. If you want to change the list of items generated by the report, you must edit the query on which the report is based or associate a new query with the report. To change the appearance or layout of the report, you must use the Crystal Reports product.

1. Choose the Open Report List command from the File menu.

   The Report List window displays.

2. Use the Functional Area drop-down list to select a functional area within WebSphere Data Interchange Client.

3. In the Report Type list box, click the type of item you want to work with.

4. If you have chosen a report created by WebSphere Data Interchange Client, you can click Edit to view information pertaining to that report.

5. If you have chosen a report that you have created, you can click Edit to edit information in that report.

   The Report Editor window displays.

6. At this point, you can follow steps 6 through 8 of the procedure for creating a new report.

7. When you are finished making the desired changes to the report, click OK.

   WebSphere Data Interchange returns to the Report List window.

   At this point, you can choose another report to edit. If you are finished, click Close to close the Report Editor window.

Deleting a report

1. Click the report in the Report List window to highlight it.

2. Click Delete.

   WebSphere Data Interchange Client asks you to confirm the deletion.

3. If you are sure you want to delete the report, click Yes.

   WebSphere Data Interchange Client deletes the report.

   Note: This does not delete the report layout associated with the report.

Printing or previewing a report

The main reason you create a report is to obtain a printed copy of the results of a query, either for circulation to others, or for creating a permanent printed record if necessary. Previewing the report allows you to view the report pages before they are printed. You can also download the report into various types of files through the Print Preview screen.

To print a report:

1. Choose the Open Report List command from the File menu.

   The Report List window displays.

2. Use the Functional Area drop-down list to select a functional area within WebSphere Data Interchange Client.

3. In the Report Type list box, click the type of item you want to work with.

4. Make a selection in the Report List list box and then click Print. The report is automatically sent to the currently selected printer.

To preview a report:

1. Choose the Open Report List command from the File menu.

   The Report List window displays.

2. Use the Functional Area drop-down list to select a functional area within WebSphere Data Interchange Client.

3. In the Report Type list box, click the type of item you want to work with.

4. Make a selection in the Report List list box and then click Preview.

   The report is displayed on screen, showing you how it would display on paper.

5. To print the report, click Print.

6. To close the screen, click Close.

   Note: If you use the Print Preview function, you see a screen from Crystal Reports. Table 39 provides brief descriptions of the Print Preview controls.

Table 39. Print Preview Screen

This button. . .	Does this:
	Displays the first page of multi-page reports.
	Displays the previous page of multi-page reports.
2 of 2+	Tells you the number of the page currently on display, out of the total number of pages which have been viewed. If all of the pages have not been viewed, a plus sign displays after the second number. (For example, if you are viewing the third page and you have viewed up through page five and there are more than five pages, this display will read 3 of 5+.)
	Stops building long reports.
	Displays the next page of multi-page reports.
	Displays the last page of multi-page reports.
	Prints the selected item.
	Allows you to download the report to a variety of files and formats, including e-mail, HTML, Microsoft Word**, and Lotus Notes®**. See the Crystal Reports User's Guide for details.

Table 39. Print Preview Screen (continued)

This button. . .	Does this:
	Changes the size of the page you are viewing.

Reports: printing and previewing

Chapter 32. Transaction store

WebSphere Data Interchange Client provides the ability to view the transaction store, which is created and maintained by WebSphere Data Interchange Host. You can only view the contents of the transaction store using WebSphere Data Interchange Client if you are in client-server mode.

For a complete description of the transaction store, see the *WebSphere Data Interchange for z/OS Administration Guide*.

About the transaction store

The transaction store is created by and resides on WebSphere Data Interchange Host so that you can maintain a history of all of your translation activities and track those activities. WebSphere Data Interchange Client allows you to view information in the transaction store if you are using client-server mode. The main tools for doing this are WebSphere Data Interchange Client's default transaction store reports and its query functions.

Client overview

You view the transaction store through the transaction store list window, which you access by clicking the transaction store button on the WebSphere Data Interchange Client Navigator bar.

The transaction store list window contains two tabs, Transactions and Interchanges. To view information on Transactions or Interchanges, click the appropriate tab to display a list of transaction store items.

The Transactions List window displays a list of all the transactions in the database regardless of whether they have been enveloped. This list window, however, does not contain envelope information, nor does it display all of the information you can view using WebSphere Data Interchange Host.

The Interchanges List window displays only enveloped transactions and includes envelope information. It also includes the interchange control numbers so that you can view details related to the network and functional acknowledgment of each transaction.

To display additional columns, click the scroll bar on the bottom of the screen to scroll to the right or left. To alter the columns that display on the screen, or to change which query is executed to produce the list, click Modify Window Properties (...). To create new queries, refer to "Creating a query" on page 268.

Host setup

WebSphere Data Interchange maintains the transaction store automatically during translation, unless you have specified otherwise through an application defaults profile. For any given application, you can choose to store:
- All transactions
- Only those transactions that are successful
- Only those transactions that fail
- No transaction information at all

Transaction store: host setup

For more information see Chapter 8, "Application defaults profiles" on page 85. For general information on the transaction store, see the *WebSphere Data Interchange for z/OS Administration Guide*.

Note: Translations done with the PERFORM TRANSFORM command do not use the transaction store.

Using transaction store queries

The transaction store contains a vast amount of data. WebSphere Data Interchange Client provides you with the tools you need to limit the number of transactions that display as a result of running a query.

The transaction store list window runs a Find query as the default query (unlike other WebSphere Data Interchange Client list windows, which run an All query). The Find query allows you to limit the number of transactions that display as a result of the query. When a Find query runs, it displays the Specify Filters Value dialog box, which is described in "Using filters in queries" on page 269.

Table 40 describes the fields for the Transactions query.

Table 41 on page 281 describes the fields for the Interchanges query.

The last word of each field name specifies what type of filter is being run on this field.
- The "Like" operator displays all values that resemble the value typed in the field.
- The "Equal to" operator displays all values that are equal to the value typed in the field.
- The "Between" operator displays all values between the value typed in the From field and the value typed in the To field.

   Attention: You must enter dates in the same format that you have defined for Windows. If you enter a date in a different format than the Windows default, you will receive errors when running queries.

Table 40. Transaction store transaction find query field descriptions

In this field. . .	Type:
Transaction Handle Between [from value]	The first in a range of IDs the transaction store assigns to a transaction. The transaction handle is a 26-byte character field that uniquely identifies each transaction. It is the key around which WebSphere Data Interchange database tables are built. This ID is a concatenation of a date, time, and a sequence number, as follows: yyyy-mm-dd-hh.mm.ss.nnnnnn For send transactions, the date and time indicate when the transactions were translated to the standard. For receive transactions, the date and time indicate when the transactions were de-enveloped. Because it is difficult to know the exact transaction handle, you can type just the date, or date and time.
Transaction Handle Between [to value]	The last in a range of IDs the Transactions Store assigns to a transaction. See the description above for contents and syntax.

Table 40. Transaction store transaction find query field descriptions (continued)

In this field. . .	Type:
TP Nickname Like	The nickname of a trading partner identified in the trading partner profile. The Like operator allows you to use the % wildcard, as described in Table 38 on page 270.
Direction (S/R)	**S** for a send transaction; **R** for a receive transaction.
Standard Transaction ID Equal To	The name of the standard transaction you want to view.
Network ID Equal To	The Network ID of the transactions you want to view.
Internal TP Nickname Like	The Internal Trading Partner ID you want to view.
Application Control Number Like	The Application Control Number you want to include in your query. The value must match exactly the application control value in the data, including upper and lower casing of characters.
Map Name Like	The name of the map upon which the transactions you want to view are based.
Test Transaction (Y/N)	**Y** indicates that you want to view test transactions; **N** indicates that you do not.
Date Created Between [from value]	The beginning date from which you want to see transactions.
Date Created Between [to value]	The ending date to which you want to see transactions.

Table 41. Transaction store interchange find query field descriptions

In this field. . .	Type:
Transaction Handle Between [from value]	The first in a range of IDs the Transactions Store assigns to a transaction. The transaction handle is a 26-byte character field that uniquely identifies each transaction. It is the key around which WebSphere Data Interchange database tables are built. This ID is a concatenation of a date, time and a sequence number, as follows: yyyy-mm-dd-hh.mm.ss.nnnnnn For send transactions, the date and time indicate when the transactions were translated to the standard. For receive transactions, the date and time indicate when the transactions were de-enveloped. Because it is difficult to know the exact transaction handle, you can type just the date, or date and time.
Transaction Handle Between [to value]	The last in a range of IDs the Transactions Store assigns to a transaction. See the description above for contents and syntax.
TP Nickname Like	The nickname of a trading partner identified in the trading partner profile. The Like operator allows you to use the % wildcard, as described in Table 38 on page 270.
Direction (S/R)	**S** for a send transaction; **R** for a receive transaction.
Standard Transaction ID Equal To	The name of the standard transaction you want to view.
Network ID Equal To	The Network ID of the transactions you want to view.

Transaction store queries

Table 41. Transaction store interchange find query field descriptions (continued)

In this field. . .	Type:
Interchange Control Number Like	The interchange control numbers of the transaction you want to select. If you entered 0 in the Envelope type field, the interchange control number is the same as the group control number.
Group Control Number Like	The group control numbers of the transactions you want to select. This field applies only to transactions enveloped as part of a functional group.
Transaction Control Number Like	The transaction set control numbers of the transactions you want to select.
Application Control Number Like	The application control numbers of the transactions you want to select.
Functional Ack Expected (Y)	Enter Y in this field and leave the Functional Ack NOT Expected field blank to display all the transactions for which you expect to receive functional acknowledgments.
Functional Ack NOT Expected (Y)	Enter Y in this field and leave the Functional Ack Expected field blank to display all the transactions for which you do not expect to receive functional acknowledgments.
Functional Ack Received (Y)	Enter Y in this field and leave the Functional Ack NOT Received field blank to display all the transactions for which you have received functional acknowledgments.
Functional Ack NOT Received (Y)	Enter Y in this field and in the Functional Ack Received field blank to display all the transactions for which you have not received functional acknowledgments.
Functional Ack Status Code (A/E/M/P/R/X)	This field allows you to display transactions by Functional Acknowledgment Status Code. Valid values are: Code Description **A** Transactions for which functional acknowledgments have been accepted **E** Transactions for which functional acknowledgments are accepted, but errors are present **M** Rejected—message authentication code failed **P** Partially accepted **R** Transactions for which functional acknowledgments have been rejected **X** Rejected contents after description could not be analyzed
Map Name Like	The name of the map upon which the transactions you want to view are based.

Using transaction store reports

WebSphere Data Interchange Client is shipped with default transaction store reports that allow you to display the most common types of business information. You access transaction store reports through the Report List window, as described in "About reports" on page 273. Select transaction store in the Functional Area drop-down list.

Transaction store reports run a query, either the default Find query or one you create, to display data.

Table 42 describes default Transactions Reports.

Table 43 describes default Interchanges Reports.

Table 42. Default transaction report

This report. . .	Displays:
List of Transactions	A list of transactions as defined in the query you ran to create the report.
Pending Functional Acknowledgments	A list of transactions that have been sent but for which functional acknowledgment transactions have not been received from trading partners.
Transactions Detail	Details of transactions selected in the query you ran to create the report.
Transactions Status Summary	A summary of transactions selected in the query you ran to create the report.

Table 43. Default interchanges reports

This report. . .	Displays:
Interchanges Detail	Details of transactions selected in the query you ran to create the report.
Interchanges Status Summary	A summary of the transactions selected in the query you ran to create the report.
List of Interchanges	A list of interchanges as defined in the query you ran to create the report.
Pending Functional Acknowledgments	A list of transactions that have been sent but for which functional acknowledgment transactions have not been received from trading partners.

Transaction store reports

Chapter 33. Event logs

WebSphere Data Interchange Client provides the ability to view the data from event logs. Multiple event logs can be defined in WebSphere Data Interchange. Each activity log profile refers to a different event log, for which the data is stored in common data storage, but handled as functionally independent data.

About event logs

Event Log is a menu item under View on the tool bar. The default query for event log is a find query that displays a dialog box allowing the user to enter selection criteria for determining which event logs, user IDs, dates, etc. should be used to filter the list of event log entries. You may change the default query using the Preferences option. You can also use the query facility to create custom queries to limit event log searches.

Viewing event logs

The results of an event log query are shown in the list window. Each row contains information about an event log entry, and each column contains data stored about that entry. The Event Log List window also contains the date, time, and user ID of the entry. From the displayed list of event logs, you can select an individual event log for viewing, or a group of them for deleting or printing.
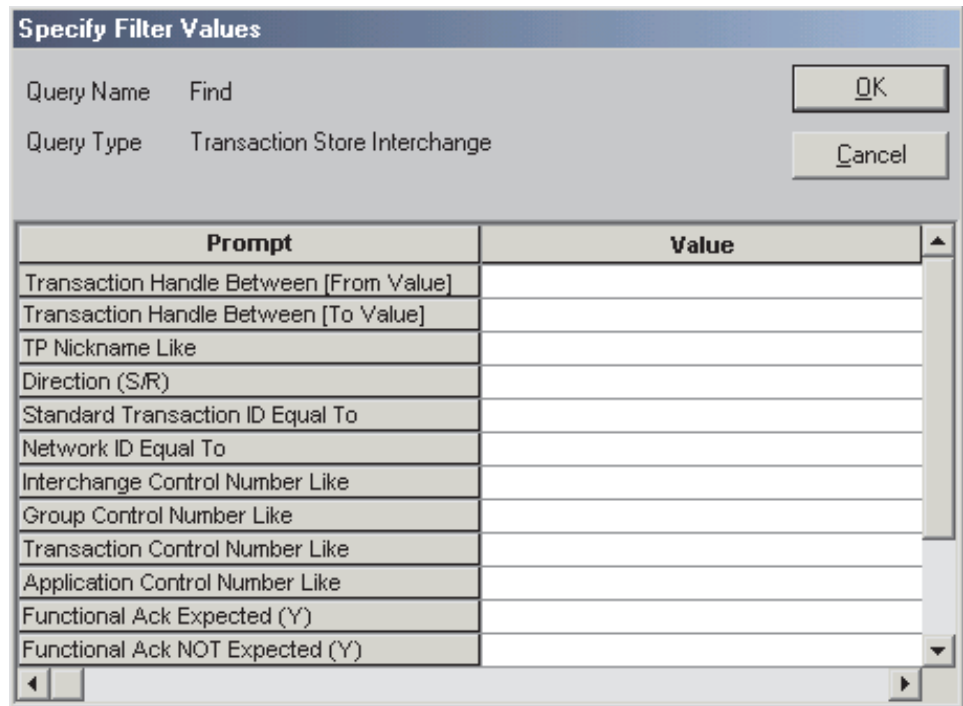
To display additional columns, click the scroll bar on the bottom of the screen to scroll to the right or left. To alter the columns that display on the screen, or to change which query is executed to produce the list, click Modify Window Properties (...). To create new queries, refer to "Creating a query" on page 268.

To view an entry:

1. Select Event Log from the View menu on the tool bar.

Event logs: viewing

The query dialog box displays.

Specify Filter Values				
Query Name	Find			OK
Query Type	Transaction Store Interchange			Cancel

Prompt	Value
Transaction Handle Between [From Value]	
Transaction Handle Between [To Value]	
TP Nickname Like	
Direction (S/R)	
Standard Transaction ID Equal To	
Network ID Equal To	
Interchange Control Number Like	
Group Control Number Like	
Transaction Control Number Like	
Application Control Number Like	
Functional Ack Expected (Y)	
Functional Ack NOT Expected (Y)	

2. Enter your selection criteria and click OK.
3. The Event Log List window displays.
4. Double-click the Event Log entry in the list window that you want to view.

   The detail dialog box displays. Click [?]
   for field descriptions.

Part 6. Appendixes

Appendix A. Mapping Binary Data

The ASC-X12 Specifications/Technical Information transaction set (841) defines a way for trading partners to exchange technical information the same way they exchange EDI transactions. This technical information, which can be graphic, image, or audio, can contain binary data. The binary data can assume any value in the range X'00' rough X'FF'.

In the syntax of X12 transaction sets, data elements that are separated by delimiters are combined into a segment that is identified by a segment ID and terminated with a segment delimiter. The binary data introduced by the 841 transaction set causes problems for this syntax because the binary data may contain a value that matches a segment delimiter. Translators and networks that support the 841 transaction set must have a way to identify binary data and determine its length so that it does not interfere with parsing the rest of the envelope. Special care must be taken if you want to send and receive files between WebSphere Data Interchange and translators on other platforms. Not all operating systems support the record types z/OS uses. For more information, see "Format specifications" on page 293.

The BIN segment ID

The binary data is identified with a BIN segment ID, which notifies the parser that data following the segment ID is binary. Although the parser must always treat the BIN segment as if it contained binary data, the segment can contain normal text. The first data element of the BIN segment contains the length of the binary data so that the parser knows the amount of data to pass without interference. The first character after the binary data should be BIN segment terminator. Any other value is a syntax error that ends parsing for the envelope.

The BIN segment ID triggers the special binary processing. Although the 841 transaction set is the only one that uses the BIN segment, binary processing is not limited to the X12 standard. In addition, WebSphere Data Interchange applies this special processing to all envelope types.

Length of the BIN segment

The value in the length data element of the BIN segment can be up to 15 characters long, which means the maximum length of a BIN segment is 999,999,999,999,999 bytes (fifteen 9s). However, the maximum size that WebSphere Data Interchange supports is 2,147,483,647 bytes, which is the maximum signed integer value that a fullword can hold. The binary segment is a repeating segment with an unlimited number of repetitions, therefore it is unlimited for EDI.

Data Transformation for Binary Data

This section covers mapping binary data for data transformation to and from an application field.

Mapping a binary segment

The BIN segment, as mentioned earlier, can have a rather impressive length. A new segment BDS has been introduced to transmit binary data in X12V4R4, transaction 102 (Associated Data).

The BIN and BDS segments

The binary data is identified with a BIN or BDS segment ID, which notifies the parser that data following the segment ID is binary. Although the parser must always treat the BIN and BDS segments as if it contained binary data, the segment can contain normal text.

The first data element of the BIN segment contains the length of the binary data so that the parser knows the amount of data to pass without interference.

The second data element of the BDS segment contains the length of the binary data so that the parser knows the amount of data to pass without interference.

The first character after the binary data should be BIN or BDS segment terminator. Any other value is a syntax error that ends parsing for the envelope.

The BIN and BDS segment IDs triggers the special binary processing.

Although using a file name is the primary way of providing data for a binary Segment, with Data Transformation maps you provide the data by simply passing it to WebSphere Data Interchange in application fields, the way all other application data is passed. The binary data itself must be defined in the Data Format with a data type of BN. The application may provide the length of the binary data If the length of the binary data is not provided by the application the length will be the length of the application data field defining the binary data.

However, WebSphere Data Interchange has a maximum length of 32767 bytes for application fields, which is significantly less than the 999,999,999,999,999 defined by the standard or the 2,147,483,547 that WebSphere Data Interchange allows. WebSphere Data Interchange knows that it is dealing with binary data, and that some special processing is needed to put it into a BIN or BDS segment. The easiest way to pass this binary data to WebSphere Data Interchange is with a data format such as:

```
BINARY_LOOP1   999 Repeating loop
BINARY_STRUCTURE1 Record
     LENGTH N0 15 15-byte length field
     BINARY_DATA_LOOP 99 Repeating loop
         BINARY_DATA1 Record
             BINARY_FIELD BN 32767 Binary data field
```

Both BINARY_STRUCTURE1 and BINARY_DATA1 are passed separately records.

Now the application provides WebSphere Data Interchange with a BINARY_STRUCTURE1 and as many BINARY_DATA1 records as necessary to satisfy the length specified in the LENGTH field. WebSphere Data Interchange builds one BIN or BDS segment for each occurrence of BINARY_STRUCTURE1.

The BIN and BDS segment has a length value as specified by the LENGTH field and contains data from the BINARY_DATA1 records.

The EFI segment

For the 841 transaction set, an Electronic Format Identification (EFI) segment precedes a group of repeating BIN segments. The EFI segment provides information about the binary data file. WebSphere Data Interchange provides special processing for only the following data elements in the EFI segment:

- File name

- Block type
- Record length
- Block length

Data elements such as *security technique* and *compression technique* do not receive special processing. When data compression is used, the sender must compress the data before translation, and the receiver must decompress it after translation. Any security that occurs is for the entire transaction or for the group containing the transaction. WebSphere Data Interchange does not provide a special security interface for data in the BIN segments.

Use of the four data elements listed above are covered later in the discussions of send and receive processing.

The following information may help explain how WebSphere Data Interchange treats two of them: file name and block type.

File name: The EDI standard essentially defines file name as free format with a maximum length of 64 characters. Its format depends on the computer operating system being used. Accordingly, WebSphere Data Interchange treats this element as a fully qualified data set name, including the owner ID. A file name value with a length greater than 44 characters (the maximum length of a data set name) will be ignored.

Block type: The standard defines block type as free format with a maximum length of 4 characters. Its value indicates the organization of data in the BIN segments. Examples are fixed length, variable length, and spanned.

However, the definition does not provide any codes to represent the organizations, such as F for fixed and V for variable. In the absence of standard codes, WebSphere Data Interchange interprets the block type as a string of characters that can have the following values:

A   ASA printer control characters

B   Block

F   Fixed

M   Machine code printer control characters

S   Spanned

U   Undefined

V   Variable

Send processing for the binary segment

This section covers mapping a file or an application field to a binary segment.

Mapping data from a file to a binary segment

The primary intent of the 841 transaction set and the BIN segment is to transmit files between trading partners. Therefore, WebSphere Data Interchange provides a way for you to map a *file* to a data element in the EDI standard. Normally, you would map a data element to an application field. Mapping data from a file to the BIN segment requires a new application data type, FN (file name).

Binary segment: mapping from file

The FN data type has special meaning only when it is mapped to the BIN02 data element of the BIN segment. At all other times, the FN data type is treated as an alphanumeric (AN) field. The special meaning for the FN data type when mapped to the BIN02 element is as follows:

- Rather than moving the field containing data to BIN02, WebSphere Data Interchange moves the entire contents of a file named by the field to the BIN02 data element.
- WebSphere Data Interchange automatically supplies the length (BIN01), based on the amount of data read from the file.

Aside from the special processing described above, the mapping of an FN field is the same as any other field, including the use of a translation table or user exit routine. For example, the application may not know the data set name of a file to be transmitted. The application may know it by a coded name. A translation table or a user exit provides a way of transforming the coded name to a data set name. You can also use a literal if the application data does not contain a value at all or if the value it supplies is all blanks. Because the name of the file mapped to the BIN segment can also be specified in the EFI segment, you should use the same field for mapping both segments.

If an EFI segment is built and it contains a value in the file name data element, that file name is used if the field mapped to the BIN02 data element is all blanks and no literal value is supplied.

The format of data in an FN type field is:

```
type:name
```

In this format, `type` is an optional value that indicates the type of file being provided. Valid values are as follows:

DD   Data definition name (ddname)

DS   Data set name

MQ   WebSphere MQ queue profile member name

VS   VSAM entry sequenced data set

TD   Transient data queue

TM   Temporary storage queue (main)

TS   Temporary storage queue (auxiliary)

Note: VS, TD, TM, and TS are for CICS only. The default is DS in z/OS and TS in CICS.

The `name` is either a ddname or data set name based on the value of `type`. It must conform to the conventions for its type. A ddname has a maximum length of 8. A data set name has a maximum length of 44.

The value of name can also be the literal &IV, which tells WebSphere Data Interchange to substitute a value that represents the current binary file being processed. Each time a binary file is processed, WebSphere Data Interchange increments this internal number. Thus, a specification of DD:EDIBF&IV equates to a ddname value of EDIBF1 the first time a binary file is processed, EDIBF2 the second time a binary file is processed, and so on.

The BIN segment, as mentioned earlier, can have a rather impressive length, and you may think that the entire file can be put into a single BIN segment. This is not always true. The number of BIN segments required to hold any file depends on the size of the file and the format of records in the file.

Format specifications

A file should be transmitted in a form that permits the receiver to recreate an exact copy of the file.

For z/OS based systems, a file consists of individual records and each record consists of some number of characters. Records can either have a fixed format where each record has the same number of characters, or a variable format where each record has a variable number of characters. For variable length records, the number of characters in the record is maintained in a record header which is the first 4 bytes of the record and has a format of LLXX, where LL is the number of characters in the record (including the length of the LLXX field) and XX is reserved.

Files on other systems (such as Windows or AIX) do not have record boundaries but are treated as a stream of characters with record boundaries (if any) established by the program that is reading the stream. WebSphere Data Interchange provides 8 different ways to combine data from a file into BIN segments so that the receiver of the file can recreate an exact copy of the file being sent. The 8 different ways represent combinations of how records are put into binary segments (either combined or individually) and the format of the record header within the BIN segment. There are four possible header formats for records within the BIN segment:

1. No header - the data is put into the binary without a header.
2. z/OS format - each record in the binary segment is preceded by a record header of the form LLXX.
3. SHORT format - each record in the binary segment is preceded by a record header of the form LL.
4. LONG format - each record in the binary segment is preceded by a record header of the form LLLL.

WebSphere Data Interchange, by default, transmits all files with a fixed format by combining the records into a single binary segment without any record headers. Because the file has a fixed format, the record headers are not needed for the receiving side to reconstruct the file.

WebSphere Data Interchange, by default, transmits all files with a variable format by sending each record within the file as a single binary segment. Again record headers are not used because they are not necessary to be able to reconstruct the file.

If the defaults are not satisfactory, (for example, you want to send a file with variable length records as a stream of data in a single binary segment) then a special literal value can be supplied at mapping time that overrides the defaults. This literal is specified when mapping the binary element in the BIN segment (BIN02). You can either type it in the literal field on the mapping screen, or map an application field to the BIN02, which at run time supplies the literal to WebSphere Data Interchange. This literal consists of a series of keywords. Except for the first keyword (BinSpec), you can specify them in any order. In the following list, the keywords are shown with acceptable abbreviations in CAPS:

Format specifications

BinSpec

Signals that binary specifications follow.

BINary(C)

Indicates records should be combined into a single binary segment (default for fixed record lengths).

BINary(R)

Indicates each record should be a binary segment (default for variable record lengths).

Format(N)

No header should precede each record in the binary segment (default).

Format(V)

Use a header of the form LLXX, where LL is the number of bytes in the record and XX is binary zeros.

Format(S)

Use a header of the form LL, where LL is the number of bytes in the record.

Format(L)

Use a header of the form LLLL, where LLLL is the number of bytes in the record.

LIMIT(*nnnnn*)

If records are being combined, then *nnnnn* is the maximum size for a binary segment. After a binary segment reaches this limit, it is stopped and a new binary segment is started. The default limit is over 2 gigabytes or the amount of virtual storage available to the program, whichever is lower.

Examples

Sending PC executable files

In order to send a Windows or AIX executable file received from a personal computer back to a personal computer, use the combined and no headers options:

```
BINSPEC BINARY(C) FORMAT(N)
```

Note: A variable length file sent with the above options cannot always be recreated, because the records have been combined and no record length information has been added. This format is needed to send data to some PC translators.

Sending variable length files to PCs

In order to send a variable length file (such as a LIST3820) to a personal computer, so that none of the record length information is lost, even if all of the records are combined into one file, use the combined and any format option other than N.

```
BINSPEC BINARY(C) FORMAT(S)
```

In order to convert a file formatted as described above back to its original format, use the same options on receive processing.

Sending a file to a system with a limit

Some systems have a limit to the size of a binary segment they can process. In order to limit the size of the data inside the BIN02 element to 1000 bytes, use the following option:

```
BINSPEC LIMIT(1000)
```

Note: A variable length file with records larger than 1000 bytes sent with the above options cannot always be recreated, because the records have been split and no record length information has been added. The format parameter could also be used to add record length information.

Mapping an application field to a binary segment

Although using a file name is the primary way of providing data for a binary segment, you might find it useful to provide the data by simply passing it to WebSphere Data Interchange in application fields, the way all other application data is passed. If you do this, the application provides the length of the binary data and the binary data itself.

However, WebSphere Data Interchange has a maximum length of 999 bytes for application fields, which is significantly less than the 999,999,999,999,999 defined by the standard or the 2,147,483,547 that WebSphere Data Interchange allows. WebSphere Data Interchange knows that it is dealing with binary data, and that some special processing is needed to determine the length of data being passed, gather it, and put it into a BIN segment. The easiest way to pass this binary data to WebSphere Data Interchange is with a data format such as this:

```
BINARY_LOOP1  999  Repeating loop
    BINARY_STRUCTURE1 Record
        LENGTH N0 15 15-byte length field
        BINARY_DATA_LOOP 99 Repeating loop
            BINARY_DATA1  Record
                BINARY_FIELD CH 999 Binary data field
```

Both BINARY_STRUCTURE1 and BINARY_DATA1 are passed separately. Now the application provides WebSphere Data Interchange with a BINARY_STRUCTURE1 and as many BINARY_DATA1 structures as necessary to satisfy the length specified in the LENGTH field. WebSphere Data Interchange builds one BIN segment for each occurrence of BINARY_STRUCTURE1.

The BIN segment has a length value as specified by the LENGTH field and contains data from the BINARY_DATA1 structures.

If the BINARY_DATA1 structures you provide exceed the length you specified, WebSphere Data Interchange ignores the extras. If you provide fewer than needed, binary zeros are added to the BIN segment until the LENGTH value is satisfied. The number of repetitions you specify for the structures is not important. WebSphere Data Interchange accepts as many as the application passes.

As another example, suppose the application wants to pass in data from the file and have structures built in the same way described above for variable-length files. The normal variable-length record format for z/OS records is a binary halfword containing the length of the data followed by the data itself. The data format definition required to accomplish this is:

```
BINARY_LOOP 999 Repeating loop
    BINARY_STRUCTURE2   Record
        LENGTH  I0 2 2-byte binary length (the I0 is I (as in I am) 0)
        BINARY_DATA_LOOP 32767 Repeating loop
            BINARY_DATA2 ST   Record
                BINARY_FIELD CH 1 Binary data field
```

BINARY_DATA2 is defined as physically part of its parent. Its maximum use count is 32756, because that is the maximum logical record length for z/OS physical sequential files.

Binary segment: mapping from application field

The only difference between this example and the previous one is in the way data is passed to WebSphere Data Interchange. In the first case, BINARY_STRUCTURE1 is passed followed by multiple BINARY_DATA1 structures. In the second case, only BINARY_STRUCTURE2 is passed because BINARY_DATA2 is defined as physically part of BINARY_STRUCTURE2.

There is a trade-off in choosing between the two cases. Structures always have a fixed length in WebSphere Data Interchange. Therefore, when WebSphere Data Interchange is presented with BINARY_STRUCTURE2, it always expects to get 32758 bytes. On the receive side, it always creates 32758 bytes. In effect, each variable-length record is expanded to its maximum length. The method used with BINARY_STRUCTURE1 conserves main storage at the expense of making the application break up the data into 999-byte chunks, in order for WebSphere Data Interchange to put it back together.

You might ask, "But a structure can repeat only 32757 times, and a field can be only 999 bytes long, for a maximum record size of 32,724,243 bytes? What if I want to pass in more than that as a single structure?" WebSphere Data Interchange assumes that if the field mapped to the binary data element of the BIN segment is the first field of a structure, then the length of the *structure*, not the length of the field, is used to move data. Furthermore, if the structure is defined to be physically part of its parent, and its parent is defined to be physically part of its parent, and it has no other fields, WebSphere Data Interchange uses the length of the parent structure. For example:

```
BINARY_LOOP 999 Repeating loop
   BINARY_STRUCTURE3   Record
      LENGTH N0 15 15-byte length field
      BINARY_DATA_LOOP1 10000 Repeating loop
         BINARY_DATA3  Record structure
         BINARY_DATA_LOOPA 1000 Repeating loop
            BINARY_DATA3A   Record
            BINARY_DATA_LOOPB 10000 Repeating loop
               BINARY_DATA3B  Record
                  BINARY_FIELD1 CH 999 Binary data field
                  BINARY_FIELD2 CH 1 Binary data field
```

BINARY_DATA3, BINARY_DATA3A, and BINARY_DATA3B are all defined to be physically part of their parent. Using this definition and mapping the binary data field in the standard to BINARY_FIELD1, the translator arrives at a length value of 1000 (the length of BINARY_FIELD1+BINARY_FIELD2) * 10000 (the number of times BINARY_DATA3B repeats) * 10000 (the number of times BINARY_DATA3A repeats) * 10000 (the number of times BINARY_DATA3 repeats). This yields an effective length of 10 to the 15th (1 greater than allowed by the standard).

Of course this structure could not really be used because the maximum length allowed by WebSphere Data Interchange is 2,147,483,647. Nevertheless, the example illustrates the technique of creating an effective length greater than 999.

Receive processing for the binary segment

This section covers mapping a binary segment to a file or an application field.

Mapping data from a binary segment to a file

Receiving presents a set of problems that are not present on the send side. During send processing, it is reasonable to assume that senders know what they are sending, when they are sending it, and the data set names of the files they are sending. Receivers, on the other hand, can have little control over what they

receive, when they receive it, and how many files they receive at any one time. Therefore, WebSphere Data Interchange must be capable of creating files when necessary.

As with sending, if you want the binary data passed to you in a file rather than the application fields, map the binary data to a field in the application defined with the FN data type. You will not receive the binary data in the field; instead you will receive the name of the data set the data was written to.

On the send side, you can provide the file name in an application field buffer or as a literal value. You can also provide a literal on the receive side. If you provide a literal for an FN data type that is mapped to the binary data element of the BIN segment (BIN02), WebSphere Data Interchange gives the literal special processing. Normally, a literal mapped for receiving is used only if the standard data element is blank. The special processing for this literal is that you can use it to provide the file name and allocation parameters that WebSphere Data Interchange needs to create the file on your system.

The literal has this format:

```
type:name parameters
```

type

An optional value that indicates the type of file being provided. Valid values are:

DD   Data definition name (ddname)

DS   Data set name

MQ   WebSphere MQ queue profile member name

VS   VSAM entry sequenced data set

TD   Transient data queue

TM   Temporary storage queue (main)

TS   Temporary storage queue (auxiliary)

VS, TD, TM, and TS are for CICS only. The default is DS in z/OS and TS in CICS.

name

is either a ddname or data set name based on the value of *type*. It must conform to the conventions for its type. A ddname has a maximum length of 8 characters. A data set name has a maximum length of 44 characters.

name can also contain one of the following special literals to substitute other values in this field:

&IV   A value that represents the current binary file that is being processed. An F is inserted in front of this value if the IV value begins another level of qualification on the data set name.

&U   The current user ID. A period is inserted after the user ID to force another level of qualification on the data set name.

&D   The current date as a yymmdd value. D is inserted in front of this value if it begins another level of qualification on the data set name.

&T   The current time as an hhmmss value. A T is inserted in front of this value if it begins another level of qualification on the data set name.

Binary segment: mapping to file

&E The current file name from the EFI segment. Another level of qualification is forced both before and after the EFI file name.

name is optional. If you do not provide it, the file name from the EFI segment is used. If an EFI segment does not exist or does not contain a file name field, a default data set name &U.BIN&V.D&D.T&T is used. The name that is used is returned to the application in the FN field. The maximum length of a data set name is 44 characters, and truncation occurs if necessary.

parameters

A series of keywords and values that supply the data needed to create a new data set on your system. Except for the first keyword (ALL), you can specify them in any order. The keywords are shown with an acceptable abbreviation in CAPS:

ALLocate
Signals that allocation parameters follow.

TRacKs
Allocation unit is tracks.

CYLinders
Allocation unit is cylinders. This is the default value if TRK or BLK is not specified.

BLocKs(*b*)
Allocation unit is BLOCKS with an average block length of *b*.

Space(*p*, *s*)
Primary space quantity of *p* (default value of 10). Secondary quantity of *s* (default value of 10).

UNit(*vvvv*)
vvvvv is the unit name for the allocation. Default value is SYSDA.

UCount(*n*)
n is the number of units for the allocation. Default value is 1.

Lrecl(*l*)
l is the logical record length. Default value is 32756 or the value taken from the EFI segment.

Blksize(*b*)
b is the block size. Default value is 32760 or the value taken from the EFI segment.

Recfm(*xxxx*)
xxxx is the record format for records in the file. Default value is VB or the value taken from EFI segment. Valid values are:

A	ASA printer control characters
B	Block
F	Fixed
M	Machine code printer control character
S	Spanned
U	Undefined
V	Variable

LIKE(dsname)

> dsname is the name of some other data set from which the LRECL, BLKSIZE, and RECFM are copied.

WebSphere Data Interchange first tries to open a given data set name. If the attempt to open the data set is successful, the file is used, and any data in the file is overlaid by the new data. If the attempt fails, WebSphere Data Interchange tries to create a new file using the allocation value (or defaults) shown above. The new data set has a specification of (NEW,CATLG,CATLG), and all unused space in the data set is released when the data set is closed. For WebSphere Data Interchange for CICS, if the attempt to open the data set fails, a unique temporary storage queue is created, and the data is written to this queue. Default values are overridden by values from the EFI segment, which in turn are overridden by literal values supplied at mapping time.

If the data being received is being written to a fixed-length file, WebSphere Data Interchange assumes the binary segment consists of multiple records without any record headers. If the data being received is being written to a variable-length file, WebSphere Data Interchange assumes each binary segment contains data for a single record without any record headers.

WebSphere Data Interchange takes data from the binary segments and adds them to the file. For each BIN segment, the amount of data defined by the logical record length of the file is extracted and written to the file. This continues until the amount of data left in the BIN segment is less than or equal to the logical record length. If the amount of data left in a binary segment is less than the logical record length, fixed-length records are padded with binary zeros; variable-length records are written with the shortened logical record length. Data from the end of one BIN segment is never combined with the data from the beginning of the next BIN segment. At a minimum, each BIN segment defines the beginning of a new record in the file.

If the defaults previously described are not desired, then the binary specifications described in "Format specifications" on page 293 (BINary and Format) can be provided as part of the allocation parameters in the receive literal value.

Mapping data from a binary segment to an application field

The considerations described for mapping from an application field to a binary segment also apply when mapping data from a binary segment to an application field. See "Mapping an application field to a binary segment" on page 295.

Appendix B. Data Transformation mapping commands and functions

In addition to allowing you to map a source element to a target element by dragging and dropping, the Data Transformation Map editor includes a powerful set of commands. This allows you to use conditional logic such as if/then/else, qualify repeating elements, save values in variables, create complex expressions, and many other functions. This appendix describes the command language used in the Data Transformation Map editor to perform these tasks.

When you create a map, you are basically creating a sequence of commands. Each command is attached to an object in the source or target document, such as a segment, data element, field, XML element, and so on. Whenever the translator finds an object in the source or target document, it will execute any commands associated with it. You can also add commands before and after objects in the document. The translator will execute these commands, even if the elements before and after are not found. You can create the commands by dragging and dropping a source element to a target element (or a variable) or vice versa. This results in a **MapTo** (for source-based mapping) or **ForEach** (for target-based mapping) command or an assignment command being created. You can also create commands by right-clicking a node in the map command display to start the mapping command wizard.

The following sections describe the supported commands and functions, and provides information about other important features of the mapping language.

Map variables

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 by the user. WebSphere Data Interchange supports three types of variables: *local*, *global*, and *special variables*.

Local variables are unique to the map they are defined in. 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. A loop variable does not disturb the value of another variable with the same name at another level of looping. Local variables will be maintained within the map in which they are defined. You can add, delete, and alter the properties of any local variable.

Special variables are a group of predefined variables used by WebSphere Data Interchange. They function much like local or global variables, except they each have a special purpose. A user can view properties of a special variable, but no changes can be made. Special variables always start with "DI", which is reserved.

Special variables are write-only. That means you can set the value of a special variable, but cannot read it or use it in an expression.

Map variables

Table 44 lists the special variables supported:

Table 44. Supported special variables

Name	Scope	Type	Length	Description
DIOutFile	Document	Character	8	Specifies the name of the file to which the translated data will be written. It will override any value that was specified in the data transformation rule or data format definition.
DIOutType	Document	Character	2	Specifies the file type to which the translated data will be written to. It will override any value that was specified in the data transformation rule or data format definition.

Naming variables

Variable names can be up to 30 characters long and can contain the characters `0-9`, `A-Z`, `a-z`, and the special characters "@#$_". Special variables always start with "DI". Local variables cannot start with a number, dollar sign, or ampersand. Variables cannot be the same as any reserved word (command name, function name, or other reserved word).

Names can be in mixed case. Case is not used in determining uniqueness, for instance, the variable "MyValue" is the same as the variable "MYVALUE".

Note: During translation, WebSphere Data Interchange might use variables whose names start with $R to hold intermediate values. These variables are not visible in the map, but might appear in messages during translation.

Literals

The Data Transformation Map editor uses the term "literals" differently from the send/receive map editor. In the Data Transformation Map editor, the term "literal" refers to a static value, such as a default value. The "special literals" used in send and receive maps are replaced by "mapping commands" in data transformation maps.

Literals can be either numeric or character strings. Character strings can be contained in either single or double quotes. For example:

"This is a character literal"

'This literal contains "quotes" within'

Numeric literals are made up of a sequence of digits, and can include a decimal point. (Negative numbers can also be used, but the negative sign is considered separate from the numeric literal.) For example:
```
99
1.23
```

Comments

The Data Transformation Map editor allows the user to enter comments by creating comment nodes in the mapping commands window. A comment group node allows the user to add several comments within a single node.

Keywords

This section identifies keywords that have special meaning to WebSphere Data Interchange. In addition to the keywords listed in this section, all mapping commands, logical operators, comparison operators, and arithmetic operators are considered keywords.

False   Used as a boolean value to test boolean expressions and set boolean variables.

targetRoot
Can be used on the **MapCall** command. See "Commands" on page 308 for more information.

This   Can be used as a source or target path in mapping commands to indicate the current source or target element.

True   Used as a boolean value to test boolean expressions and set boolean variables.

Specifying a path

Paths are used to identify a source or target element in various mapping commands. To specify a source or target path in a command, simply drag the element from the source or target tree display to the command window.

The path indicates the position of a compound element or a simple element in the source or target document definition. Paths are used in various mapping commands to identify the compound element or simple element involved in the command. Paths are syntax dependent. For instance, all of the path examples below use paths from EDI standard X12, version 3, release 6, transaction 837.

A path can include compound element identifiers and simple element identifiers. The path begins with a backslash to indicate that it starts at the root of the document. Each element of a path is separated by a backslash. The path is always terminated by two backslashes. An example of a path is:

```
\T 2\L 2300\S CLM 130\C C023 5\E 1332 2\\
```

This path translates to: table 2, loop Id 2300, CLM segment at position 130, composite data element C023 at position 5, and component data element 1332 at position 2.

The following path refers to the CLM segment at position 130 within table 2 and loop Id 2300:

```
\T 2\L 2300\S CLM 130\\
```

Note: Source document elements are read-only. You cannot update values in the source document. Target document elements are write-only. You cannot read the value that has been assigned to an element in the target document.

Forward and reverse references

WebSphere Data Interchange always has a *current position* in the source data. The current position is the path just located and/or obtained from the source data.

In source data, WebSphere Data Interchange is able to locate paths or simple elements before or after the current position. Doing this does not change the current position. The forward or reverse reference can be helpful in examining simple data or determining if a particular path exists.

For repeating items (loops, segments, elements, and so on), you cannot refer to a previous or later occurrence of the item. For example, if you are currently processing the second occurrence of a repeating segment, you cannot refer to an element from the first or third occurrence of the segment. If you have not yet started processing a repeating item, the first occurrence of the item is considered the current occurrence.

A more efficient option to using references is to save the information you need in variables. Then use the variable in place of the forward or reverse reference.

Forward and reverse references can be specified in expressions. A reference is specified by using a path.

Data types supported by mapping commands and functions

WebSphere Data Interchange supports a wide variety of data types for simple elements. When the input data is parsed, the values are kept internally as one of the following data types:

character   Character data
int   Integer data
real   Floating-point data
boolean   Boolean data
binary   Binary data

Variables can also be any of these data types. Literals are always either real or character. Implicit type conversion is done by the translator when an element, variable, or literal is assigned to another type of element or variable, or is used in a function or command that expects a different type. Integer values can be converted to real or character; real values can be converted to integer or character; character values can be converted to integer or real values. If an invalid type conversion is attempted, for example trying to assign a boolean variable to a character variable, an error will be issued during the control string compilation.

Many of the commands and functions expect numeric values for their arguments or return a numeric value. All functions, commands, and expressions process numbers internally using data type real. However, because real values are implicitly converted to or from integers, integer variables and elements can be used in place of any of the real arguments or return values described below.

Expressions

Expressions are commonly used in the mapping command language. The basic form of an expression is:

```
token operator token [operator token [operator token [...]]]
```

Where:

token One of the following:
- A variable, either local, global, or special
- A path identifying a source simple element
- A numeric constant, such as 1024
- A string constant, such as 'ABCD'
- A mapping function

operator One of the following types of operators:
- Logical
- Comparison
- Arithmetic
- Unary

Quotation marks must be used at the start and end of a string constant. Either single (') or double quotes (") can be used, but whichever is used to start the string constant must also be used to end it.

All variables, literals, and simple source and target elements have a specific data type. The translator will convert the value to the appropriate type for the expression if needed. If it cannot convert the value to the appropriate type because the types are incompatible (for example, boolean to binary), an error will be issued during the control string compilation. If it cannot convert the value to the appropriate type because the values are incompatible, (for example "abc" to real), the translator will issue an error or warning during translation.

Logical operators

Data transformation maps allow logical operators **AND**, **OR** and **NOT**.

AND returns a value of **True** if both conditions are true. **False** is returned when either condition is false.

OR returns a value of **True** if either condition is true. **False** is returned when neither condition is true.

NOT reverses the boolean result of an expression. For logical expressions, **NOT(**expression) will return **False** if the result of **expression** is **True**. **NOT(expression)** will return **True** if **expression** is **False**.

NOT has a short form that can be specified as an exclamation point (!). For example, the following are valid:

```
NOT(expression)
!(expression)
NOT(myBoolFlag)
!myBoolFlag
```

Case is not relevant with the logical operators. They can be entered as uppercase, lowercase, or mixed.

Comparison operators

Comparison operators tell the WebSphere Data Interchange translator to compare two numeric objects. WebSphere Data Interchange processes comparison operators in the same way that it handles logical operators. If the comparison is true, **True** is returned. If the comparison is false, **False** is returned. Table 45 on page 306 lists the comparison operators supported.

Operators

Table 45. Comparison operators

Operator	Alternate	Description
EQ	=	The first value is the same as the second
GE	>=	The first value is greater than or equal to the second value
GT	>	The first value is greater than the second value
LE	<=	The first value is less than or equal to the second value
LT	<	The first value is less than the second value
NE	!= or <>	The first value is not equal to the second value

Case is not relevant with the comparison operators. They can be entered as uppercase, lowercase, or mixed.

Note: The comparison operators are for numeric (integer and real) data types only. Character strings can be compared using the string comparison functions.

Arithmetic operators

Data transformation maps allow addition, subtraction, multiplication, and division of numeric values.

Data transformation maps also allow the additional operator modulus. The modulus operator (%) will return the remainder of the first operand divided by the second.

Division by zero using the division operator ("/") will produce an error. Division by zero using the modulus operator ("%") will also produce an error.

Unary operators

One unary (single component) operator is supported by the WebSphere Data Interchange translator, "–". This operator changes the sign of the expression it precedes. For example, if the value of *var1* is 9, specifying *–var1* returns –9. It does not change the value of *var1*.

Order of precedence

During processing, all expressions are evaluated from left to right. The order of precedence is:
1. Unary minus (-)
2. Multiple (*), Divide (/)
3. Modulus (%)
4. Addition (+), Subtraction (-)
5. Relational Operators (GT, GE, LT, LE, EQ, NE)
6. Logical NOT Operator
7. Logical AND Operator
8. Logical OR Operator

Precedence can be overridden by using parentheses within an expression. For example, 2+3*5 will equal 17 because the multiplication is done first, then the addition. (2+3)*5 equals 25 because the parentheses indicate that the addition is done first, then the multiplication.

Assignment

A value can be assigned to any variable or any target simple element. This is accomplished using an assignment statement. An assignment statement has the following format:

```
target = expression
```

Where:

target Is any defined variable or simple element in the target document definition.

expression Is any valid expression.

WebSphere Data Interchange will attempt to convert the result from *expression* to the same data type of *target*, if needed. If it is unable to make the conversion, an error will be issued.

Conditional commands

If / ElseIf / Else / Endif

The If command is used to conditionally perform one or more mapping commands. The If command uses the following format:

```
If (condition)
 mapping commands
[ElseIf (condition)
 mapping commands]
[ElseIf (condition)
 mapping commands]
[Else
 mapping commands]
EndIf
```

Where:

condition Any valid expression that evaluates to **True** or **False**.

mapping commands One or more mapping commands.

The If command marks the beginning of the If condition block. EndIf is used to mark the ending of the If condition block.

When the If command is encountered by the translator, the *condition* will be evaluated. When *condition* evaluates to **True**, the mapping commands immediately following the If command will be executed. If *condition* evaluates to **False**, the translator will look for an ElseIf statement within the If condition block. If an ElseIf statement is found, its associated *condition* will be evaluated. When *condition* evaluates to **True**, the mapping commands immediately following the ElseIf statement will be executed. If *condition* evaluates to **False**, the translator will look for the next ElseIf statement within the If condition block. When all **ElseIf** statements have been tested and evaluated to False, the translator will look for an **Else** statement within the If condition block. If an Else statement is found, the mapping commands immediately following the Else statement will be executed.

The ElseIf and Else statements are optional. When present, an Else statement always precedes the EndIf statement. ElseIf statements always follow the If statement and precede the Else statement (if present) or the EndIf statement (when the Else statement is not present).

The translator will examine the If and ElseIf statements sequentially until an associated *condition* evaluates to **True**. When a *condition* has evaluated to **True**, only the mapping commands associated with the corresponding If or ElseIf statement will be executed. After the mapping commands have been executed, the translator will exit the If condition block and resume processing after the EndIf statement. If no *condition* evaluates to **True**, the mappings commands associated with the Else statement will be executed and the translator will resume processing after the EndIf statement. If no *condition* evaluates to **True** and there is no Else statement, no mapping commands within the If condition block will be executed and the translator will resume processing after the EndIf statement.

Commands

Mapping commands all perform a specific action as described below. The command name is not case sensitive. For example, the Error command could also be specified as ERROR. Most commands can take any expression for their arguments, as long as the expression evaluates to the appropriate type (or its result can be converted to the appropriate type). The only exception is when the argument is specified as a *sourcepath* or *targetpath*. For those commands, the argument must be a source or target path. Below is a description of each of the supported commands.

CloseOccurrence

Use the CloseOccurrence command to close the current occurrence of a repeating element and force the creation of another instance of the element. This command can be used for both source-based and target-based maps.

The CloseOccurrence command has the following format:

```
CloseOccurrence(targetpath)
```

Where:

targetpath   Specifies a repeating target element that is ″closed″ to any new elements in the current occurrence. Any additional mappings to this element or its children result in a new occurrence of the *targetpath* element.

The CloseOccurrence command can be shortened to CloseOccur.

Example
Suppose your source document has the following structure:

```
 Rec1
  Field1
  Field2
  Field3
  Field4
```

Suppose your target document has the following structure:

```
 Seg (repeats)
  Seg01
  Seg02
```

Field1 and Field2 are to be mapped to the first occurrence of Seg (Seg01 and Seg02), and Field3 and Field4 to the second occurrence of Seg (again, Seg01 and Seg02). To accomplish this, you would:

*   Map Field1 to Seg01, Field2 to Seg02.
*   Insert a CloseOccurrence(Seg) command to close the first occurrence of Seg.
*   Map Field3 to Seg01, Field4 to Seg02, which creates a second occurrence of Seg.

Error

Use the Error command to issue an error condition. This command allows you to establish your own errors for a translation. Typically, the error is issued from within an If conditional block.

The Error command has the following format:

```
Error(real level, real code, char text)
```

Where:

level   Indicates the severity of the error. It should be a value 0, 1, or 2.

code   Is the unique error code that should be associated with the error. This can be any value from 5000 to 5999.

text   Is the string value, which will be included in an error message issued by WebSphere Data Interchange when this command is executed.

The message is issued as a TR0026 error message. Within this message, *text* and *code* will be included.

The *level* that is assigned affects the extended return code from the translator, and thus the JCL condition code in the WebSphere Data Interchange utility. If *level* exceeds the acceptable error level specified in the document usage, the translation will not be successful.

FAError

Use the FAError command to set a functional acknowledgement code. You can only use this command in validation maps.

Use this command at any point at which you want to identify an error condition. This command allows you to establish your own error codres for translation. Typically, the error is issued from within an If conditional block.

The FAError command has the following format:

```
FAError(real level, real code, char facode, char text)
```

Where:

level   Indicates the severity of the error. It should be a value of 0, 1, or 2

code   Is the unique error code that should be associated with the error. This can be any expression that evaluates to an integer with a value from 5000 to 5999.

facode   Is an acknowledgement code that should be associated with this error. This value is placed in the functional acknowledgement if a functional acknowledgement is being generated. This can be any

expression that evaluates to a string. You must ensure that the string is valid for the specific syntax type.

text   Is any expression that evaluates to a string. It is included in an error message issued by WebSphere Data Interchange when this command is executed.

The message is issued as a TR0026 error message. Within this message, *text* and *code* are included.

The *level* that is assigned affects the extended return code from the translator, and thus the JCL condition code in the WebSphere Data Interchange utility. If *level* exceeds the acceptable error level specified in the document usage, the translation will not be successful.

ForEach

Use the ForEach command to execute mapping commands for each occurrence of the specified source node. Each occurrence of the source results in a new occurrence of the current target element.

You can include multiple ForEach command blocks within a single target element. The target element can be either a repeating node or a non-repeating node, and can be either a simple or a compound node.

If the target node is not repeating, you might need to use the Qualify command or conditional logic (If/Then/Else) so that only one occurrence of the source value is written to the target. If the target is not repeating and multiple values are written to it, later values will overwrite the earlier values.

You can only use the ForEach command in a target-based map.

The ForEach command has the following format:

```
ForEach(sourcepath)
```

Where:

sourcepath

Identifies an element in the source document definition. Each occurrence of the element will result in the creation of the target element which this command is under.

Establishes a current source position for the commands within the ForEach block.

The mapping commands within each ForEach command are executed for each occurrence of the specified source node (contrast this with the Qualify command, in which the mapping commands are executed only if a specific expression evaluates to True). See "Example 1:" on page 311.

Forward and backward references to source elements outside the domain are allowed with the same limitations that they have in source-based mappings. That is, you can refer to other siblings, cousins, and so on, but cannot refer to other elements within other occurrences of the *sourcepath*. Also, any nested ForEach elements must be within the current domain. This is illustrated in "Example 2:" on page 311.

A single ForEach can apply to multiple subdomains. "Example 3:" provides an example of this.

Example 1:
A map might be represented something like:

```
N1 Loop (element node)
 ForEach (\Order\Header\Sender\\)  (command node)
   N1  (element node, might contain commands)
   N2  (element node, might contain commands)
   ...
 ForEach (\Order\Header\Receiver\\) (command node)
   N1  (element node, might contain commands)
   N2  (element node, might contain commands)
   ...
```

In this example, there are two ForEach commands within the N1 Loop. Each occurrence of source elements Sender or Receiver will result in a new occurrence of a target N1 Loop. When a source Sender element is being processed, the mapping commands within the first ForEach block are executed to create the target N1 Loop occurrence. When a source Receiver element is processed, the mapping commands within the second block are executed

Example 2:
A source document might have the following structure (where "*" indicates repeating element):

```
Order
  Header
   PONumber
   Date
   NameAddress*
    Type
    Name
    ...
  DetailLoop*
   LineItemNumber
   SubDetailLoop*
    Description
    ...
  Trailer
   Total
```

If your map has a ForEach(\Order\DetailLoop\\) command, the domain for this block is the current DetailLoop occurrence. Commands in this block can refer to nonrepeating elements outside of the domain, such as \Order\Header\PONumber\\ and \Order\Trailer\Total\\. The commands can also refer to repeating elements outside the domain, such as \Order\Header\NameAddress\\. However, with forward and backward references to repeating elements it is more difficult to predict which occurrence will be used (it depends on what commands have been processed previously). If another ForEach command is nested within the ForEach(\Order\DetailLoop\\), it must refer to an element within \Order\DetailLoop\\, such as \Order\DetailLoop\SubDetailLoop\\. The nested ForEach command cannot refer to an element outside the current domain, such as \Order\Header\NameAddress\\.

Example 3:
For example, a source document has the following structure:

```
Order
  DetailLoop
   SubDetailLoop
```

```
     SubDetailLoop
  DetailLoop
    SubDetailLoop
    SubDetailLoop
```

If the the current domain is the root of the document (Order), and you specify ForEach(\Order\DetailLoop\SubDetailLoop\\) without the intermediate ForEach (\Order\DetailLoop\\), all four occurrences of SubDetailLoop (across both DetailLoop occurrences) are processed. If only the SubDetailLoop occurrences within a current DetailLoop are to be used, a ForEach(\Order\DetailLoop\\) command should be specified, then a ForEach(\Order\DetailLoop\SubDetailLoop\\) command should be specified within it.

HLLevel

Use the HLLevel command to create an HL loop within an EDI source or target based map.

The HLLevel command has the following format:

```
HLLevel( char levelcode)
```

Where:

levelcode   Indicates the level code for this level. You must select this value from the drop-down list of level codes that are valid for the code list for HL03 (element 735). This value identifies the context of a series of segments following the current HL up to the next occurrence of an HL, or the end of the loop.

If you are working with a source-based map:

• Drag a structure or record from the target window and drop it on to the HL loop in the source window. A MapTo command is created under the loop.

• To create the underlying HL nesting levels or children for this HL level, right-click the HL loop and select AddChild, or select AddPeer to create a sibling.

If you are working with a target-based map:

• Right-click the HL loop again and select the ForEach command

• Drag and drop the source path to the command wizard window.

• To create the underlying HL nesting levels or children for this HL Level, right-click the HL loop and select AddChild, or select AddPeer to create a sibling.

For more information about HL loops, see Appendix D, "Hierarchical loops" on page 377.

MapCall

Use the MapCall command 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 source element (for target-based maps).

When this command is encountered, a new copy of the translator is loaded. It receives the data from the current 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). The copied translator shares global variables with the parent translator.

When the copied translator has completed translation, it terminates, and control is returned to the parent translator. The parent translator checks if the imbedded translation was successful, and if so, resumes its own translation processing.

The output depends on the specified targetPath and the target document definition in the called map.

The MapCall command has the following format:

```
MapCall(sourcepath, char mapname, targetpath)
```

Where:

sourcepath   Identifies a source element that is to be used as the root of the source tree for the imbedded map. If the *sourcepath* element does not exist in the input message, the MapCall command is not executed.
- For source-based maps, this value must be the keyword *This*, which indicates the current source element.
- For target-based maps, the *sourcepath* element must be within the current domain.

mapname   Is a character string that specifies the map name. You can specify a literal value or a more complex expression that evaluates to a character string.

targetpath   Identifies the target element where the output of the imbedded map will go.
- If this is a simple binary element, such as a BIN02, the output from the called map is serialized and placed into this element. Use of compound elements and other data types for the targetPath on the MapCall command is not supported in WebSphere Data Interchange Version 3 Release 2.
- If this is the keyword *targetRoot*, the following logic is applied:
  - If the target document on the submap is the same as the target document on the primary map, the submap output is inserted into the same message as the primary map output. Children of the root in the submap output become children of the root in the primary map output.
  - If the target document on the submap is different than the target on the primary map, a separate message output is created.

MapChain

Use the MapChain command to indicate that the document needs to be translated by another map after the current translation has completed. The subsequent translation only occurs if the current translation is successful.

The MapChain command has the following format:

```
MapChain(char mapname)
```

Each MapChain command encountered during translation of a map indicates that the document will be translated an additional time using the newly specified map. This process stops if an error is encountered in any one translation. The output from a translation is independent from the other translations.

MapFrom

You can use the MapFrom command in the following ways:

- To move data from a repeating simple element in the target document definition to a corresponding repeating simple element in the source document definition.

  When you use the MapFrom command on a repeating target node, multi-occurrence mapping is required, and the MapFrom command must be included within a ForEach command.

- To move data from a nonrepeating simple element or variable in the target document definition to a corresponding simple element in the source document definition.

You cannot use the MapFrom command with compound elements.

Use the MapFrom command for target-based maps only: you must use the MapTo command for source-based maps. If the map is source-based and this command is used, the compiler command processor issues an error.

The MapFrom command has the following format:

```
MapFrom(expression)
```

Where:

expression   Is the expression whose value will be placed into the simple element. This expression can be any data type, as long as it is compatible with the target element that it is being mapped to.

The expression is evaluated, just as it would be for source-based transformation. The expression can be a simple source element, a variable, or a complex expression. If the expression does not evaluate to an empty string, the expression result is saved in the current target node. This is functionally equivalent to:

```
If (not IsEmpty(TrimRight(expr)))
   This = expr
```

MapSwitch

Use the MapSwitch command to indicate that the document needs to be translated by another map instead of the current map. Any translation performed by the current map is terminated and the document is translated by the map specified in the MapSwitch command.

MapSwitch has the following format:

```
MapSwitch(char mapname)
```

Use this command when data in the document must be inspected before it can be determined which map to use. It 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 mapping 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 the correct map.

MapTo

You can use the MapTo command in several different ways:

- To associate a repeating compound element in the source document definition to a corresponding repeating compound element in the target document definition.
- To move data from a repeating simple element in the source document definition to a corresponding repeating simple element in the target document definition.
- To move data from a nonrepeating simple element or variable in the source document definition to a corresponding simple element in the target document definition.

A MapTo command is generated when you drag a source element to a target element or a target element to a source element.

Use the MapTo command for source-based maps only: you must use the MapFrom command for simple assigments for target-based maps, or the ForEach command for multi-occurrence mapping with target-based maps. If the map is target-based and the MapTo command is used, the compiler command processor issues an error.

The MapTo command has the following format:

```
MapTo(targetpath [, expression])
```

Where:

targetpath   Identifies the path in the target document definition that is being mapped.

expression   Is the expression whose value will be placed into the simple element identified by *targetpath*. This expression can be any data type that is compatible with the target element that it is being mapped to.

When placed on a repeating or nonrepeating compound element and only *targetpath* is specified, the compound element in the source document definition is associated with the compound element identified by *targetpath* in the target document definition. Each occurrence of the source compound element results in the creation of a corresponding target compound element.

When placed on a repeating simple element and only *targetpath* is specified, the simple element in the source document definition is associated with the simple element identified by *targetpath* in the target document definition. Each occurrence of the source simple element results in the creation of a corresponding target simple element and data being moved from the source simple element to the target simple element. This is equivalent to the assignment statement:

```
targetpath = (current source element)
```

When placed on a nonrepeating simple element and only *targetpath* is specified, the simple element in the source document definition is associated with the simple element identified by *targetpath* in the target document definition. Encountering the source simple element results in the creation of a corresponding target simple element and data being moved from the source simple element to the target simple element. This is equivalent to the assignment statement:

```
targetpath = (current source element)
```

When *targetpath* is specified with *expression*, the result of *expression* will be moved to the simple element identified by *targetpath* in the target document definition. This is equivalent to the assignment statement:

```
targetpath = expression
```

Qualify and Default

The Qualify command is used on repeating compound or repeating simple elements in the source document definition. It is used to indicate that a specific iteration or iterations of the elements are to be handled differently than other iterations of the element. For instance, you might want to say that the first iteration of a loop is handled differently than all other iterations of the loop. The Qualify command uses the following format:

```
Qualify( bool boolExpr)
```

Where:

boolExpr Is the boolean value or expression that, when True, indicates the mapping commands within the qualification will be executed by the translator.

boolExpr can indicate if this is a particular occurrence of a repeating compound or simple element. Values of variables or simple elements in the source document definition can also be checked to determine when to execute the qualification. For instance, you can check to see if a specific simple element has a value of "ABC". If it does, the mapping commands within the qualification will be executed by the translator.

Example 1:

```
Qualify (StrComp(\Table 1\310 O N1 Loop\310 O N1\1 M 98\\, "ZZ") = 0)
```

In this example, the value contained in simple element "\Table 1\310 O N1 Loop\310 O N1\1 M 98\\" is compared to the literal "ZZ". If the function returns 0 (values are equal), the mapping commands within the qualification are executed.

Example 2:

```
Qualify(Occurrence() = 2 OR Occurrence() = 3)
```

In this example, the mapping commands within the qualification will be executed if it is the second or third occurrence of the simple or compound element.

Example 3:

```
Qualify(Occurrence() =
 2 AND StrComp(\Table 1\310 O N1 Loop\310 O N1\1 M 98\\, "ZZ") = 0)
```

In this example, the mapping commands within the qualification are executed if it is the second occurrence of the current compound source element and simple element "\Table 1\310 O N1 Loop\310 O N1\1 M 98\\" is "ZZ".

The Occurrence and StrComp functions are described in more detail in "Functions" on page 317. Any boolean expression or function can be used as the expression on the **Qualify** command.

The Default command can be used to specify the commands that should be executed if none of the Qualify expressions evaluate to **True**.

SetProperty

The SetProperty command is used to set a special processing property of the target message. Various special properties are defined to control things such as the EDI envelope fields or the XML prolog. The SetProperty command uses the following format:

```
SetProperty(char propertyName, char propertyValue)
```

Where:

propertyName   Is the property name to be set in the target message.

propertyValue   Is the value that it should be set to.

See "Message properties" on page 333 for a list of the property names that can be specified.

Functions

All functions take zero or more arguments as input and return a value. The data type of the return value, as well as the number and data types of the arguments varies from one function to the next. Appropriate type conversions are done implicitly if needed (and possible), just as they are done for the expressions.

Some functions have optional parameters. If the optional parameters are omitted from the function call, a default value is is used for that argument.

Most functions can take an expression as an argument, as long as the result of the expression is (or can be converted to) the correct data type. For example, the Char function converts a value to a character string. The command:

```
Var1 = Char ( 1 + 2 )
```

Is equivalent to :

```
Var1 = Char ( 3 )
```

The only time you cannot use an expression as an argument is when the argument is identified as a source or target path. For example, you cannot pass an expression to the Created function.

The input arguments for a function are never modified by the function. The return value and argument list for each of the functions is described below.

The function names are not case sensitive. For example, the function name Char is the same as CHAR.

Char

The Char function returns a character representation of a numeric value. The Char function uses the following format:

```
char Char(real value)
```

Where:

value   Is the value to be converted to a character value. It can be of data type real or int. (Data type character is allowed, but does not cause any conversion.)

Results:
The character representation of *value.*

This function is not normally needed, since the conversion is generally done implicitly. However, it can be used to force the conversion to a character value, or to clarify when the conversion is done.

Concat

The Concat function concatenates one character string to another. The Concat function uses the following format:

```
char Concat(char value1, char value2)
```

Where:

value1   Is the value that is appended to.

value2   Is the value that is appended to the end of *value1*.

Results:
The concatenated string.

Example:
```
chVar = Concat("abc", "def")
```

Would set chVar to "abcdef".

Created

The Created function is used to determine if the specified path was created in the target data. The Created function uses the following format:

```
boolean Created(targetPath)
```

Where:

targetPath   Identifies the path in the target document that is being checked.

Results:
True   When the specified path has been created in the target data

False   When it has not been created in the target data.

This function can only be used with paths in the target document definition. If the element is within a repeating compound element (such as a repeating loop or segment), then only the current (or most recent) iteration is searched for the *targetPath*.

This function can only be used with paths that can occur in the target document definition.

Example:
```
If (!Created(\T 2\L 2300\S CLM 130\C C023 5\\))
 mapping command
EndIf
```

In this example, if path "\T 2\L 2300\S CLM 130\C C023 5\\" has not been created, *mapping command* is executed by the translator.

Date

The Date function returns the system date as a character string in the format `yyyymmdd`. The Date function uses the following format:

```
char Date()
```

Results:
The current system date in the format `yyyymmdd`.

DateCnv

The DateCnv function is used to assist in converting one date format to another. The DateCnv function uses the following format:

```
char DateCnv(char sourceDate, char frommask, char tomask)
```

Where:

sourceDate   Is the string that contains the source date.

frommask   Is the mask that identifies the format of the date in *sourceDate*.

tomask   Is the mask that identifies the format of the date desired in the result string.

Results:

The converted date as a character string.

The mask format for the DateCnv function is the same as the mask format used in send and receive maps. "Date conversion special operators" on page 357 for information about the values that can appear in the from and to mask.

Exit

The any-to-any mapping function is called **Exit**. **Exit** can be used within transformation maps to invoke a field exit and will return a string value. Results that are returned can be used to update a variable or target a simple element in an assignment statement. The **Exit** function uses the following format:

```
 char Exit(char exitname[, char parameter[, char parameter[, ...]]])
```

Where:

result   A string returned by the user exit

exitname   A string expression that results in the **User Exits** profile member name that contains the function information to be executed.

parameter   Evaluates to a string that will be passed to the field exit. Parameter can not evaluate to a simple element in the target data. Up to 4 parameters can be passed to the **Exit** function.

You must define field exits in the **User Exits** profile (See *WebSphere Data Interchange User's Guide*.) If the exit is not properly defined, an error is issued. Zero to four parameters can be passed to the exit and it is the job of the field exit to determine how many parameters to expect.

Find

The Find function is used to determine if a string is contained within another string. The Find function uses the following format:

```
real Find(char toSearch, char searchVal, real startPos)
```

Where:

toSearch   Is the string to be searched.

searchVal   Is the string that is to be searched for within *toSearch*.

startPos   Is the starting position within *toSearch* where the search will be started.

Results:

The position where *string* occurs in the simple element or variable.

Zero is returned when:

* *searchVal* is not found within toSearch
* The *toSearch* value has less than *startPos* characters
* *toSearch* is empty

startPos is one-based, so a *startPos* of one will start the search from the beginning of the string, a position of two begins the search at the second character, and so on.

Examples

Assuming *var1* contains the string "ABCDEFG",

```
Find(var1, "CDE", 1) will return 3
```

```
Find(var1, "CDE", 3) will return 3
```

```
Find(var1, "CDE", 4) will return 0
```

```
Find(var1, "CDE", 0) will cause an error to be issued
```

Found

Found is used to determine if a specified path exists in the source data. The Found function uses the following format:

```
boolean Found(sourcePath)
```

Where:

sourcePath   Identifies the path that is being checked.

Results:

True   When the specified path is found in the source data

False   When it is not in the source data.

This function can only be used with paths in the source document definition. If the element is within a repeating compound element (such as a repeating loop or segment), then only the current iteration is searched for the *sourcePath*.

Example:

```
If (!Found(\T 2\L 2300\S CLM 130\C C023 5\\))
 mapping command
EndIf
```

In this example, if path "\T 2\L 2300\S CLM 130\C C023 5\\" is not found, *mapping command* is executed by the translator.

GetProperty

The GetProperty function is used to get a property of the source message. This can be used to retrieve information such as EDI envelope header elements. The GetProperty function uses the following format:

```
char GetProperty(char propertyName)
```

Where:

propertyName

Is the name of the property you want to retrieve. See "Message properties" on page 333 for a list of properties that you can retrieve.

Results:
The value associated with the specified message property.

If the propertyName is not set in the source message, an empty string is returned.

Example:
```
var1 = GetProperty("ISA04")
```

Will set var1 to the value that was in the ISA04 element in the X12 ISA segment.

HexEncode

The HexEncode function is used to encode binary data to a character format. The HexEncode function uses the following format:
```
char HexEncode(binary binValue)
```

Where:

binValue   Is the binary value to be encoded.

Results:
An encoded character string that represents the binary data.

Each byte of the binary value will be encoded as two characters in the resulting string. For example, if the input is a 4 byte binary value: 0x01020A0B, the result would be the 8-character string: "01020A0B".

HexDecode

The HexDecode function is used to decode a character string that contains encoded binary data. The HexDecode function uses the following format:
```
binary HexDecode(char encodedStr)
```

Where:

encodedStr   Is an encoded string.

Results:
A binary value derived from the encoded character string.

Each byte of the binary value will be derived from two characters in the encoded string. For example, if the input is the 8-character string: "01020A0B", the result would be a 4 byte binary value: 0x01020A0B.

IsEmpty

The IsEmpty function is used to determine if a character string contains any data. The IsEmpty function uses the following format:
```
boolean IsEmpty(char value)
```

Where:

value   Is the value to check.

Results:

True   When *value* is empty (set to an empty string: "").

False   When *value* contains data.

If the value is a source element, this will also return **True** if the element is not found. The expression:

```
IsEmpty(value)
```

is equivalent to the following string comparison expression:

value   Is a string, usually a variable or a simple element in the source document definition.

Results:

The length of the character string value.

Example:

```
If (Length(var1) < 5)
 mapping command
EndIf
```

In this example, if *var1* has a length that is less than 5, *mapping command* is executed by the translator.

Lower

The Lower function is used to convert a string to lowercase. The Lower function uses the following format:

```
char Lower(char value)
```

Where:

value   Is a string, usually a variable or a simple element in the source document definition.

Results:

A copy of *value*, with all characters converted to lowercase.

Example:

```
var1 = Lower("ABC")
```

In this example, *var1* will be set to "abc".

Number

The Number function converts a character value to a real value. The Number function uses the following format:

```
real Number(char value)
```

Where:

value   Is the character value to be converted to a real value. (Data type real or int are also allowed, but do not cause any conversion.)

Results:

The (data type) real representation of *value*.

This function is not normally needed, since the conversion is generally done implicitly. However, it can be used to force the conversion to a numeric value, or to clarify when the conversion is done.

NumFormat

The NumFormat function formats a real number or integer as a character string using a specified number of decimal positions. Unused digits are either truncated or rounded. The NumFormat function uses the following format:

```
char NumFormat(real value, real decimals[, char flag])
```

Where:

value   Is the number that will be formatted as a character string.

decimals   Is the number of decimal positions that should be included in the returned value.

flag   Indicates whether unused digits are rounded or truncated. Specify "ROUND" to round unused digits or "TRUNCATE" to truncate unused digits. These can be shortened to "R" or "T". If no flag is specified, the value defaults to "ROUND".

Results:

The character representation of *value*, containing the specified number of decimal places.

Example 1:

```
var1 = NumFormat(1234.567, 2, "T")
```

This example will set *var1* to "1234.56".

Example 2:

```
var1 = NumFormat(1234.567, 2, "R")
```

This example will set *var1* to "1234.57".

Occurrence

Use the Occurrence function to obtain the current occurrence number of:

- The current repeating compound element or simple element
- A specific repeating compound element or simple element

Occurrence can be shortened to Occur. The Occurrence function has the following format:

```
real Occurrence([sourcepath])
```

Where:

sourcepath   Is the path that refers to a compound or simple element in the source document definition. If no *sourcepath* is specified, the current compound or simple element in the source document is assumed.

If this function is used in a target-based map, you must include the source element as a parameter. The source element must be the current source domain, or an ancestor (for example, parent or grandparent) of the current source domain. If you omit the *sourcepath*, the compiler command processor issues an error.

For source-based maps, the source element parameter continues to be optional (if omitted, the current source element is assumed).

Results:

The current occurrence number of the specified element.

When Occurrence is specified with no parameters, it will return the occurrence number of the current compound or simple element. When *sourcePath* is specified, it will return the current occurrence number of the specified repeating compound element or simple element. The specified element must be the current element, a parent of the current element, or one of the other elements on the path to the current element (such as the grandparent of the current element).

Example 1:

```
If (Occurrence (\Table 2\10 M PO1 Loop\\) = 2)
 mapping command
EndIf
```

In this example, if this is the second occurrence of the PO1 loop, then *mapping command* will be executed by the translator.

Example 2:

```
If (Occurrence() != 1)
 mapping command
EndIf
```

In this example, if this is not the first occurrence of the current source element, *mapping command* is executed by the translator.

Overlay

The Overlay function is used to overlay a portion of a string with data from another string. The Overlay function uses the following format:

```
char Overlay(char string1, char string2, real position [,
real length])
```

Where:

string1   Evaluates to a string that will be overlaid with data from string2.

string2   Evaluates to a string that will overlay data in string1.

position   Indicates the position where the overlay will begin in *string1*.

length   Is an optional parameter that indicates the number of characters to be overlaid in *string1*. If omitted or less than 0, all of the characters following *position* will be overlaid until all characters from *string2* have been used.

Results:

The resulting string *position* is one-based, so a position of one will begin overlaying characters at the beginning of *string1*, a position of two begins overlaying at the second character, etc. If the length of *string1* has less than *position* characters, then blanks will be used to fill the positions between the end of *string1* and the starting position.

Example:

```
var1 = overlay("ABCDEFG", "WXYZ", 3, 3)
```

After this code has been executed, *var1* will equal "ABWXYFG".

Right

The Right function is used to obtain the last few characters from a string. The Right function uses the following format:

```
char Right(char stringIn, real length)
```

Where:

stringIn   Is the input string that characters are copied from.

length   Indicates the number of characters to be copied.

Results:

A character string containing the rightmost characters of *stringIn*.

Only available characters from *stringIn* will be copied. Therefore, the returned string will contain *length* characters unless *stringIn* has less than *length* characters. See Example 2 below.

Example 1:

```
var1 = Right("ABCDEFG",3)
```

After this code has been executed, *var1* will equal "EFG".

Example 2:

```
var1 = Right("ABCD",6)
```

After this code has been executed, *var1* will equal "ABCD" because there are not enough characters in the *stringIn* argument to fulfill the request.

Round

The Round function rounds the input value to the specified number of decimal places. The Round function uses the following format:

```
real Round(real value, real decimals)
```

Where:

value   Is the value that will be rounded.

decimals   Is the number of decimals to round to.

Results:

A real number rounded to the specified number of decimal places.

For instance, `Round(4321.556, 2)` will return a result of 4321.560000. `Round(4321.556, 0)` will return a result of 4322.000000.

Note: This function returns a numeric (real) value. If the return value will be assigned to a character string and trailing zeros are to be removed, the NumFormat function should be used.

StrComp

The StrComp function is used to compare two character strings. The StrComp function uses the following format:

```
real StrComp(char string1, char string2)
```

Where:

string1   Is the first string to be compared.

string2   Is the second string to be compared.

Results:

-1   If *string1* < string2
0   If the strings are equal
1   If *string1* > string2

Example:
```
If (StrComp(var1, "ABC") = 0)
 mapping command
EndIf
```

In this example, if *var1* is "ABC", *mapping command* is executed by the translator.

StrCompI

The StrCompI function is used to compare two character strings without regard to case. The StrCompI function uses the following format:
```
real StrCompI(char string1, char string2)
```

Where:

string1   Is the first string to be compared.

string2   Is the second string to be compared.

Results:
-1   If *string1 < string2* (case insensitive)
0   If the strings are equal (case insensitive)
1   If *string1 > string2* (case insensitive)

Example:
```
If (StrCompI(var1, "ABC") = 0)
 mapping command
EndIf
```

In this example, if *var1* is "ABC", "abc", "Abc", *mapping command* is executed by the translator.

StrCompN

The StrCompN function is used to compare the first *length* characters of two character strings. The StrCompN function uses the following format:
```
real StrCompN(char string1, char string2, real length)
```

Where:

string1   Is the first string to be compared.

string2   Is the second string to be compared.

length   Is the number of characters to compare.

Results:
-1   If the first *length* characters of *string1* < the first *length* characters of string2
0   If the first *length* characters of the two strings are equal
1   If the first *length* characters of *string1* > the first *length* characters of string2

Example:
```
If (StrCompN(var1, "ABC", 3) = 0)
    mapping command
EndIf
```

In this example, if the first 3 characters of *var1* are "ABC", then *mapping command* will be executed by the translator.

StrCompNI

The StrCompNI function is used to compare the first *length* characters of two character strings without regard to case. The StrCompNI function uses the following format:

```
real StrCompNI(char string1, char string2, real length)
```

Where:

string1 Is the first string to be compared.

string2 Is the second string to be compared.

length Is the number of characters to compare.

Results:

-1 If the first *length* characters of *string1* < the first *length* characters of *string2* (case insensitive).

0 If the first *length* characters of the two strings are equal (case insensitive).

1 If the first *length* characters of *string1* > the first *length* characters of *string2* (case insensitive).

Example:

```
If (StrCompNI(var1, "ABC", 3) = 0)
   mapping command
EndIf
```

In this example, if the first 3 characters of *var1* are "ABC", "abc", "Abc", and so on, *mapping command* is executed by the translator.

SubString

The SubString function is used to obtain a portion of a string. The SubString function uses the following format:

```
char SubString(char string, real position [, real length])
```

Where:

string Is the string value.

position Indicates the starting position of the substring within the *string*.

length Is an optional parameter that indicates the number of characters to be copied into the substring. If this argument is omitted or the length is less than 0, all remaining characters to the end of the string are copied.

Results:

The resulting character string returned by the function.

position is one-based, so a *position* of one will start copying characters from the beginning of *string*, a position of two begins copying data from the second character, etc.

Only available characters from the *string* value will be copied. Therefore, the substring will contain *length* characters unless *string* has less than *position* plus *length* minus one characters. See example 2 below. If the *string* value has less than *position* characters, an empty string will be returned.

Example 1:
```
var1 = substring("ABCDEFG",3,3)
```

After this code has been executed, *var1* will equal "CDE".

Example 2:
```
var1 = substring("ABCD",3,3)
```

After this code has been executed, *var1* will equal "CD" because there are not enough characters in the *string* argument to fulfill the request.

Time

The Time function returns the system time as a character string in the format `hhmmss`. The Time function uses the following format:
```
char Time()
```

Results:
The system time as a character string in the format `hhmmss`.

Translate

The Translate function attempts to locate a specified string in a translation table. If the string is found in the table, the corresponding value in the table is returned. The Translate function uses the following format:
```
char Translate(char table, char direction, char valueIn [,
boolean error[, char default]])
```

Where:

table   Is the string that identifies the translation table.

direction   Is either "SOURCE" or "TARGET". This can be shortened to either "S" or "T".

valueIn   Is the string value that will be looked up in the translation table.

error   Indicates whether a not found condition will result in a warning message. If **True**, a warning message will be issued if the value is not found in the table. If **False**, no warning message will be issued. The default is **True**.

default   Is a string that provides a default value that is returned when *valueIn* is not found in the translation table.

Results:
The resulting character string.

table must be a valid *Forward Translation* table. If *table* does not exist, a warning will be issued and it will be treated as a not found condition.

A translation table contains two values for each entry in the table, one called the *Local Value* and the other called the *Standards or Trading Partner Value*. When *direction* is specified as "SOURCE", an attempt is made to find *valueIn* in the *Local Value* column in the table. If it is found, the value from the corresponding *Standards or Trading Partner Value* column is returned by the function. When *direction* is specified as "TARGET", an attempt is made to find *valueIn* in the *Standards or Trading Partner Value* column in the table. If it is found, the value from the corresponding *Local Value* column is returned by the function.

In a *Forward Translation* translation table, the *Local Value* column must be unique. The *Standards or Trading Partner Value* column does not have to be unique. A *Forward Translation* translation table used in the Translate command will always produce predictable results when SOURCE is specified as the direction. The results are also always predictable when TARGET is specified as the direction *if* the values in the *Standards or Trading Partner Value* column are unique. If the values are not unique in the *Standards or Trading Partner Value*, the results might not be predictable when TARGET is specified as the direction.

If *valueIn* is not found in the translation table, a warning message will be issued if *error* is **True**. Also, if *valueIn* is not found in the translation table, the *default* value will be returned if specified. If a *default* value is not specified, an empty string ("") is returned.

Example:

```
var1 = Translate("MYTABLE", "S", stringVar, False, "ABC")
```

This command will get the data contained in variable *stringVar* and attempt to locate that value in the *Local Value* column of translation table "MYTABLE". If the value is not found in the translation table, a warning message is not issued and *var1* will be set to "ABC". If the value is found in the translation table, *var1* will be set to the value in the corresponding *Standards or Trading Partner Value* column.

TrimLeft

The TrimLeft function is used to remove unwanted leading characters from a string. The TrimLeft function uses the following format:

```
char TrimLeft(char value, [char trimList])
```

Where:

value   Is the string value that is to be trimmed.

trimList   Is an optional string value that includes all the characters that are to be removed. If this is not specified, all leading whitespace characters are removed.

Results:
The resulting character string.

All characters contained in *trimList* will be removed from the beginning of the *value* string. The operation ends when the first character not contained in *trimList* is encountered. If *trimList* is omitted, whitespace is removed from the beginning of the *value* string. Whitespace includes blanks, carriage returns, line feeds and tab characters.

Example 1:

```
var1 = TrimLeft(" ABCD")
```

In this example, *var1* will equal "ABCD" after the function has been executed.

Example 2:

```
var1 = TrimLeft("ZZYDABCD", "DYZ")
```

In this example, *var1* will equal "ABCD" after the function has been executed. All characters "D", "Y" and "Z" will be removed from the beginning of *value* until an unspecified character is encountered.

TrimRight

The TrimRight function is used to remove unwanted trailing characters from a string. The TrimRight function uses the following format:

```
char TrimRight(char value, [char trimList])
```

Where:

value   Is a string value that is to be trimmed.

trimList   Is an optional string value that includes all the characters that are to be removed. If this is not specified, all trailing whitespace characters are removed.

Results:
The resulting character string.

All characters contained in *trimList* will be removed from the end of the *value* string. The operation ends when the first character not contained in *trimList* is encountered. If *trimList* is omitted, whitespace is removed from the end of the *value* string. Whitespace includes blanks, carriage returns, line feeds and tab characters.

Example 1:
```
var1 = TrimRight("ABCD    ")
```

In this example, *var1* will equal "ABCD" after the function has been executed.

Example 2:
```
var1 = TrimRight("ABCDZZY", "AYZ")
```

In this example, *var1* will equal "ABCD" after the function has been executed. All characters "A", "Y" and "Z" will be removed from the end of the string until an unspecified character is encountered.

Truncate

The Truncate function is used to truncate a number to the specified number of decimal places. The Truncate function uses the following format:

```
real Truncate(real value, real decimals)
```

Where:

value   Is the value that will be truncated.

decimals   Is the number of decimals to truncate to.

Results:
A real number truncated to the specified number of decimal places.

The real number *value* is truncated to *decimals* decimal places - no rounding occurs. For instance, **Truncate**(4321.556, 2) will return a result of 4321.550000. **Truncate**(4321.556, 0) will return a result of 4321.000000.

Note: This function returns a numeric (real) value. If the return value will be assigned to a character string and trailing zeros are to be removed, the NumFormat function should be used.

Upper

The Upper function is used to change a string to upper case. The Upper function uses the following format:

```
char Upper(char value)
```

Where:

value Is a string, usually a variable or a simple element in the source document definition.

Results:
A copy of *value*, with all characters converted to uppercase.

Example:
```
var1 = Upper("abc")
```

In this example, *var1* will be set to "ABC".

Validate

The Validate function attempts to locate a specified string in a validation table. The Validate function uses the following format:

```
boolean Validate(char table, char value [, boolean error])
```

Where:

table Is a string that identifies the validation table.

value Is a string value that will be looked up in the validation table.

error Indicates whether a not found condition will result in a warning message. If **True**, a warning message will be issued if the value is not found in the table. If **False**, no warning message will be issued. The default is **True**.

Results:
True If the string resulting from *value* is found in the validation table.

False Is returned if the string is not found in the validation table.

table must be a valid *Code List* table. If *table* does not exist, a warning message will be issued and it will be treated as a not found condition.

The *value* string is used to search the validation table. If the string is found in the validation table, **True** is returned by the function. If the string is not found in the validation table, **False** is returned.

Example:
```
If (Validate("MYTABLE",\Table 1\20 M BEG\1 M 353\\))
 mapping command
Else
 Error(1, 5001, , "Invalid value specified")
EndIf
```

This code will get the data contained in simple element "\Table 1\20 M BEG\1 M 353\\" and attempt to locate that value in the validation table "MYTABLE". If the value is found in the validation table, the *mapping command* will be executed. If the value is not found in the validation table, an error is issued with error code 5001.

Message properties

This section lists the message properties that you can access using the SetProperty command and GetProperty function:
- "Target document properties"
- "EDI envelope standard generic properties" on page 334
- "EDI envelope standard specific properties" on page 335

Source document properties

The following properties are specific to source documents. They can be checked by using the GetProperty command.

Alphanum   Inbound alphanumeric data validation table from the rule

Charset   Inbound character data validation table from the rule.

GrpLvlFA   Group level functional acknowledgement only flag from the rule

Process   The process value from the trading partner profile

ValLevel   Inbound validation level from the rule

ValErrLevel   Inbound acceptable error level from the rule

ValMap   Inbound validation map name from the rule

Target document properties

The following properties are specific to target documents. They can be set using the SetProperty command.

Setting these properties does not effect the current map, but might affect processing of the target document in later steps

AckReq   Overrides the acknowledgement expected setting on the rule

Alphanum   Overrides the outbound alphanumeric data validation table from the rule.

Charset   Overrides the outbound character data validation table from the rule.

CtlNumFlag   Overrides the control numbers by transaction id flag from the rule.

DIDocId   Document id value. This appears in the PRTFILE message when the output is written to a file or queue.

DIProlog   Used to override the default XML prolog in the target document.

EdiDecNot   Overrides the EDI decimal notation character from the trading partner profile

EdiDeDlm   Overrides the EDI data element delimiter from the trading partner profile

EdiDeSep   Overrides the EDI repetition separator from the trading partner profile

EdiRlsChar   Overrides the EDI release character from the trading partner profile

EdiSeDlm   Overrides the EDI subelement delimiter from the trading partner profile

EdiSegDlm   Overrides the EDI segment delimiter from the trading partner profile

Message properties

EdiSegSep   Overrides the EDI segment id separator from the trading partner profile

EnvProfName Overrides the envelope profile name from the rule.

EnvType   Overrides the envelope type from the rule.

NetProfId   Overrides the network profile id from the trading partner profile

SegOutput   Overrides the segmented output flag from the trading partner profile

ValLevel   Overrides the outbound alphanumeric data validation table from the rule.

ValErrLevel Overrides the outbound acceptable error level from the rule.

ValMap   Overrides the outbound alphanumeric data validation table from the rule.

EDI envelope standard generic properties

The EDI envelope standard generic properties allow you to get or set information in the EDI envelope standard segments. They have generic names that can apply to any of the EDI standard types. For example, IchgSndrId is used for the interchange sender ID, regardless of the EDI standard type.

These properties are available when the source document is received within an EDI envelope standard. The properties can be obtained using the GetProperty function. If you request a property that is not present, an empty string is returned.

These values can also be set for the target EDI document using the SetProperty command. If set, they will override the values specified in the envelope profile.

Here is a list of the EDI envelope standard generic properties.

IchgCtlNum	Interchange control number
IchgSndrId	Interchange sender ID
IchgRcvrId	Interchange receiver ID
IchgSndrQl	Interchange sender qualifier
IchgRcvrQl	Interchange receiver qualifier
IchgDate	Interchange date
IchgTime	Interchange time
IchgPswd	Interchange password
IchgUsgInd	Interchange usage indicator
IchgAppRef	Interchange application reference
IchgVerRel	Interchange version/release
IchgGrpCnt	Number of groups in interchange
IchgCtlTotal	Control total from interchange trailer segment
IchgTrxCnt	Number of transactions in interchange
GrpCtlNum	Group control number
GrpFuncGrpId	Functional group ID
GrpAppSndrId	Group application sender ID

GrpAppRcvrId	Group application receiver ID
GrpDate	Group date
GrpTime	Group time
GrpPswd	Group password
GrpVer	Group version
GrpRel	Group release
GrpTrxCnt	Number of transactions in group
TrxCtlNum	Transaction control number
TrxCode	Transaction code
TrxVer	Transaction version
TrxRel	Transaction release
TrxSegCnt	Number of segments in the transaction
BegEnv	Begin Envelope
EndEnv	End Envelope
BegGrp	Begin Group
EndGrp	End Group

EDI envelope standard specific properties

The EDI envelope standard specific Properties also allow you to get or set information in the EDI envelope standard segments. These have names that are specific to a particular EDI standard, and specify a particular segment and element number. For example, ISA04 is used for the fourth element of the ISA segment in an X12 envelope.

All of these properties are used to obtain the value of common data contained in an EDI envelope standard segment. These properties are available when the source document is received within an EDI envelope standard. These properties can be obtained using the GetProperty function. If you request a property that is not present, an empty string is returned.

These values can also be set for the target EDI document using the SetProperty command. If set, they will override the values specified in the envelope profile.

Here is a list of the EDI envelope standard specific properties.

ISAnn   "nn" is 01 through 16. Use attribute ISA01 through ISA16 to obtain information from each element in the ISA segment contained in the envelope.

GSnn   "nn" is 01 through 08. Use attribute GS01 through GS08 to obtain information from each element in the GS segment contained in the envelope.

STnn   "nn" is 01 or 02. Use attribute ST01 and ST02 to obtain information from each element in the ST segment contained in the envelope.

SEnn   "nn" is 01 through 02. Use attribute SE01 and SE02 to obtain information from each element in the SE segment contained in the envelope.

Message properties

GEnn "nn" is 01 through 02. Use attribute GE01 and GE02 to obtain information from each element in the GE segment contained in the envelope.

IEAnn "nn" is 01 through 02. Use attribute IEA01 and IEA02 to obtain information from each element in the IEA segment contained in the envelope.

UNBnn "nn" is 01 through 18. Use attribute UNB01 through UNB18 to obtain information from each element in the UNB segment contained in the envelope.

UNGnn "nn" is 01 through 13. Use attribute UNG01 through UNG13 to obtain information from each element in the UNG segment contained in the envelope.

UNHnn "nn" is 01 through 09. Use attribute UNH01 through UNH09 to obtain information from each element in the UNH segment contained in the envelope.

UNTnn "nn" is 01 through 02. Use attribute UNT01 and UNT02 to obtain information from each element in the UNT segment contained in the envelope.

UNEnn "nn" is 01 through 02. Use attribute UNE01 and UNE02 to obtain information from each element in the UNE segment contained in the envelope.

UNZnn "nn" is 01 through 02. Use attribute UNZ01 and UNZ02 to obtain information from each element in the UNZ segment contained in the envelope.

BGnn "nn" is 01 through 07. Use attribute BG01 through BG07 to obtain information from each element in the BG segment contained in the envelope.

EGnn "nn" is 01 through 04. Use attribute EG01 through EG04 to obtain information from each element in the EG segment contained in the envelope.

Appendix C. Advanced send and receive mapping

This appendix describes how to map your application data to an EDI standard transaction set. This appendix assumes that you are familiar with your application data layout and have already defined your application data to WebSphere Data Interchange. It also assumes you are familiar with the EDI standard transaction set you are using.

Using accumulators

You can use accumulators to count occurrences of an event, such as counting the detail line items in a purchase order to provide a hash total. You can also use accumulators to total fields for control purposes, such as totaling the quantity field to cross check the number of items sent or received.

Accumulators can apply to individual transactions or to all transactions in a translation session.

Valid accumulators are:

T0--T9 Transaction accumulators that apply to one transaction. They are reset at the beginning of each transaction.

G0--G9
   Global accumulators that apply to an entire translation session. They are reset at the beginning of each translation session.

Basic accumulator actions are adding, incrementing, zeroing, and mapping. These actions can also be combined. Actions and combinations of actions possible are:

A   For sending, adds to the accumulator the EDI standard data just mapped. For receiving, adds to the accumulator the EDI standard data received. The data must be numeric.

I   Increments the accumulator by 1.

M   Maps the accumulator to the EDI standard data element (send) or the application field (receive).

Z   Zeroes the accumulator.

MI   Maps the accumulator, then increments it.

IM   Increments the accumulator, then maps it.

MZ   Maps the accumulator, then zeroes it.

AM   Adds to the accumulator, then maps it.

MA   Maps the accumulator, then adds to it.

For send transactions, accumulator actions will not be processed unless one of the following occurs:
- Data is generated for the EDI standard data element.
- The variable is mapped.

For receive transactions, accumulator actions will not be processed unless one of the following occurs:
- The data element associated with the accumulator is received.

337

Using accumulators

- The accumulator is mapped and at least the segment containing the data element is received.

To map both an accumulator and a received value for the same data element, map the data element to a field. Then use the Repeat button in the Mapping Data Element Editor to create another mapping occurrence of the data element. In the new mapping occurrence of the data element, map from the accumulator to a field.

Accumulators have the following limitations:
- Each accumulator holds a maximum of 31 digits.
- Each data element mapping can support up to 4 accumulators.
- You can use up to 10 transaction accumulators for a transaction.
- You can use up to 10 global accumulators for an entire translation session.

Using literals

For send transactions, literals let you supply data that is not in your application data. For receive transactions, literals let you supply data required by your application that is not received with the standard data. WebSphere Data Interchange offers a variety of options for using literals. This section provides details on each option available, including rules for use, keywords, and syntax.

Transactions are always processed starting with the first data element of the first segment and proceeding to the last data element of the last segment, as documented in the standard. This is true for both send and receive processing.

Using literals for send mapping

When mapping literals for sending data, you can map both an application field and a literal to the same data element. The literal value is used if one of the following conditions occur:
- The record containing the application data field was provided, but the application data field does not supply a value or contains all blanks.
- The application data field supplies a value that does not match a value in the validation or translation table specified in the mapping.
- The application data field supplies a value, but the conversion from the application data type to the EDI standard data type fails.

WebSphere Data Interchange attempts a conversion for any application data field defined as a numeric field. The conversion removes leading and trailing blanks, leading zeros before the decimal, trailing zeros after the decimal, and changes the decimal point if the application decimal notation is different from the EDI standard decimal notation. If the data types are numeric, the conversion fails if the data contained anything other than a number or a decimal point. The literal value is used if the conversion fails. See "Validation during mapping" on page 374 for more information.
- If a translation table is specified, the literal value is checked against this table. If no matching entry is found, the translator logs a warning message in the event log, then uses the literal value. For information about the event log, see Chapter 33, "Event logs" on page 285.

Segment creation for send mapping

When an EDI standard segment is mapped with a combination of literal values and application data, the application data determines if the segment is produced. When the values from the application data produce data for the segment, the EDI standard segment is created. When the values in the application data do not produce data for the segment, such as when the records are not found, the application fields contain all blanks or zeros, or the application field values are not found in the associated validation or translation tables, then the EDI standard segment is not created. An exception is when zeros are passed to a mandatory numeric data element, in which case the segment is created. If this is not desirable, then pass blanks in the application data instead, or use the logic discussed below to suppress the segment.

In some cases it may be desirable to suppress a segment altogether depending on input application data values. It is possible to suppress a segment using *IF* logic. When all contributing data element mappings for a segment contain conditional *IF* statements and all conditions are false, the segment is not produced. A contributing data element mapping is one that may produce output in the corresponding data element; such as specifying an application field name or a literal value, or specifying *USE* on a variable. Noncontributing data element mappings, such as *SAVE* or *SET*, do not have any effect on segment creation. Refer to Table 46 on page 341 for details on *IF, USE, SAVE,* and *SET* usage.

Note: If the segment being suppressed begins a loop of segments, the entire loop is suppressed.

Using literals for receive mapping

If a data element mapping provides a literal, the literal value is used if one of the following conditions occur:

- The EDI standard data includes the segment being mapped, but the data element within the segment does not supply a value.
- The EDI standard data element supplies a value that does not match a value in the validation or translation table specified in the mapping.
- The EDI standard data element supplies a value, but the conversion from the EDI standard data type to the application data type fails. WebSphere Data Interchange attempts a conversion for any EDI standard data field defined as a numeric field. If the data types are numeric and the EDI standard data field contains nonnumeric characters, the conversion would fail and the literal value would be used.
- The EDI standard data element supplies a value, but the &FORCE special literal is used to force the literal into the application field regardless of the EDI standard data element's contents.

Format of literal data

When using literals in send or receive mapping:

- For data types BN, B*n*, HX, H*n*, IT, I*n*, L*n*, PD, P*n*, ZD, and Z*n*, the translator converts the literal before placing it in the standard data (send) or the application data (receive). For sending, it converts the literal to character data. For receiving, it converts the literal to the application data type. For example, if the data type is binary, the translator converts the literal to binary, then moves it to the application field.

- Do not type a decimal point when it is implied. For example, if you want to use a default value of 9.99 for a field defined as data type P2 (packed number with two implied decimal positions), enter 999 as the value of the literal.

- Enter literal values for hexadecimal fields as hexadecimal strings. For example, if the application field is defined as a one-byte hexadecimal field and you want to use a default value of X'FF', enter FF as the value of the literal. For receiving, the translator converts each two bytes of literal value to a single byte of application data.

- You can specify a literal value of zero to move a value into the standard field. WebSphere Data Interchange generally removes leading zeros from an application field so that an application field containing nothing but blanks or zeros will not result in a value for the data element. A value of zero is treated the same as all other literal values when determining if a segment should be created. The &ZEROSIG special literal may also be used to indicate that zeros within the application field are significant.

In translation and validation tables, enter numeric values left-justified and formatted according to the application data format. For example, if the data format defines a field as R2, enter the value 7 as 7.00 or the value 7.1 as 7.10.

Accumulator literals

Accumulator literals allow access to the local and global accumulators. You can map an accumulator using the accumulator action, or you can map it using the literal associated with the accumulator. Using the literal allows a user exit to gain control. An accumulator has a data type of R. The literals are:

&Tn   (Where n can be 0 through 9) identifies the accumulators T0 through T9.

&Gn   (Where n can be 0 through 9) identifies the accumulators G0 through G9.

Conditional processing literals

Conditional processing lets you define how you want data processed, based on rules you establish. The following terms will be used in discussing conditional processing:

Named variable
A name used to represent data whose value can be changed while a program is running. You supply the name you want to use.

Expression
A sequence of instructions which can consist of named variables, literals, operators, and constants. When processed, this sequence of instructions provides a single value.

Constant
A value that does not change.

Operator
A symbol that represents an operation to be done. WebSphere Data Interchange uses arithmetic operators, Boolean operators, comparison operators, unary operators, relational operators, and special operators.

Operation
An action performed on one or more data items such as multiplying, comparing or moving.

Value A data value you provide. You can use any of the special literals that

provide a data value; for example, &DATE, &TIME, &E, &ICN. This includes &T0 through &T9 and &G0 through &G9 to get the values for global and local accumulators.

Default value

A default data value you provide. This value should not be enclosed in quotation (") marks. You can use any of the special literals that provide a data value; for example, &DATE, &TIME, &E, &ICN. This includes &T0 through &T9 and &G0 through &G9 to get the values for global and local accumulators.

Note: When using conditional processing on a data element used to qualify a loop or repeating segment, the first occurrence of the qualifying data element mapping must specify a literal or application data field.

Literal keywords

Table 46. Literal keywords

Keyword	S/R	Keyword description and syntax
&ACFIELD	S/R	**Syntax:** &ACFIELD Substitutes the application control value established by the mappings of the AC field from the data format or the concatenation of the fields specified during creation of the transaction mapping. **Note:** Literals specified with &LIT and variables specified with &VAR in the mapping are not available in the AC field until the end of the translation process and will not appear in the AC field data that is moved using the &ACFIELD keyword.
&ASSERTn	S/R	**Syntax:** &ASSERT*n*(*expression*) The action associated with this literal is only executed if the *expression* is false and the assertion level is not greater than *n*. The &ASSERT keyword can be combined with: • A mapping between an application data field and EDI standard data elements • The &SET, &SAVE, and &USE keywords • An error condition; for example, &ASSERT*n* followed by &ERR. The differences between &ASSERT and &IF are: • &ASSERT is a statement about the transaction that is expected to be true. For example, the number of items processed in the transaction should equal the total number of items claimed to be in the transaction (value from CTT segment). If the &ASSERT is not true, special action should take place. An &ASSERT is usually associated with an &ERR condition, but this is not required. • With &IF, if the expression is true, special action should take place. An &IF is usually associated with a mapping or named variable, but this is not required. • &ASSERT has 10 levels (&ASSERT0 through &ASSERT9). Level 9 assertions are always executed. Assertions at level 0 (&ASSERT0) through 8 are controlled by the assertion level used to start the translation. Thus, it is possible to turn off assertions, but &IF conditions are always checked. When using the API, the assertion level is set in the ASSERTLVL field of the translator control block. When using the WebSphere Data Interchange Utility, the assertion level is set with the ASSERTLVL keyword on the PERFORM command. See the *WebSphere Data Interchange Programmer's Reference* for additional information on the API. See "Expressions" on page 351 for more information. For an example of using this keyword, see "Example 8" on page 366.

Using literals

Table 46. Literal keywords (continued)

Keyword	S/R	Keyword description and syntax
&DATE	S/R	**Syntax:** &DATE Substitutes the system date. The length of the date field in the EDI standard data (send) or application data (receive) determines whether the date is formatted as yyyymmdd or yymmdd. The date is then edited as requested by the date edit specified in mapping. You can use the &DATE keyword as source data for any of the EDI standard data types. You can also combine the &DATE keyword with the &IFDATA, &IFNODATA, or &FORCE keywords.
&DEFERRED	S	**Syntax:** &DEFERRED &USE variable Allows you to signal that at a later time a value will be set using the same name as specified in the &DEFERRED &USE mapping. When the value is set, the data is put into the mapped segment. For example, if you had an HDR segment that contained a total number of items field, you could use &DEFERRED &USE TOTITEMS while mapping the HDR segment field. **Note:** It is not recommended that the value be set within the same segment mapping. This will produce unexpected results. If the entire segment is mapped based on the use of deferred mapping, the segment could be produced without data. At the end of the mapping, you would then repeat map some field with &SET TOTITEMS X, which will move the literal value 'X' into the HDR segment field. If X is used as a variable, the mapping would be &SET TOTITEMS &E(X), which would move the contents of the variable X into the HDR segment field.
&E	S/R	**Syntax:** &E(*expression*) The *expression* is evaluated and the result used as if it had been entered directly as a literal value. This keyword allows calculations without using a user exit. See "Expressions" on page 351 for more information. For examples of using this keyword, see "Example 7" on page 366.

Table 46. Literal keywords (continued)

Keyword	S/R	Keyword description and syntax
&ERR	S/R	**Syntax:** &ERR(*level*,*code*,*facode*,*text*) Allows you to establish your own errors for a transaction. This literal keyword can be used in one of three ways: • On the Literal line itself • &IF (expression) • &ASSERT (expression) **Note:** If you specify a field in the application name field, you cannot use &ERR on this occurrence of the element mapping. *level* is the severity of error where 1=data element, 2=segment, 3=transaction. *code* is the unique error code that should be associated with the error. This value can range from 0 to 999. WebSphere Data Interchange automatically adds 5000 to separate this value from WebSphere Data Interchange detected errors. *facode* is the functional acknowledgment error code that should be associated with this error (receive only). *text* is some text that is included in an error message logged by WebSphere Data Interchange if this error is detected. If an &ERR special literal is executed (normally controlled by either an &IF or an &ASSERTion), then WebSphere Data Interchange will log message TR0026. Within this message, the *text* and *code* will be identified. The *level* that you assign in the &ERR becomes the extended return code from the translator and thus the JCL condition code in the WebSphere Data Interchange Utility. If *level* exceeds the acceptable error level specified in the transaction usage, then the translation will not be successful and the application data will not be returned. The value of *code* plus 5000 will also be added to the list of errors for the transaction. These are available to the API programs in the ERRCDE field of the translator control block. For an example of using this keyword, see "Example 8" on page 366.
&FORCE	R	**Syntax:** &FORCE value Forces a literal value into an application field regardless of the EDI standard data element's contents. A literal value specified for receive mapping is normally used only if the EDI standard data element does not contain any data, or if an error occurs while processing the data (see "Using literals for receive mapping" on page 339). For the &FORCE keyword to be effective, you must use it in a data element of a segment that is present in the input transaction.
&FORMAT	S/R	**Syntax:** &FORMAT Substitutes the data format ID. You can also use the &FORMAT keyword with the &IFDATA, &IFNODATA, or &FORCE keywords.

Using literals

Table 46. Literal keywords (continued)

Keyword	S/R	Keyword description and syntax
&IF	S/R	**Syntax:** &IF(*expression*) The action associated with this literal is only executed if the *expression* is true. The &IF literal can be combined with: • A mapping between application fields and EDI standard data elements • A request to SET/SAVE/USE a named variable • An error condition (for example, an &IF followed by &ERR) The differences between &ASSERT and &IF are: • &ASSERT is a statement about the transaction that is expected to be true. If the &ASSERT is not true, special action should take place. With &IF, if the expression is true, special action should take place. An &IF is usually associated with a mapping or named variable, but this is not required. • &IF conditions are always checked. See "Expressions" on page 351 for more information. For examples of using this keyword, see "Example 5a" on page 364, "Example 5b" on page 365, and "Example 5a" on page 364.
&IFDATA	S/R	**Syntax:** &IFDATA value For sending, uses a literal value only if an application field contains data. For example, a segment has a pair of data elements for a qualified value and its qualifier. The application data has a corresponding field for the qualified value, but not for the qualifier. You want to supply the qualifier with a literal, but only when the application supplies a qualified value. You can do this using the &IFDATA keyword with the Application field name and Literal fields on the Map Data Element panel (TP10). &IFDATA can be used in combination with &SET or &SAVE keywords. The test performed by the translator to determine whether or not a source field contains data is different depending on the data type of the source field. For numeric data types, zeros are not considered significant, in which case the assumption is made that the field does not contain data. &IFDATA can be used in combination with &ZEROSIG if you want a zero value to be considered significant. &ZEROSIG must precede the save keyword. &IFDATA is allowed on receive but only when used in combination with the &SET or &SAVE keywords. For an example of using this keyword, see "Example 2" on page 363.
&IFNODATA	S/R	**Syntax:** &IFNODATA value Uses a literal value only if an application field does not contain data. If you do not use a keyword before the literal, &IFNODATA is used by default. &IFNODATA can be used in receive transactions, but only when used in combination with the &SET keyword.

&IFNOVAR	S/R	**Syntax:** &IFNOVAR
		
		This keyword is only valid when used with &SAVE, &LSAVE, &SET, or &LSET:
		
		`&IFNOVAR &SAVE variable`
		`&IFNOVAR &LSAVE variable`
		`&IFNOVAR &SET variable`
		`&IFNOVAR &LSET variable`
		
		The named variable will only be created if it does not already exist. If the named variable already exists, the data in it is not overlaid.
		
		For an example of using this keyword, see "Example 3" on page 363

Table 46. Literal keywords (continued)

Keyword	S/R	Keyword description and syntax
&LOOPBREAK	S	**Syntax:** &LOOPBREAK Use when an outer and an inner loop are qualified on the same record, and when you want the inner loop to be generated more than once. By putting a &LOOPBREAK in the inner loop, then the inner loop will be repeated until the &LOOPBREAK condition is met. The &LOOPBREAK condition is established using &IF; for example, `&IF((A < B) AND (X > Y)) &LOOPBREAK` **Note:** You cannot enter an application field name on the same mapping occurrence that uses &LOOPBREAK.
&LOOPCHECK	S	**Syntax:** &LOOPCHECK This keyword is very similar to &LOOPBREAK, but &LOOPCHECK does all the conditional processing automatically, based on the application field name used in the mapping that specifies &LOOPCHECK. WebSphere Data Interchange will save this field value the first time the loop is created, and will continue to create inner loops until a record with a different non-blank field value is found. **Note:** You cannot enter an application field name on the same mapping occurrence that uses &LOOPCHECK.

Using literals

Table 46. Literal keywords (continued)

Keyword	S/R	Keyword description and syntax
&LSAVE	S/R	**Syntax:** &LSAVE *variable<,position><,length> <default value>*
		Saves the value from the current data element in a named variable, but only for the duration of the loop instance. A value saved in an outer loop is available within associated inner loops. A repeating segment is considered to be a loop; therefore, values saved in one instance of a repeating segment are not available in subsequent segment iterations.
		The value to be saved is normalized prior to being stored in the variable. The normalization that takes place depends on the data type of the source field being saved. For character data types A, AC, AN, CH, or ID, any trailing blanks are removed from the source value. If the value in the field (defined with a character data type) contains all numbers, the data will be treated as a numeric data type. Leading zeros will be removed from numeric data types. For numeric data types, the source value has all leading and trailing blanks removed, it is converted to a REAL value, and all leading zeros before the decimal and trailing zeros after the decimal are removed. If a source field is numeric and contains a zero value, the variable will contain null. If a value of zero needs to be saved, &ZEROSIG must be used in combination with the save. &ZEROSIG must precede the &LSAVE keyword.
		position is optional, but if provided, it indicates the position within the current variable where the information should be saved. You can use an asterisk (*) to save at the end of the variable; for example, `&LSAVE name,*`. If *position* is not specified, the data replaces the current value of the variable.
		length is optional, but if provided, it indicates the length of data that should be saved. You can use an asterisk to save the total length of the data being supplied; for example, `&LSAVE name,*,*`.
		For example, if you want data to be saved starting in position 4 and the data you want saved is 5 characters long, the position and length would be stated as 4,5. Stating both the position and length, you can combine more than one data element into a single named variable. Special processing may be needed when combining data elements with different data types into a single named variable. See "Example 9" on page 368 and "Example 10" on page 369
		The *default value* is also optional, and if provided, the value is saved in the *variable* when the current data element contains no value.
		If the named variable does not already exist, it is created. If it already exists, the data in the existing variable is overlaid with the new data. If the data element is empty and no default value is supplied, the named variable is created but it contains no data.
		A variable established with &LSAVE does not affect the value of a variable with the same name at any other looping level. If the variable was created outside the loop with LSAVE or SAVE, and the variable is created again inside the loop, the USE for the variable inside the loop will use the value which was saved inside the loop. If the USE for the variable is outside the loop, then the value which was saved outside the loop will be used.
		For an example of using this keyword, see "Example 3" on page 363.

Table 46. Literal keywords (continued)

Keyword	S/R	Keyword description and syntax
&LSET	S/R	**Syntax:** &LSET *variable<,position><,length> value* The named variable is created or overlaid with the stated value, but only for the duration of the loop instance. A value set in an outer loop is available within associated inner loops. A repeating segment is considered to be a loop; therefore, values set in one instance of a repeating segment are not available in subsequent segment iterations.*position* is optional, but if provided, it indicates the position within the current variable where *value* should be set. You can use an asterisk to set *value* at the end of the variable, for example, *&LSET name,* abcd*. If *position* is not specified, *value* replaces the current value of the variable.
		length is optional, but if provided, it indicates the length of *value* that should be set. You can use an asterisk to set the total length of *value*, for example, &LSET name,*,* abcd. Special processing may be needed when combining data elements with different data types into a single named variable. See "Example 9" on page 368 and "Example 10" on page 369
		A variable established with &LSET does not affect the value of a variable with the same name at any other looping level. If the variable was created outside the loop with LSET or SET and the variable is created again inside the loop, the USE for the variable inside the loop will use the value which was set inside the loop. If the USE for the variable is outside the loop, then the value which was set outside the loop will be used.
		When zeros are passed in a DT application field and mapped to a DT element, they are considered significant data, and are populated to the element. In other words, WebSphere Data Interchange treats DT data as character versus numeric.
&LSID	R	**Syntax:** &LSID *value*
		Identifies the instance of the LS loop, where *value* is equal to the LS01 value in the EDI standard data. This special literal is required only if the translator has no other way to determine which LS loop is provided.
&SAMEAS	S/R	**Syntax:** &SAMEAS seqno
		Indicates when a mapping for a current data element should be exactly the same as the data element identified by <seqno>.
		For example, if the mapping for element 2 of the POC segment should be exactly the same as the mapping for element 1 of the POC segment, then specify the special literal value of '&SAMEAS 1' when mapping element 2. This results in the mapping for element 2 being the same as element 1, which is useful when qualifying (Q/S) data elements on a receive mapping.

Using literals

Keyword	S/R	Keyword description and syntax
&SAVE	S/R	**Syntax:** &SAVE *variable<,position><,length> <default value>* Saves the value from the current data element (inbound) or application field (outbound) in a named variable. The value to be saved is normalized prior to being stored in the variable. The normalization that takes place depends on the data type of the source field being saved. For character data types A, AC, AN, CH, or ID, any trailing blanks are removed from the source value. For numeric data types, the source value has all leading and trailing blanks removed, it is converted to a REAL value, and all leading zeros before the decimal and trailing zeros after the decimal are removed. If a source field is numeric and contains a zero value, the variable will contain null. If a value of zero needs to be saved, &ZEROSIG must be used in combination with the save. &ZEROSIG must precede the &SAVE keyword. *position* is optional, but if provided, it indicates the position within the current variable where the information should be saved. You can use an asterisk to save at the end of the variable; for example, `&SAVE name,*`. If *position* is not specified, the data replaces the current value of the variable. *length* is optional, but if provided, it indicates the length of data that should be saved. You can use an asterisk to save the total length of the data being supplied, for example, `&SAVE name,*,*`. For example, if you want data to be saved starting in position 4 and the data you want saved is 5 characters long, the position and length would be stated as 4,5. Stating both the position and length, you can combine more than one data element into a single named variable. Special processing may be needed when combining data elements with different data types into a single named variable. See "Example 9" on page 368 and "Example 10" on page 369. The *default value* is also optional, and if provided, the value is saved in the *variable* when the current data element contains no value. If the named variable does not already exist, it is created. If it already exists, the data in the existing variable is overlaid with the new data. If the data element is empty and no default value is supplied, the named variable is created but it contains no data. For examples of using this keyword, see "Example 1" on page 363. When zeros are passed in a DT application field and mapped to a DT element, they are considered significant data, and are populated to the element. In other words, WebSphere Data Interchange treats DT data as character versus numeric.

Table 46. Literal keywords (continued)

Keyword	S/R	Keyword description and syntax
&SET	S/R	**Syntax:** &SET variable< ,position>< ,length> value The named variable is created or overlaid with the stated value. If no default value is specified, &SET clears the variable. *position* is optional, but if provided, it indicates the position within the current variable where value should be set. You can use an asterisk to set value at the end of the variable, for example, &SET name,* abcd. If no default value is specified, &SET clears the variable. *length* is optional, but if provided, it indicates the length of value that should be set. You can use an asterisk to set the total length of value; for example, `&SET name,*,* abcd`. Special processing may be needed when combining data elements with different data types into a single named variable. See "Example 9" on page 368 and "Example 10" on page 369 When zeros are passed in a DT application field and mapped to a DT element, they are considered significant data, and are populated to the element. In other words, WebSphere Data Interchange treats DT data as character versus numeric.
&THANDLE	R	**Syntax:** &THANDLE Substitutes the WebSphere Data Interchange archive key. Can be used to assist in mapping the SAP IDOC. It enables mapping of the WebSphere Data Interchange archive key to the SAP IDOC for inbound processing. The length of the THANDLE field is 20 characters and is formatted as YYYYMMDDHHMMSSnnnnnn. It is the concatenation of the date, time, and a sequence number to ensure uniqueness.
&TIME	S/R	**Syntax:** &TIME Substitutes the system time. The length of the time field in the EDI standard data (send) or application data (receive) determines whether the time is formatted as `hhmm` or `hhmmss`. You can use the &TIME keyword as source data for any of the EDI standard data types. You can also combine the &TIME keyword with the &IFDATA, &IFNODATA, or &FORCE keywords.
&TPID	S/R	**Syntax:** &TPID Substitutes the value of the internal trading partner ID. You can also use &TPID keyword with the &IFDATA, &IFNODATA, or &FORCE keywords.
&TPNICKN	S/R	**Syntax:** &TPNICKN Substitutes the value of the trading partner nickname. You can also use &TPID keyword with the &IFDATA, &IFNODATA, or &FORCE keywords.

Using literals

Keyword	S/R	Keyword description and syntax
&USE	S/R	**Syntax:** &USE *variable<,position><,length> <default value>* For inbound transactions, the value of the named variable is the source of data for the application field. The value of the EDI standard data element being mapped is ignored. For outbound transactions, the value of the named variable is used to provide data for the EDI standard data element being mapped. An application field cannot be specified. The named variable must be saved or set using the appropriate keyword before you can use it with this keyword. *position* is optional, but if provided, it indicates the position within the variable from which the data should be retrieved. *length* is optional, but if provided, it indicates the length of data that should be retrieved. You can use an asterisk to move all data beginning at the location specified by the position parameter through the end of the variable, for example, `&USE var 3,*`. *default value* is optional, and if provided, the default value is used when the variable contains no value. **Note:** For numeric elements, a variable value of zero causes WebSphere Data Interchange to use the default value. In the case where no default is specified, as in `literal = &USE X`, there will be no output if variable X contains zero. At times, however, zero needs to be considered significant. In these cases, the user should specify the default, as in `literal = &USE X 0`. For an example of using this keyword, see "Example 1" on page 363.
&ZEROSIG	S	**Syntax:** &ZEROSIG *<default value>* Use the keyword &ZEROSIG to indicate that a zero in an application field is significant when mapping to optional or conditional data elements. Without use of this special literal, WebSphere Data Interchange considers a zero value being mapped to an optional or conditional data element as insignificant and produces no output during translation. You can combine the use of &ZEROSIG with &IFDATA, &IFNODATA, &SAVE, and &LSAVE keywords. &ZEROSIG must precede the other keyword. &ZEROSIG can also be used with a default literal to indicate that the application field is used if it contains a value, including zero, but the default literal is used if the field is blank. Binary data types preclude this because blanks represent real values.

Named variables

For receiving transactions, one way to use data in one data element for multiple application fields is to map the data element to each application field, using the repeat mapping capability. The disadvantage is that the application structures are always created, whether or not they are needed. Sometimes, you do not want the structure to be created unless some other data within the transaction is present. *Named variables* let you save the value of a data element until the time when the application structure should be created. Then when you are mapping the data element that creates the structure, you can use the repeat mapping capability to map the value in the named variable.

Named variables are also critical to the use of expressions and conditional processing. See "Expressions" on page 351 and "Conditional processing literals" on page 340 for additional information. Data values from an application field for outbound processing or standard data elements for inbound processing are not

directly available for use in an expression. The values must first be saved to a named variable using for example, the &SAVE literal, then the named variable can be used within the expression.

A variable name can be the same as your application field name, up to 16 characters, but it cannot start with any of the following:

- A numeric digit (0 through 9).
- The letter P. These variables are reserved for future use.
- The letters DI. These variables are reserved for WebSphere Data Interchange.
- An ampersand (&), so they do not get confused with special literals.
- A left parenthesis, so they do not get confused with the start of an expression.

A variable name cannot contain any of the special characters designated as Arithmetic Operators or Alternate Comparison Operators. Use of these special characters within a variable name will produce unpredictable results.

The first character of a variable name determines the life span or scope of the variable:

- If the first letter is anything other than G, the variable has transaction scope. The variable is deleted after the transaction is translated. This is the same scope as local accumulators (T0 through T9). For more information, see "Using accumulators" on page 337.
- If the first letter is G, the variable has translator session scope. The variable is not deleted until the session with the translator is terminated. This is the same scope as global accumulators (G0 through G9). For more information on accumulators, see "Using accumulators" on page 337.
- To create a variable that will exist only for the duration of the loop in which it was created, use the &LSAVE or &LSET literal keywords. When the loop repeats or terminates, any variable created using these keywords is deleted. A variable established with &LSAVE or &LSET does not disturb the value of a variable with the same name at any other looping level.

Variable names are not case sensitive. *TOTALITEMS* and *totalitems* are the same variable. Special processing may be needed when combining data elements with different data types into a single named variable. See "Example 9" on page 368 and "Example 10" on page 369.

You can combine the substring capability in mapping with the substring capability of named variables. For example, you can use sub strings to get the first 4 bytes of a standard data element, then use the *position* or *length* options to put the data in a specific place in the named variable.

Expressions

Special literals that use expressions are:

- &IF
- &ASSERT
- &E

The syntax for an expression is:

Expressions

Expression syntax

```
>>--- token ---+--- OP --- token ---+--------------------------><
               ^                     |
               +---------------------+
```
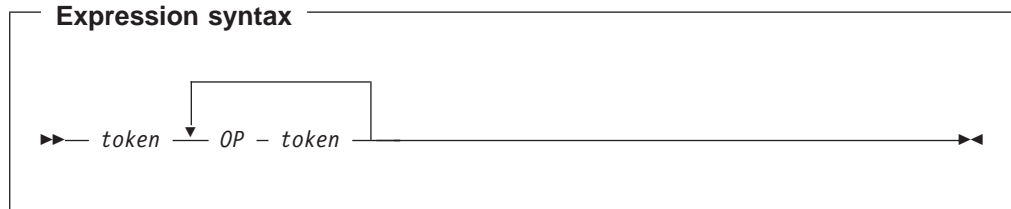
Where *token* can be any of:
- A named variable, such as TOTALS
- A numeric constant, such as 1024
- A text constant, such as 'ABCD'

  Note: You must use quotation marks around text constants to distinguish them from named variables. You can use single or double quotes (' or "), but whichever is used to start the text constant must also be used to end it.

and where *OP* is one of the following types of operators:
- Boolean
- Comparison
- Arithmetic
- Unary
- Special

By default, the data type of a variable is implicitly assigned. If the contents are all numeric digits, the data type is assumed to be numeric and is treated as such in subsequent comparisons. This assignment does not take into consideration the data type of the source data field that originally supplied the value for the variable. It is based simply on the contents of the variable. If a variable contains any nonnumeric characters, the variable data type is assumed to be character. You can override this implicit data type assignment with the use of the CHAR or NUMBER operators which work as follows:

CHAR The CHARacter operator forces WebSphere Data Interchange to treat a value as a character value rather than a numeric value.

NUMBER

The NUMBER operator forces WebSphere Data Interchange to treat a value as a numeric value rather than a character value. WebSphere Data Interchange will normally treat a value in quotes as a character value and, if both operands look like character values then what might be thought of as a numeric operator will be treated as a string operator. Thus, the expression &E('12' + '34') will yield '1234' because both operators are flagged as character data. If what you really wanted was an arithmetic addition rather than a string concatenation then you could use &E(NUMBER('12') + NUMBER('34')) which will yield 46 (as would &E(12 + 34)).>

Boolean operators

WebSphere Data Interchange has two Boolean operators: AND and OR. AND returns a value of 1 if both conditions are true, or a value of 0 if either condition is not true. OR returns a value of 1 if either condition is true, or a value of 0 if both conditions are not true. AND and OR must be entered in uppercase because they are case sensitive.

Comparison operators

Comparison operators tell WebSphere Data Interchange to compare two objects.

WebSphere Data Interchange processes comparison operators the same way it does Boolean operators. If the comparison is true, it returns a value of 1. If it is not true, it returns a value of 0.

WebSphere Data Interchange has the following comparison operators (you can use either the letters or the symbols):

EQ (=)
> The first value is the same as the second value.

GE (>=)
> The first value is greater than or equal to the second value.

GT (>) The first value is greater than the second value.

LE (<=)
> The first value is less than or equal to the second value.

LT (<) The first value is less than the second value.

NE (!=)
> The two values are not equal (including being of different data types).

Notes:
1. If you use the alphabetic operators, such as EQ, they must be uppercase.
2. Comparing a numeric constant or variable to a character constant or variable results in an invalid comparison. This produces a false result for all comparisons with the exception of Not Equal (NE or!=), in which case a true result is returned.
3. If you enclose a numeric constant in quotation marks ("), a numeric data type is still assumed for the data. Therefore, comparison to a variable that contains numeric data will not fail.

Arithmetic operators

In the following descriptions, single quote marks (') surround absolute values. WebSphere Data Interchange has the following arithmetic operators:

+ *a + b* has a value equal to the sum of the two. For numeric values, the sum is numeric. For example, 2 + 2 = 4. For nonnumeric values, the sum is a concatenation of the values. For example, 'AB' + 'CD' equals 'ABCD'. The use of quotation marks around a numeric constant has special meaning when used during addition. The enclosed value will be treated as character data and concatenated to the second value. For example, '19' + 940323 = 19940323.

- *a - b* has a value equal to the difference between the two. For numeric

values, the result is numeric. For example 4 - 2 = 2. For nonnumeric values, the result is a deconcatenation of the two values. For example, 'ABCD' - 'CD' equals 'AB'. The use of quotation marks around a numeric constant has special meaning when used during subtraction. The enclosed value will be treated as character data and deconcatenated from the second value. For example, 19940323 - '23' = 199403.

*   *a * b* has a value equal to the multiplication of *a* and *b*, if both *a* and *b* are numeric. For example, 4 * 4 = 16. If either variable is nonnumeric, the result is created by concatenating the value of the *b* after each character in *a*. For example, 'ABCD'*'Z' equals 'AZBZCZDZ'.

/   *a / b* has a value equal to the division of *a* by *b*, if both *a* and *b* are numeric. For example, 16 / 4 = 4. Division by 0 yields a 0 value. If either variable is nonnumeric, the result is created by removing each occurrence of *b* from *a*. For example, 'ACDBCD'/'CD' equals 'AB'.

CHAR

The CHARacter operator forces WebSphere Data Interchange to treat a value as a character value rather than a numeric value.

NOT   An operator to reverse the Boolean value of an expression. The exclamation point (!) is a short hand for the NOT operator. NOT(value) has the following meanings:

1.   If value is numeric, then NOT(value) is 1 if the value is 0 and 1 otherwise. Thus, NOT(0) yields 1 and NOT(1) yields 0.

2.   If value is a string value, then NOT(value) is 1 if the string has no length and zero otherwise. Thus, NOT('abc') yields 0 and NOT('') yields 1.

NUMBER

The NUMBER operator forces WebSphere Data Interchange to treat a value as a numeric value rather than a character value. WebSphere Data Interchange will normally treat a value in quotes as a character value and, if both operands look like character values then what might be thought of as a numeric operator will be treated as a string operator. Thus, the expression &E('12' + '34') will yield '1234' because both operators are flagged as character data. If what you really wanted was an arithmetic addition rather than a string concatenation then you could use &E(NUMBER('12') + NUMBER('34')) which will yield 46 (as would &E(12 + 34)).

RD or:

a RD *n* has a value equal to *a* rounded to *n* decimal places if *a* is numeric. For example, 4321.556 RD 2 equals 4321.56. If *a* is nonnumeric, the result is created by taking the first *n* characters from *a*. For example, 'ACDBCD' RD 3 equals 'ACD'. If *n* is less than or equal to zero it is interpreted to be a request to remove leading blanks. Thus, ' ABC' RD 0, yields 'ABC'. If using the character string RD for rounding instead of the special character:, it must be entered using uppercase.

TU or;

a TU *n* has a value equal to *a* truncated to *n* decimal places if *a* is numeric. For example, 4321.556 TU 2 is 4321.55. Notice that the number 6 is dropped from the result and not rounded. If *a* is nonnumeric, the result is created by taking the last *n* characters from *a*. For example, 'ACDBCD' TU 3 yields 'BCD'. If *n* is less than or equal to zero it is interpreted to be a request to remove trailing blanks. Thus, 'ABCƀƀƀ' TU 0 yields 'ABC'. If using the character string TU instead of the special character;, it must be entered using uppercase.

Unary operator

WebSphere Data Interchange has one unary (single component) operator, "-".

&E(-*a*) changes the sign of *a*. If *a* does not exist, the value is 0.

Special operators

UE   &E(*a* UE 'MYPROG') returns a value from the user-written program MYPROG.

> **Note:** As mentioned under the descriptions of "&LSAVE" on page 346 and "&LSET" on page 347, variables are normalized to REAL value equivalents. When using this special operator, the variable value that is passed is converted back to its N*n* numeric format.

TS   &E(*a* TS 'MYTABL') translates the local value *a* to the standard value using the MYTABL translation table.

> **Notes:**
> 1. As discussed under the descriptions of "&LSAVE" on page 346 and "&LSET" on page 347, variables are normalized to REAL value equivalents. When using this special operator, the variable value that is passed is converted back to its N*n* numeric format.
> 2. If the translation table specified does not exist, or the value passed does not match an entry in the table, then no data is produced from this instruction and no errors or exceptions are issued.

TL   &E(*a* TL 'MYTABL') translates the standard value *a* to the local value using the MYTABL translation table.

> **Notes:**
> 1. As discussed under the descriptions of "&LSAVE" on page 346 and "&LSET" on page 347, variables are normalized to REAL value equivalents. When using this special operator, the variable value that is passed is converted back to its N*n* numeric format.
> 2. If the translation table specified does not exist, or the value passed does not match an entry in the table, then no data is produced from this instruction and no errors or exceptions are issued.

IN   &E(*a* IN 'MYTABL') returns a value equal to 1 if *a* exists in the MYTABL validation table.

> **Notes:**
> 1. As discussed under the descriptions of "&LSAVE" on page 346 and "&LSET" on page 347, variables are normalized to REAL value equivalents. When using this special operator, the variable value that is passed is converted back to its N*n* numeric format.
> 2. If the validation table specified does not exist, or the value passed does not match an entry in the table, then no data is produced from this instruction and no errors or exceptions are issued.

SC   &E(*a* SC *max.dec*) scales a real number to a maximum of *max* digits with a maximum of *dec* decimal places, truncating unused digits. For example:

```
&E(1234.56 SC 4.2) gives 1234
&E(1234.56 SC 6.2) gives 1234.56
&E(1234.56 SC 6.1) gives 1234.5
```

```
&E(1234.56 SC 8.1) gives 1234.5
&E(1234.56 SC 8.8) gives 1234.56
&E(1234.56 SC 3.3) gives 1234.56
&E(1234.56 SC 4.5) gives 1234
```

- If *max* is less than the number of significant digits to the left of the decimal point, the SC is ignored.
- If *dec* is greater than *max*, *dec* is set equal to *max*.

SC applied to strings provide a substring capability. When applied to strings, the format is &E('STRING' SC *pos.len*). For example:

```
&E('ABCDEF' SC 4.2) gives 'DE'
&E('ABCDEF' SC 1.5) gives 'ABCDE'
&E('ABCDEF' SC 9.1) gives ''
&E('ABCDEF' SC 1.9) gives 'ABCDEF'
&E('ABCDEF' SC .1)  gives 'A'
&E('ABCDEF' SC 5)   gives 'E'
```

If you do not provide *pos* or *len*, WebSphere Data Interchange uses 1 for that value.

SR &E(*a* SR *max.dec*) scales a real number to a maximum of *max* digits with a maximum of *dec* decimal places, rounding the value. For example:

```
&E(1234.56 SR 4.2) gives 1235
&E(1234.56 SR 6.2) gives 1234.56
&E(1234.56 SR 6.1) gives 1234.6
&E(1234.56 SR 8.1) gives 1234.6
&E(1234.54 SR 8.1) gives 1234.5
```

IS &E(*a* IS *pattern*) has a value equal to *a* but establishes a *pattern* for the data within *a*. This *pattern* only has meaning when using the TO operator (next). All variables have a default pattern of "ABCDEFGHIJKLMNOPQRSTUVWXYZ".

TO &E(*a* TO *pattern*) has a value that is created by matching the pattern associated with *a* with the *pattern* in this expression. A character from TO *pattern*, if located in the IS *pattern* and if found the corresponding character from *a*, is moved to the result field. If a match cannot be found, the character from the TO *pattern* is moved to the result field. For example:

- To reverse a string:

  ```
  &E('PLEH' IS 'ABCD' TO 'DCBA') gives 'HELP'
  ```
- To insert delimiters:

  ```
  &E('HHMM' IS 'ABCD' TO 'AB:CD') gives 'HH:MM'
  ```
- To remove delimiters:

  ```
  &E('HH:MM' IS 'ABCDE' TO 'ABDE') gives 'HHMM'
  ```

The last three examples all show the use of both the IS and TO operators for clarity. The IS operator is not really necessary because all variables automatically have the default pattern described in the IS operator. Therefore:

```
&E('PLEH' TO 'DCBA') gives 'HELP'
&E('THISAMEG ' TO 'ABCDICDIEIFGDDEHG') gives 'THIS
IS A MESSAGE'
```

Date conversion special operators

WebSphere Data Interchange allows for any-to-any date conversions. The format of the any-to-any date conversion operator is:

`&E(`*variable* `FD` *mask* `TD` *mask*`)`

where:

variable
> The value to be converted.

FD   From Date operator. This signals that the following token is the mask that describes the date format in *variable*.

TD   To Date operator. This signals that the following token is the mask that describes the date format that is wanted.

mask   The mask that describes the FROM or TO date format. A mask consists of symbols which identify the date provided or date wanted. Symbols may be in upper or lower case. Any value in the mask that is not one of the symbols below is expected to be physically part of the source data (FD) or will become physically part of the result data (TD).

CC	Century
YY	Year
MM	Month of year
DD	Day of month
D	Day of month as a single character, if possible
HH	Hour of day
MM	Minute of hour
II	Minute of hour

> MM can be used when it immediately follows HH as in HHMM; however, if you want minute followed by hour, you must use IIHH, because MMHH would be interpreted as month of year and Hour of day.

SS	Second of minute
WW	Week of Year (1 through 52)
K	Day of Week (Monday=1, Tuesday=2, etc.)

> D can be used if it immediately follows WW as in WWD; however, if you want day of week followed by week, you must use KWW, because DWW would be interpreted to be day of month and Week of year.

JJJ	Julian day of year
Q	Quarter (1,2,3,4)
E	Semester
ZZZ	Time zone
TM	Textual month (i.e., January, February, etc.)

> TM may be followed by the name of a translate table to convert a textual month to a numeric month (FD), or from a numeric month to

a textual month (FD). If a table name is not provided, the default table names are DIMONTXT to translate from text to numeric and DIMONNUM to translate from numeric to text. A table name is indicated using parentheses, for example:

```
TM(tablename)
```

where *tablename* must be a constant.

After processing the From Date and before creating the To Date, the following processing will be done.

1. SS (seconds) will be defaulted to 0.
2. CC (century) will be defaulted to 19 when the YY (year) is greater than 10 and to 20 otherwise.
3. JJJ (julian day) will be created based on WW (week of year) and K (day of week) if not otherwise provided and WW and K were provided.
4. JJJ (julian day) will be created based on MM (month of year) and DD (day of month) if not otherwise provided and MM and DD were provided.
5. JJJ (julian day) will be used to determine WW (week of year), K (day of week), if either WW or K was not provided.
6. JJJ (julian day) will be used to determine MM (month of year), DD (day of month), if either MM or DD was not provided.
7. Q (Quarter) will be determined based on MM (month of year) if not otherwise provided.
8. E (Semester) will be determined based on MM (month of year) if not otherwise provided.

Here are some examples using CONSTANTS for all values:

- Simple example to remove delimiters.

  ```
  &E('96/06/07' FD 'YY/MM/DD' TD 'YYMMDD') yields '960607'
  ```

- Simple example to remove delimiters and rearrange.

  ```
  &E('96/06/07' FD 'YY/MM/DD' TD 'MMDDYY') yields '060796'
  &E('96/06/07 EDT' FD 'YY/MM/DD ZZZ' TD 'ZZZ MMDDYY') yields 'EDT 060796'
  ```

- Change delimiters and convert from one form to another.

  ```
  &E('96/06/07' FD 'YY/MM/DD' TD 'YY:JJJ') yields '96:157'
  ```

- Textual month to Numeric month.

  ```
  &E('June 7, 1996' FD 'TM D, CCYY' TD 'YYMMDD') yields '960607'
  ```

- Numeric month to Textual month.

  ```
  &E('060796' FD 'MMDDYY' TD 'TM D, CCYY') yields 'June 7, 1996'
  ```

- Numeric month to Textual month with a special translate table that uses abbreviations for the months.

  ```
  0 FD 'MMDDYY' TD 'DDTM(ABBREV)CCYY') yields '07JUN1996'
  ```

Newer releases of X12 and EDIFACT standards contain segments with variable date/time formats. The format is determined by a qualifier value in the segment. WebSphere Data Interchange provides two tables for dynamically translating the qualifier into a mask. Table EDIXDTMSK is used for X12 and table EDIEDTMSK is used for EDIFACT.

Assume the following X12 data is received in the DTP segment (Date or Time period):

- `DTP**D8*19940927!`

Where D8 is the date or time period format qualifier (CCYYMMDD) and 19940927 is the date or time period.

Assume your application requires the date in YYMMDD format. You map the DTP segment as follows:

- DTP03 - &SAVE Qual

  Results in Qual is D8

- DTP03 - &SAVE Date

  Results in Date is 19940927

- DTP03 - &FORCE &E(Date FD (QUAL TS 'DIXDTMSK') TD 'YYMMDD')

  Results of (Qual TS 'DIXDTMSK') is CCYYMMDD

  Results of &FORCE is 940927

Order of precedence

During processing, all expressions are evaluated from left to right. The order of precedence is:

1. Unary minus (-)
2. Rounding (RD) and truncating (TU)
3. Special operators (UE, TS, TL, IN, IS, TO, SC, SR, FD, TD)
4. Multiply (*), Divide (/)
5. Addition (+), Subtraction (-)
6. Relational Operators (GT, GE, LT, LE, EQ, NE)
7. Boolean (AND)
8. Boolean (OR)

Precedence can be overridden with parentheses embedded within an expression. For example, &E(2+3*5) equals 17 because the multiplication is done first, then the addition. &E((2+3)*5) equals 25 because the parentheses indicate that the addition is done first, then the multiplication.

DI variables

WebSphere Data Interchange has reserved the prefix DI for variables that will be reserved for use WebSphere Data Interchange to accomplish special functions. The following DI variables are currently available and are used with the &SET keyword:

DIAPPFILE

Use this variable to change the name of the file to which the translation application data will be written during a Receive Translate. It will override any value that was used in the receive usage or in the application data format definition. It provides the capability for data that is being received to influence the final destination for the data. For example, the statement

```
&SET DIAPPFILE SPECIAL
```

would force the current transaction to be written to the application file identified by the ddname SPECIAL.

DIAPPTYPE

This variable sets the application file type that corresponds with the file name provided by DIAPPFILE.

DIAUTOCC

Allows automatic century manipulation for both inbound and outbound translation. Century will be automatically added or removed from the date using the length of the standard data element or application field. For example, the following statement uses a value of 1:

```
&SET DIAUTOCC 1
```

DICCCTRL

Use to remove the century control year from the translator and allow selection of the century control year. If year is greater than 10, century is 19; otherwise century is 20. The century control year is 10. For example, if year is less than 95, century is 20, control year is 95:

```
&SET DICCCTRL 95
```

DICUSERDATA

This variable is used to set the data value that will be inserted in the TRCB field cuserdata. This field is copied to any output 'C' record before the 'C' record is written. Received data can be placed into a named variable in any combination up to 256 bytes. Then use the reserved variable DICUSERDATA anytime the value of the named variable needs to be placed into the TRCB. For example, suppose a named variable tvar has been created and filled with the data from a previously mapped data element.

That data can be placed in the TRCB cuserdata field by including the following in a map:

```
&SET DICUSERDATA &E(TVAR)
```

DIERRFILTER

This variable can be used to control which errors are actually meaningful to you at a point in time during a translation. A description of the error filter can be found in the *WebSphere Data Interchange Programmer's Reference*.

DIEXPTRACE

This variable, when given a nonzero value (&SET DIEXPTRACE 1), causes WebSphere Data Interchange to create a TRACE of the results of all expression evaluations. When tracing is active, WebSphere Data Interchange will write out message TR0411 to the PRTFILE for each expression. The message will show the expression being evaluated and the result of the evaluation. Tracing will remain active until the DIEXPTRACE is given a zero value (&SET DIEXPTRACE 0).

Note: The TR0410 and TR0411 messages always occur as the first messages for a transaction. They are not merged with any other error messages for the transaction.

DIMAPCHAIN

This variable can be used when an inbound transaction is required by more than one application program. It allows more than one mapping to be executed for the specified transaction. The last value given to DIMAPCHAIN in a mapping will establish the application sender ID value that will be used to locate the next mapping to execute. For example, if MAPABC had this coded:

```
&SET DIMAPCHAIN APPLICATIONB
```

The inbound transaction would be translated using map MAPABC, and then it would be translated using the map that is associated with application

sender ID APPLICATIONB. The DIMAPCHAIN command will cause all maps indicated by each DIMAPCHAIN command to be translated, whereas the DIMAPSWITCH command will stop translating the map that has the DIMAPSWITCH variable in it, and literally switch to the new map indicated in the command.

DIMAPSWITCH

This variable can be used when data being received needs to be inspected before it can be determined exactly what mapping should be done against the transaction. It allows you to switch the map that is being executed dynamically based on the data that is being received. A map could be created to initially look at the data being received. Only those data elements necessary to make a mapping decision would be mapped. WebSphere Data Interchange would determine the real map to be used by interpreting values resulting from conditional logic expression. For example, a map would contain conditional logic expression:

```
&IF(X > Y) &SET DIMAPSWITCH APPLICATIONA
```

Here, if X is greater than Y, the mapping identified with an application sender ID value of APPLICATIONA would be used to translate the transaction.

DISAPSEQ

This variable can be used to allow saving of the SAP IDOC record sequence number on the first error encountered during outbound processing. The sequence number may be provided through the application or using the WebSphere Data Interchange accumulators. Variable DISAPSEQ is captured in the SAP status record to indicate the first record in error. For more information, see the *WebSphere Data Interchange Programmer's Reference*.

DIVALLEVEL

This variable can be used to control the level of validation done. It can have the same values as the validation level specified in a usage record, which are: 0 (no validation), 1 (validation tables activated), and 2 (validation tables plus type checking). Any value other than 0, 1, or 2 will be treated as a 0.

DIVALTYPE

This variable can be used to control the data types for which data type checking is done (validation level of 2). The types that may be specified are DT, TM, N, R, CH, AN, A, and HX. They must be specified in uppercase and separated by a comma. Any value specified that is not in the list above will be ignored. For example, to activate DT, TM and HX validation, the following could be done:

```
&SET DIVALTYPE DT,TM,HX
```

DIVARTRACE

This variable, when given a nonzero value (&SET DIVARTRACE 1), causes WebSphere Data Interchange to create a TRACE of all accesses to variables. When tracing is active, WebSphere Data Interchange will write out message TR0410 to the PRTFILE for each variable access. The message will indicate the variable being accessed and its current value. Tracing will remain active until the DIVARTRACE is given a zero value (&SET DIVARTRACE 0).

Mapping techniques for literal keywords

As you will see in the following examples, deciding where to save variables, execute expressions, and subsequently use variables is fundamental. To determine where these operations should be done, it is essential to understand the order in which the translator executes element mapping instructions. As stated earlier, transactions are always processed starting with the first data element of the first segment and proceeding to the next element of the same segment, after which the next segment is processed, and so on. If one element has repeat mappings, the instructions are executed in a top-down fashion.

It is equally important to understand the difference between a mapping that may contribute to the output versus one that does not. For example, specifying &SAVE *variable* or &SET *variable* will not contribute to the output directly. Only direct mappings such as specifying an Application Field Name, forcing a literal value, specifying &USE *variable*, or evaluating an &IF expression, can contribute to the output. Determining where to save and use variables is different for inbound and outbound translation.

For send processing, proper placement is easy for mappings that contribute to the output because you know which element requires the result. Proper placement for a mapping that does not contribute is not as obvious because there may be no data element relationship with this action. In most cases, you will save an application value into a variable, check or manipulate the variable, then use it. In other cases, it may not be appropriate to perform all these actions in successive repeat element mappings. For instance, you may have two independent looping structures (records) and you need to save a value from a particular iteration in the first loop. This particular value must be saved while the translator is processing this first loop. The saved variable can then be inspected and manipulated in the second repeating loop to provide the desired result.

In summary, the location of a mapping that saves a variable can be far from a corresponding mapping that actually uses the variable. The best technique for deciding where to save values is to do so at or near processing points in the map when the translator is working on the corresponding record.

Note: Typically, the map for saving a value into a variable will precede the event of actually using the variable. A feature called &DEFERRED &USE can be utilized in the event that output is necessary in an earlier segment than where the final result will be set.

For receive processing, the situation is reversed. It is easy to decide where to save variables because you already know which element value in the input you need to work with. Proper placement for a mapping that contributes to your output application data is not as obvious because there may be no data element relationship with this action. In most cases, you will save a data element value into a variable, check or manipulate the variable, then use it in your application record. In other cases, it might not be appropriate to perform all these actions in successive repeat element mappings. For instance, you might need to save values from two different segments, compare them, and write the result to an application field. One input element that needs to be saved is in the header section of the EDI transaction and the second is in an outer loop (the name loop for example).

The output application field that needs to be output after comparing these two variables is related to the detail loop (line item loop for example). It is most appropriate to use the result within the detail loop because this may have an

independent repeating record that is not associated with the name or header segments. Hence, the comparison and use of the result should be done within this detail loop that controls the creation of the detail record. The best technique for deciding where to use variables is to do so at or near processing points in the map when the translator is working on the corresponding record.

Note: For receive processing, the map for saving a value into a variable must precede the event of actually inspecting or using the variable. The feature of &DEFERRED &USE cannot be used during inbound translation.

Examples of using literal keywords and named variables

Example 1

For an inbound transaction, if your trading partner sends you the city name, state abbreviation, and zip code in three fields, but your database puts all of this information in one 30-byte field, ADDRESS, you could use &SAVE to put the address information into the single field:

1. Map the city name data element using the literal &SAVE citystzip,1,19.

   This creates the named variable citystzip and places the received city in the first 19 bytes of the named variable.

2. Map the state abbreviation data element using the literal &SAVE citystzip,20,2.

   This places the received state abbreviation in the bytes 20 and 21 of the named variable.

3. Map the zip code data element using the literal &SAVE citystzip,22,9.

   This places the received zip code in bytes 22 through 30 of the named variable.

4. Repeat the mapping of the zip code data element. Specify application field ADDRESS and the literal &USEƀcitystzip.

   WebSphere Data Interchange uses the value in the named variable that was concatenated in the first three steps.

Example 2

For an outbound transaction, you want to provide a telephone number, either of a specific contact (CONTACTPHONE) or of the organization (ORGPHONE). If a contact phone number is provided, you want to use it; otherwise, you want to use the organization phone number. CONTACTPHONE occurs before ORGPHONE.

1. In the first mapping of the phone number data element, specify application field CONTACTPHONE and the literal **&IFDATA &SAVE Tphone**.

   This creates the named variable *Tphone* only if the CONTACTPHONE field contains data.

2. Repeat the mapping of the phone number data element. Specify application field ORGPHONE and the literal **&IFNOVAR &SAVE Tphone**.

   This creates the named variable *Tphone* only if the named variable did not already exist (CONTACTPHONE did not contain any data).

3. Repeat the mapping of the phone number data element. Specify the literal **&USE Tphone.**

Example 3

For an inbound transaction, the application field NAME should receive the value from EDI standard data element 123 (CUSTNAME) in the first occurrence of the name and address loop, or from the data element (ORGNAME) in the second occurrence of the loop, if the first occurrence does not contain data.

Appendix C. Advanced send and receive mapping **363**

Literal keywords and named variables

1. When mapping the first occurrence of the loop, specify the literal **&IFDATA &SAVE Name**. Do not specify an application field.

   This will create the named variable Name only if the first occurrence contains data.

2. When mapping the second occurrence, specify the literal **&IF(Name EQ ") &LSAVE Name**. Do not specify an application field.

   This will create the named variable Name only if it does not already exist (the first occurrence did not contain any data). &LSAVE is used so that the value of Name from the first occurrence of the loop is not disturbed.

3. Repeat the mapping for the second occurrence. Specify application field NAME and the literal **&USE Name**.

   This will use either the value saved in step 2 (first occurrence did not contain a value) or the value saved in step 1 (first occurrence did contain a value). All future occurrences of the loop and unrelated loops and/or segments will only see the value of *Name* saved in step 1.

Example 4

For an inbound transaction, the application field NAME should receive the value from EDI standard data element 123 (CUSTNAME) in the second occurrence of the name and address loop, or from the data element (ORGNAME) in the first occurrence of the loop, if the second occurrence does not contain data.

1. When mapping the first occurrence of the loop, specify the literal **&SAVE Name**. Do not specify an application field.

   This will create the named variable Name.

2. When mapping the second occurrence, specify the literal **&IFDATA &SAVE Name**. Do not specify an application field.

   This will overlay the named variable Name only if the second occurrence contains data.

3. Repeat the mapping for the second occurrence. Specify application field NAME and the literal **&USE Name.**

   This will use either the value saved in step 2 (second occurrence contained a value) or the value saved in step 1 (second occurrence did not contain a value).

Example 5a

For an outbound transaction, either application field ORDQTY or MINQTY should supply a value for an EDI standard data element. The field with the largest value should be used. If neither ORDQTY or MINQTY contain data then a value of 100 should be used.

1. In the first mapping of the data element, specify application field ORDQTY and the literal **&SAVE QTY1**.

   This saves the value of the application field to a named variable. If the application field does not contain a value, the variable value is zero.

2. Repeat the mapping of the data element. Specify application field MINQTY and the literal **&SAVE QTY2**.

   This saves the value of the application field to a named variable. If the application field does not contain a value, the variable value is zero.

3. Repeat the mapping of the data element. Specify the literal **&IF(QTY1 >= QTY2) &USE QTY1 100**. Do not specify an application field.

   This compares the values of the named variables. If QTY1 is greater than or equal to QTY2, WebSphere Data Interchange will use QTY1. If neither ORDQTY or MINQTY contain a value then both QTY1 and QTY2 will have a

value of 0 and will therefore be equal. However, a value of 0 is not significant, which causes WebSphere Data Interchange to use the default literal value of 100.

4. Repeat the mapping of data element 123. Specify the literal **&IF(QTY1 < QTY2) &USE QTY2**. Do not specify an application field.

   This compares the values of the named variables. If QTY1 is less than QTY2, WebSphere Data Interchange will use QTY2.

Example 5b

Assume you have created a mapping similar to that of example 5a, but forgot to do step 1. Because it is necessary to define QTY1 before using it, you need to insert a mapping of this data element, making it the first occurrence of the loop. Instead of remapping all of the occurrences, perform the following:

1. Drag application field ORDQTY to the data element.
2. Open the Mapping Data Element Editor for the new mapping.
3. Enter the literal &SAVE QTY1 and press OK.
4. Drag the new mapping to the appropriate location and drop it there.

Example 6

Your application can generate three different discount rates: the regular discount (REGDISC), a volume discount (VOLDISC), and a special discount (SPECDISC). For an outbound transaction, if application field SPECDISC contains a value, you want to use it for EDI standard data element 456. However, if SPECDISC does not contain a value, then the larger value of either REGDISC or VOLDISC should be used.

1. In the first mapping of data element 456, specify application field REGDISC and the literal **&SAVE REGDISC**.

   This saves the value of the application field to a named variable. If the application field does not contain a value, the variable value is zero.

2. Repeat the mapping of data element 456. Specify application field VOLDISC and the literal **&SAVE VOLDISC**.

   This saves the value of the application field to a named variable. If the application field does not contain a value, the variable value is zero.

3. Repeat the mapping of data element 456. Specify application field SPECDISC and the literal **&IF(REGDISC >= VOLDISC) &E(REGDISC)**.

   This mapping will be executed only if the value of REGDISC is greater or equal to the value of VOLDISC. If this is the case, then SPECDISC will be mapped to the data element. However, if SPECDISC does not contain any data, then the default literal value of &E(REGDISC) will be used.

4. Repeat the mapping of data element 456. Specify application field SPECDISC and the literal **&IF(REGDISC < VOLDISC) &E(VOLDISC)**.

   This mapping will be executed only if the value of REGDISC is less than the value of VOLDISC. If this is the case, then SPECDISC will be mapped to the data element. However, if SPECDISC does not contain any data, then the default literal value of &E(VOLDISC) will be used.

Notes on examples 5 and 6

Examples 5 and 6 illustrate the differences between &IF, &USE, and &E:

- &IF is used to determine if a mapping should be executed. The value of the &IF expression is not used in the mapping; it only controls the execution of the mapping. If the expression is true (nonzero value), the mapping is executed. If

Literal keywords and named variables

the expression is false (zero value) the expression is not executed. In examples 5 and 6, only one of the maps in steps 3 and 4 will be executed because the expressions are mutually exclusive.

- &USE indicates that a named variable should be used as the primary source for data in the mapping. An application field cannot be specified, but you can have a default literal value if the variable name being used does not contain any data.

- &E is used exactly the same as a literal value, but instead of having a constant literal value, &E allows a literal value to be computed, taken, or computed and taken from a named variable. If the last operator of an expression is a Boolean or comparison operator, then the value of the expression will either be true (1) or false (0), and you can map these values.

For example, if a data element should contain a 1 if FLD1 is greater than FLD2, or a 0 if it is not, then mapping the expression &E(FLD1 > FLD2) would result in either a 1 or 0 being moved to the EDI standard data element. If a data element should contain a 1 if FLD1 is greater than FLD2, or nothing if it is not, then the conditional mapping &IF(FLD1 > FLD2) 1 would map the constant value of 1 only when FLD1 is greater than FLD2.

Example 7

For an outbound transaction, data element 321 should have a value of S if the value of application field SIZE is 6, 7, or 8; a value of M if SIZE is 9, 10, 11, or 12; and a value of L if SIZE is 13, 14, or 15. These values are specified in the translation table SIZETAB. However, for this transaction, SIZE can also be less than 6, in which case data element 321 should have a value of XS, or greater than 15, in which case data element 321 should have a value of XL.

1. In the first mapping of data element 321, specify application field SIZE and the literal **&SAVE Size**.

   This saves the value of the application field to a named variable. If the application field does not contain a value, the variable value is zero.

2. Repeat the mapping of data element 321. Specify the literal **&IF(Size > 0 AND Size < 6) XS**.

   This compares the value of the named variable to 6 and 0. If the value is less than 6 but greater than 0, WebSphere Data Interchange uses the value XS.

3. Repeat the mapping of data element 321. Specify the literal **&IF(Size > 15) XL**.

   This compares the value of the named variable to 15. If the value is greater than 15, WebSphere Data Interchange uses the value XL.

4. Repeat the mapping of data element 321. Specify application field SIZE and the literal **&IF(Size >= 6 AND Size <= 15) &E(Size TS 'SIZETAB')**.

   This compares the value of the named variable to 6 and 15. If the value is greater than or equal to 6, and less than or equal to 15, WebSphere Data Interchange uses the translation table SIZETAB to determine the corresponding value for the named variable.

Example 8

Assume you have a lumber supply business and your trading partners are home builders in the area. The unit of measure on their orders ranges from inches to rods (one rod equals 5 yards). You have implemented just-in-time inventory processes so if you receive an order that requires additional inventory, your order is immediately sent to your supplier. Your application stores all measurements in board feet, and therefore must convert all incoming data to board feet.

Your application fields are QUANTITY and UNITMEAS. Using the conditional processing literals, you would:

1. Map the unit of measure data element, using the literal `&SAVE UOM`. Do not specify an application field.

   This saves the present value of the standard data element in the named variable UOM.

2. Repeat the mapping of the unit of measure data element, using the literal **&IF(UOM EQ 'IN' OR UOM EQ 'BF' OR UOM EQ 'RD') &SET UOMOK 1**. Do not specify an application field.

   This statement enables you to determine if you have only the values you want, and if the statement is true, WebSphere Data Interchange puts a 1 in the named variable UOMOK for later use.

3. Repeat the mapping of the unit of measure data element. Specify application field UNITMEAS and the literal **&IF (UOMOK EQ 1) &FORCE BF**.

   This forces the value 'BF' into the application field only when the unit of measure is a valid value. The steps that follow will convert the data received into board feet.

4. Map the quantity data element, using the literal **&SAVE QTY**. Do not specify an application field.

   This saves the present value of the standard data element in the named variable QTY.

5. Repeat the mapping of the quantity data element. Specify application field QUANTITY and the literal **&IF(UOM EQ 'IN') &FORCE &E(QTY/144)**.

   This statement saves the value of quantity converted to board feet in the application field QUANTITY only if the incoming unit of measure was 'IN'.

6. Repeat the mapping of the quantity data element. Specify application field QUANTITY and the literal **&IF(UOM EQ 'RD') &FORCE &E(QTY*198/144)**.

   This statement saves the value of quantity converted to board feet in the application field QUANTITY only if the incoming unit of measure was 'RD'.

7. Repeat the mapping of the quantity data element. Specify application field QUANTITY and the literal **&IF(UOM EQ 'BF') &FORCE &E(QTY)**.

   This statement saves the value of quantity converted to board feet in the application field QUANTITY only if the incoming unit of measure was 'BF'.

8. Repeat the mapping of the unit of measure data element, using the literal **&ASSERT1(UOMOK EQ 1) &ERR(2,100,,'Invalid unit of measure')** Do not specify an application field.

   This statement creates a translation error for anything that does not meet our criteria. Assume you received something in yards. You could add a repeat mapping for yards, then retranslate. The transaction would then pass through the translation.

An alternative to using conditional processing literals would be a single mapping using a translation table. For example, you could set up a translation table UOMDIV as follows:

Table 47. Sample translation table UOMDIV

Local value	Standard value
IN	144
RD	1.375
BF	1

Literal keywords and named variables

If a translation table is used, then the mapping could be reduced to the following:

1. Map the unit of measure data element, using the literal **&SAVE UOM**.
2. Repeat the mapping of the unit of measure field, using the literal **&SET divisor &E(UOM TS 'UOMDIV')**.
3. Repeat the mapping of the unit of measure field, using the literal **&IF divisor NE 0) &FORCE BF**.
4. Map the quantity data element using the literal **&SAVE QTY**.
5. Repeat the mapping of the quantity data element, using the literal **&FORCE &E(QTY/divisor)**.
6. Repeat the mapping of the unit of measure data element, using the literal **&ASSERT1(divisor NE 0) &ERR(2,100,,'Invalid unit of measure')**.

This method might be preferred if the values of UOM are expected to change. If this happens, only the UOMDIV table needs to be updated rather than changing the mapping.

Example 9

For an outbound transaction, the application field TEST1 is an N2 data type and contains the value 100. You need to build a variable that contains "XX" in the first and second positions, "01" in the third, forth, fifth, and sixth positions, and the value in the application field TEST1 beginning in the seventh position.

To put this information into a single variable, do the following:

1. Map a data element using the literal **&SET TVAR**.

   This creates the named variable TVAR and sets the variable to null.
2. Repeat the data element and map the data element using the literal **&SET TVAR,*,2 XX**.

   This places "XX" in bytes 1 and 2 of the variable TVAR.
3. Repeat the data element and map the data element using the literal **&SET TVAR,*,4 01**.

   This places "01" in bytes 3 through 6 of the variable TVAR.
4. Repeat the data element and map the data element using the application field TEST1 and a literal **&SAVE TVAR,*,7.**

   This places "100" in bytes 7 through 9 of the variable TVAR. The variable TVAR now contains the following:

   ```
   XX01  100
   ```

   Note: At this point, the variable TVAR has been normalized to the data type of the application field TEST1, which is defined as an N2 data type.

To map TVAR position 3 through 6 to an EDI standard data element with an ID data type, you would need to do the following because of the normalization of the variable to an N2 data type:

1. Map a data element using the literal **&E(CHAR(TVAR) SC 3.4)**.

   The &E allows the literal to be computed. The CHAR forces the variable TVAR to have a data type of character for this instruction. The SC is a scaling function but can be used to substring character data. The resulting value would be "01".

An alternative to using the scaling function is as follows:

1. Map a data element using the literal **&E(CHAR(TVAR) TO 'CDEF')**.

The TO 'CDEF' is used to create the value by matching the default pattern ('ABCDEFGHIJK...') with the 'CDEF' pattern in this expression. The resulting value would be "01".

2. Map a data element using the literal **&USE TVAR,3,4.**

The resulting value would be "0.01".

To map TVAR position 7 through 9 to an EDI standard data element with an R data type, you can move the data.

1. Map a data element using the literal **&USE TVAR,7,3**. The resulting value would be "1.00".

Example 10

For an outbound transaction, the application field TEST1 is an N2 data type and contains the value 123. Application field TEST2 is an N0 data type and contains the value 100.

To put this information into a single variable do the following:

1. Map a data element using the literal **&SET TVAR.**

   This creates the named variable TVAR and sets the variable to null.

2. Repeat the data element and map the data element using the application field TEST1 and a literal **&SAVE TVAR,*,3.**

   This places "123" in bytes 1 through 3 of the variable TVAR.

3. Repeat the data element and map the data element using the application field TEST2 and a literal **&SAVE TVAR,*,3**.

   This places "100" in bytes 4 through 6 of the variable TVAR.

The variable TVAR now contains the following:

```
123100
```

Note: At this point the variable TVAR has been normalized to the data type of the application field TEST1 which is defined as an N0 data type.

To map TVAR position 1 through 3 to an EDI standard data element with an R data type, you would need to do the following because of the normalization of the variable to an N2 data type:

1. Map a data element using the literal **&SET TEMPVAR &E(CHAR(TVAR) SC 1.3)**.

   The &E allows the literal to be computed. The CHAR forces the variable TVAR to have a data type of character for this instruction. The SC is a scaling function but can be used to substring character data. The resulting value would be "123".

2. Repeat the data element and map the data element using the literal `&E(TEMPVAR / 100)`.

   The resulting value would be 1.23.

An alternative is as follows:

1. Map a data element using the literal **&USE TVAR,1,3**.

   The resulting value would be 123.

Control data literals

Audit and control are generally high priority items. If you have the need to add the internal trading partner nickname of your trading partner to your application data for each transaction, use the &TPID or &TPNICKN literals. If you need to include the data format ID associated with the current transaction, use the &FORMAT literal. Finally, if you require the application control value associated with the current transaction, use the &ACFIELD literal. This value will only be correct after ALL the fields that comprise the application control field have been processed.

Mapping specific service segment fields (receive only)

Table 48 lists the literals that are provided so that every field within every service segment can be accessed using a combination of the segment ID (ISA, UNB, STX, and so forth) concatenated with a 2-byte number indicating the field within the segment wanted. The names created with this concatenation match the names of the fields defined in the E, I, T, U, and X profiles.

Invalid names (for example, ISA44) are not flagged as errors, but return no data. Using names that do not match the envelope type being received (for example, using ISA01 when UN/EDIFACT service segments (UNB) are being used) is not an error, but no data is returned.

Table 48. Literals to identify fields in service segments

Literal:	nn Value:	Segment:
&ISA*nn*	01 through 16	ISA
&GS*nn*	01 through 08	GS
&ST*nn*	01 through 02	ST
&SE*nn*	01 through 02	SE
&GE*nn*	01 through 02	GE
&IEA*nn*	01 through 02	IEA
&UNB*nn*	01 through 18	UNB
&UNG*nn*	01 through 13	UNG
&UNH*nn*	01 through 09	UNH
&UNT*nn*	01 through 02	UNT
&UNE*nn*	01 through 02	UNE
&UNZ*nn*	01 through 02	UNZ
&STX*nn*	01 through 12	STX
&BAT*nn*	01 through 01	BAT
&MHD*nn*	01 through 06	MHD
&MTR*nn*	01 through 01	MTR
&EOB*nn*	01 through 01	EOB
&END*nn*	01 through 01	END
&BG*nn*	01 through 07	BG
&EG*nn*	01 through 04	EG
&ICS*nn*	01 through 10	ICS
&ICE*nn*	01 through 02	ICE

Mapping generic service segment fields (receive only)

You can map received envelope data to application fields by using substitution keywords in the **Literal** field. The keywords indicate which service segment field is mapped to the application field.

Table 49 describes the substitution keywords you can use to map service segment fields. The Envelope Data Type column indicates the required data type for the service segment field. The EDI Standard Data Type indicates the data type that WebSphere Data Interchange uses for conversions from an EDI standard data type to the application data type.

Table 49. Keywords for mapping envelope data (receive only)

Keyword:	Envelope Data Type:	EDI Standard Data Type:	Envelope Data Mapped to Application:
&I		A	Entire interchange service segment, up to the length of the application field
&ICN	CN or IV	AN	Interchange control number
&IIS	IS, AS, or RS	A	Interchange sender ID
&IIR	IR, AR, or RR	A	Interchange receiver ID
&IDT	DT	DT	Interchange date
&ITM	TM	TM	Interchange time
&IPW	PW	A	Interchange password
&IAP	AP	A	Interchange application reference
&IVR	VR	A	Interchange version/release
&IGT		N0	Interchange total number of groups
&ICT	CT	N0	Interchange control total from the interchange trailer segment
&ITT		N0	Interchange total number of transactions
&G		A	Entire group service segment, up to the length of the application field
&GCN	CN or IV	AN	Group control number
&GFG	FG	A	Functional group ID
&GAS	AS, IS, or RS	A	Group application sender ID
&GAR	AR, IR, or RR	A	Group application receiver ID
&GDT	DT	DT	Group date
&GTM	TM	TM	Group time
&GPW	PW	A	Group password
&GVR	VR	A	Group version
&GLV	LV	A	Group release
&GTT		N0	Group total number of transactions

Mapping service segment fields

Table 49. Keywords for mapping envelope data (receive only) (continued)

Keyword:	Envelope Data Type:	EDI Standard Data Type:	Envelope Data Mapped to Application:
&T		A	Entire transaction service segment, up to the length of the application field
&TCN	CN or IV	AN	Transaction control number. The &TCN keyword is valid for both send and receive. See "Mapping service segment fields (send only)" on page 373 for an explanation on the use of &TCN for application assigned control numbers.
&TTC	TC	A	Transaction code
&TVR	VR	A	Transaction version
&TLV	LV	A	Transaction release
&TTS		N0	Transaction total number of segments

WebSphere Data Interchange interprets any literal beginning with an ampersand (&) as a special keyword. To use a literal that begins with an ampersand, use two ampersands. The translator discards the first one and uses the remaining characters as literal data. For example, if you enter &T in the **Literal** field, the translator moves the entire transaction service segment to the application data. If you enter &&T in the **Literal** field, the translator removes the first & and uses &T as literal data. The service segments (ISA, GS, ST, UNB, UNG, and so on) are not provided in the list of segments that can be mapped for trading partner transactions. Because they are not provided in the list, a direct mapping of a field from a service segment is not possible. In order to use one of the literals from Table 46 on page 341 or Table 49 on page 371, you have to map, or repeat map, some other data element defined in the transaction. When one of the special literals is used, WebSphere Data Interchange knows that the value of the data element being mapped should be ignored, and the value of the special literal should be used instead.

Restrict the mapping of service segment fields to data elements in nonrepeating segments. Select a data element in the first nonrepeating segment that you know will always be present in the received transaction data. The segment *must* be present in the data being received for the mapping instructions to be executed. Repeat the element mapping as many times as is necessary to map (or &SAVE) all of the service segment fields you need. WebSphere Data Interchange recognizes the substitution keywords and moves the value from the associated service segment field rather than the transaction data element that is currently being mapped to the application field or named variable. Data conversions, table translations, and user exits are possible when service segment special literals are used.

Mapping service segment fields (send only)

During the send mapping process, the &TCN special literal may be used to indicate the application field which contains the message or transaction control number. Any field from a segment currently mapped may be chosen and either mapped or repeated, and the special literal &TCN used to identify the application field containing the control number.

- If the application field is part of a record that occurs more than once, the first record is the only one that will be used.
- It is possible to have more than one mapping which contributes to the transaction control number. The data from each mapping will be concatenated with the current data from previous mappings.
- If &DATE is used in the mapping, the format will be `yyyymmdd`. If &TIME is used in the mapping, the format will be `hhmmss`.

The transaction control number is extracted at translate time so if delayed enveloping is used and &DATE or &TIME, or both, are used to construct the control number, the date and time will be the date and time of translation and not the date and time of enveloping.

- Translation tables, validation tables, and user exits can be used during &TCN mappings the same as they can be used in any other mapping.
- The transaction control number generated by the application will be truncated to the maximum length for a control number allowed by the EDI standard, never to exceed fourteen bytes.
- It is possible to combine delayed enveloping with application assignment of control numbers. If this is done, when an envelope operation is requested, the transactions will be sorted such that all transactions for which WebSphere Data Interchange will assign control numbers will occur before the transactions for which the application has assigned control numbers. Also, during an envelope operation, a switch from a transaction that requires WebSphere Data Interchange to assign the control number to a transaction with application assigned control numbers will cause a new interchange to be started. Transactions with application assigned control numbers will be sorted by the value of the transaction control number.

Error message TR0115 will be displayed if the application field containing the control number was not provided, if it contained all blanks, or was otherwise invalid.

Error message TR0116 will be issued if the control number assigned is a duplicate within the group/interchange. The translator requires that message control numbers must be unique within the group. If groups are not being used, then message control numbers must be unique within the interchange.

These errors are considered level 3 errors.

You must fix the application so that the message control numbers are unique within the interchange or group.

The following table shows the type of validation done for every possible mapping between application data types (down) and EDI standard data types (across) during a TRANSLATE TO STANDARD processing.

Validation during mapping

Table 50 describes the type of validation used during translation, based on the type of data being validated.

Table 50. Validation used during mapping for different data types

Data type:	Validation:
A	The ALPHANUM validation table shipped with WebSphere Data Interchange, with numeric digits (0 through 9) removed, is used to validate the data.
AN	The ALPHANUM validation table shipped with WebSphere Data Interchange is used to validate the data.
BIN	A generic data type that encompasses the P, L, Z, B, I and H data types.
CH	The CHARSET validation table shipped with WebSphere Data Interchange is used to validate the data.
N	Only digits, leading or trailing sign characters, and leading or trailing blanks are allowed.
R	Only digits, decimal notation, leading or trailing sign characters, and leading or trailing blanks are allowed.
DT	Must be a valid date according to the format specified during the mapping process.
TM	Must be a valid time.
-	Validation is done automatically because a data conversion is required.

Table 51 shows the type of validation done for every possible mapping between application data types (down) and EDI standard data types (across) during a TRANSLATE TO STANDARD processing.

Table 51. Type of validation during Translate to Standard

Data type:	A:	AN:	BIN:	N:	R:	DT:	TM:
A	A	R	AN	N	R	DT	TM
AN	A	R	AN	N	R	DT	TM
AC	A	R	AN	N	R	DT	TM
CH	CH	R	CH	N	R	DT	TM
DT	DT	DT	DT	DT	DT	DT	DT
TM	TM	TM	TM	TM	TM	TM	TM
R	R	R	R	R	R	DT	TM
N	N	N	N	N	N	DT	TM
P	-	-	-	-	-	DT	TM
L	-	-	-	-	-	DT	TM
Z	-	-	-	-	-	DT	TM
B	-	-	-	-	-	DT	TM
I	-	-	-	-	-	DT	TM
FN	CH	R	CH	CH	CH	CH	CH
H	-	-	-	-	-	DT	TM

Table 52 shows the type of validation done for each possible mapping between application data types (down) and EDI standard data types (across) during TRANSLATE TO APPLICATION processing.

Table 52. Type of validation during Translate to Application

Data type:	A:	AN:	N:	R:	DT:	TM:
A	A	A	N	R	DT	TM
AN	AN	AN	N	R	DT	TM
AC	AN	AN	N	R	DT	TM
CH	CH	CH	N	R	DT	TM
DT	DT	DT	DT	DT	DT	DT
TM	TM	TM	TM	TM	TM	TM
R	R	R	R	R	DT	TM
N	N	N	N	R	DT	TM
P	R	R	N	R	DT	TM
L	R	R	N	R	DT	TM
Z	R	R	N	R	DT	TM
B	R	R	N	R	DT	TM
I	R	R	N	R	DT	TM
FN	CH	CH	CH	CH	CH	CH
H	R	R	N	R	DT	TM

Validation during mapping

Appendix D. Hierarchical loops

A hierarchical loop is similar to an organization chart. Just as an organization chart shows you the various groups of people and their relationship to the whole, a hierarchical loop shows you each group of data and its relationship to the whole. For example, 377 shows an organization chart.
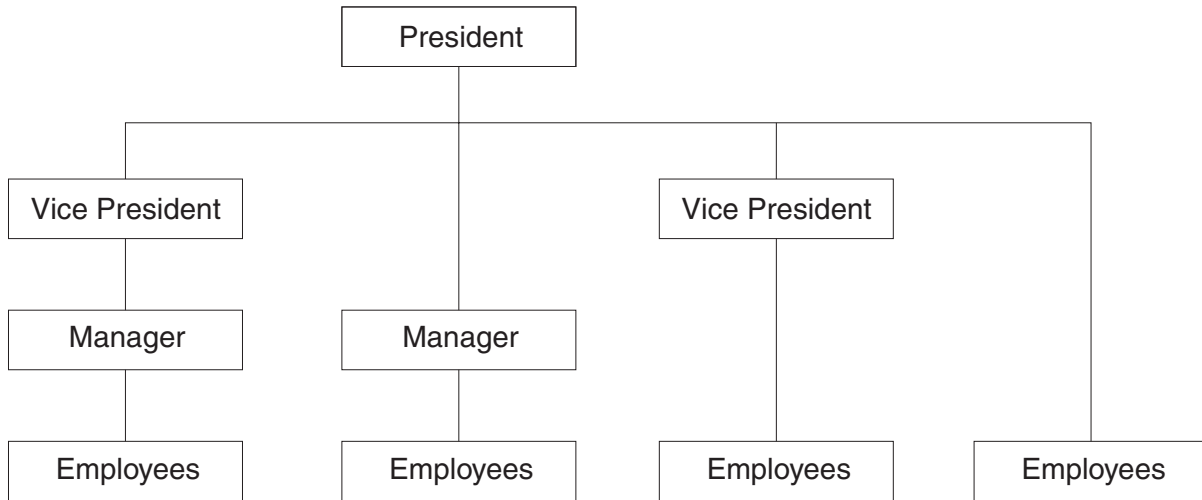
Figure 19. Hierarchical loop example (1)

You can clearly see the different levels in the organization, and who reports to whom.

Hierarchical loops define different levels of data, which can be used in any sequence, and skipped when appropriate, allowing you to fit the loop to your data.
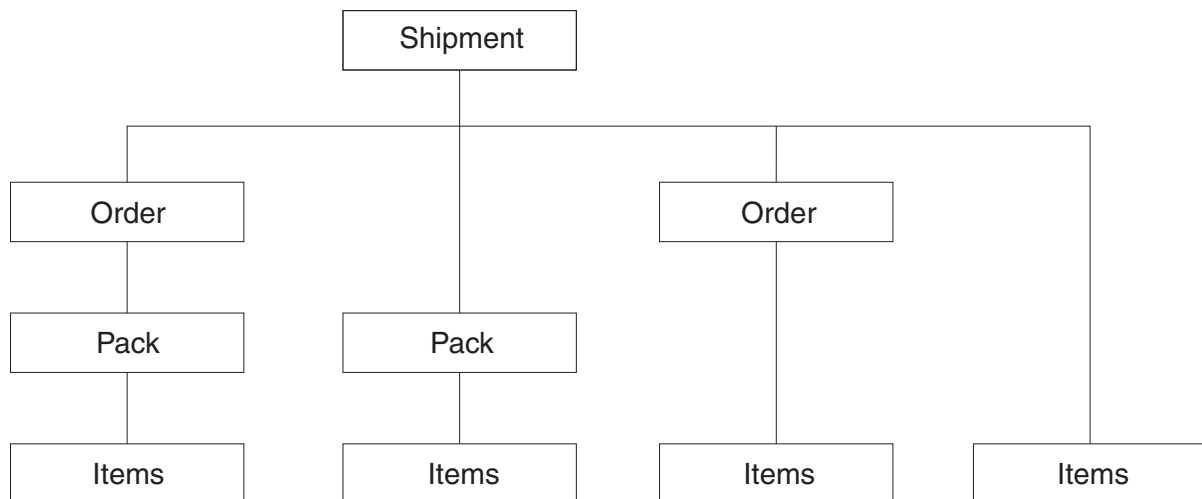
Figure 20. Hierarchical loop example (2)

The HL segment

Sometimes you have to nest loops several levels deep to map all of your data, yet each level contains similar information, such as name and address. In the following figure, loops are nested four levels deep, and the N1 and N3 segments occur at the beginning of each loop.

BSN
DTM   Shipment
N1
N3
TD3   Order
N1
N3
CLD   Pack
N1
N3
PO4
PKG   Item
N1
N3
LIN
SN1|
PID
CTT
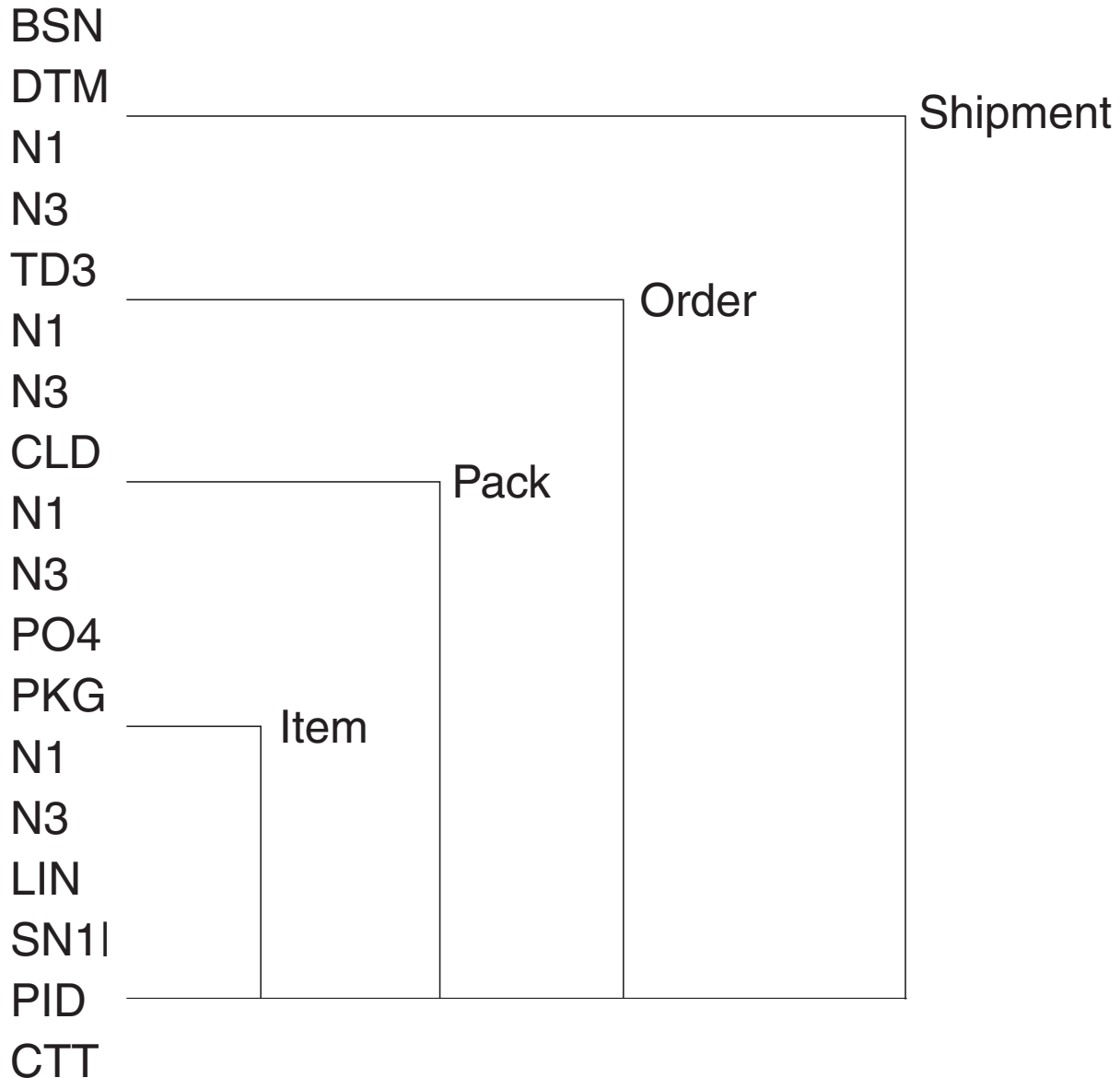
Figure 21. The HL segment

The hierarchical level (HL) segment makes it easier to map these loops. Each HL segment contains information about the relationship of segments in a hierarchical loop to the other segments in the loop. This information is described in Table 53 on page 379.

Table 53. The HL segment

Field ID	Field Name	Description
HL01	ID number	A unique number that identifies the occurrence of the HL segment. This data element is alphanumeric and has a maximum length of 12 characters. This field usually contains a sequential number that is incremented for each occurrence of the HL segment.
HL02	Parent ID	The HL01 value of the HL segment that is the parent of the current HL segment.
HL03	Level Code	A code that indicates the level of the HL segment in the current HL loop. For example, the level code could refer to the shipment, order, or item level information in the ANSI X12 Shipping Notice transaction set.
HL04	Child code	A code that indicates if the segment has subordinate segments: 1 for subordinate segments, or 0 for no subordinate segments. The default is 0.

HL01 and HL02 provide the information for WebSphere Data Interchange to determine the nesting of loops within each other.

Note: The HL segment is not supported by all standards.

Preparing hierarchical loops

WebSphere Data Interchange's HL support allows you to specify unique mapping instructions for each identifiable group of structures in a hierarchical loop. WebSphere Data Interchange can handle 16 levels of nesting within the HL loop structure. To begin mapping a hierarchical level loop, follow these steps:

1. Create the hierarchy for your application data.
2. Assign node IDs to your hierarchy to uniquely identify logically grouped segments in the hierarchical loop. This ensures that WebSphere Data Interchange processes your data the way you want it to be processed. The logical grouping of segments within the HL and the hierarchical relationship is defined by standards organizations or industry groups. Trading partners then agree which segments they will use and how they will be mapped.

The following diagram shows how each group of segments in your hierarchy should be numbered in a top-down, left-right order. Use this number as the node ID value.

Hierarchical loops

Level code
(HI03)
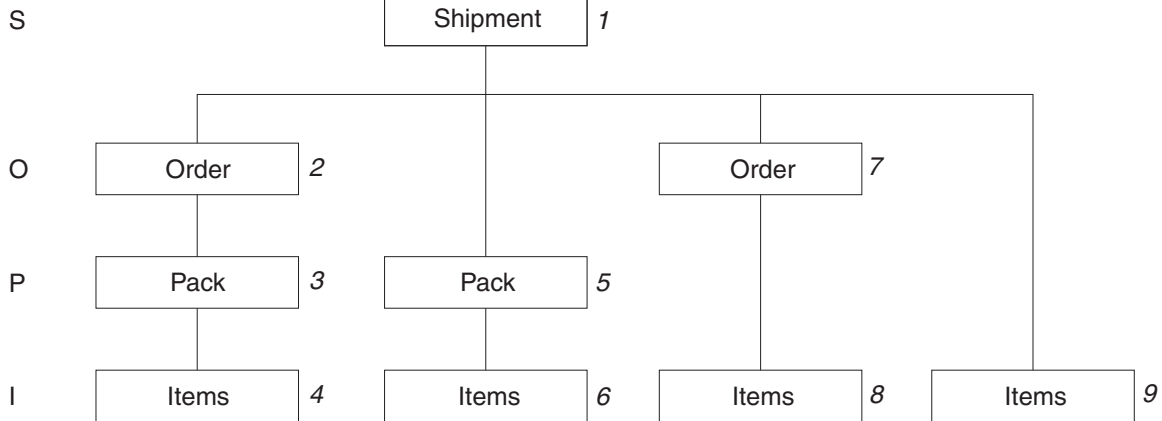
Figure 22. Preparing Hierarchical Loops

1. Create the application data format for this hierarchical loop. The data format and the HL mapping tell WebSphere Data Interchange how to build the hierarchical loop.

   The application data can take one of three basic formats.

   - The application data format can exactly match the hierarchy being created. This optimizes performance during translation because WebSphere Data Interchange does minimal processing to match the data with the hierarchy.

   ```
   SHIPBASE
       SHIPMENT
           ORDER1
               PACK1
                   ITEMS1
               PACK2
                   ITEMS2
           ORDER3
               ITEMS3
               ITEMS4
   ```

   Because a structure can only occur once within a given application data format, numbers to distinguish between the different ORDER, PACK, and ITEMS structures.

   - The application data format can have all the structures defined at the same level. In this case, the order of the structures passed to WebSphere Data Interchange during translation helps determine how the hierarchy is built.

   ```
   SHIPBASE
       SHIPMENT
       ORDER
       PACK
   ITEMS
   ```

   Given the following application data:

   ```
   SHIPMENT
   ORDER
   PACK
   ITEMS
   PACK
   ITEMS
   ORDER
   ITEMS
   ```

```
ITEMS
SHIPMENT
ITEMS

ITEMS
```

The following occurrences of the hierarchy would be built by WebSphere Data Interchange during translation:

```
** occurrence 1 **

SHIPMENT,   ORDER,   PACK,   ITEMS      (based on nodes 1, 2, 3, and 4)
                     PACK,   ITEMS      (based on nodes 5 and 6)
            ORDER,           ITEMS      (based on nodes 7 and 8)

** occurrence 2 **

SHIPMENT,   ITEMS                       (based on nodes 1 and 9)
            ITEMS                       (based on node 9)
```

When data is passed to the translator like this, WebSphere Data Interchange builds a child level of the hierarchy for each structure received in the application data between occurrences of the parent structure or structures. For example, only the ITEMS between the current SHIPMENT and the next SHIPMENT, or end of file, will be used to create this occurrence of the hierarchical loop. Any ITEMS received before the first SHIPMENT will be ignored.

- The application data can supply a structure that contains hierarchical data.

```
SHIPBASE
   HLDATA
       SHIPMENT
       ORDER
       PACK
       ITEMS
```

When data is passed to the translator like this, the fields within HLDATA define exactly how the hierarchy will be built. HLDATA must contain the value for the HL03 element, which contains the hierarchical level code. It can also contain the HL01, HL02, and HL04 elements of the HL segment, but these are not required. If values for these fields are not present in the application data, the HL special literals, which are described in may be used.

To create our sample hierarchy, the application data being passed to WebSphere Data Interchange for translation could look like:

```
                                 where HLDATA contains
                                 HL01  HL02  HL03  HL04
-------------------------------------------------------
HLDATA                           1           S     1
SHIPMENT
HLDATA                           2     1     0     1
ORDER
HLDATA                           3     2     P     1
PACK
HLDATA                           4     3     I     0
ITEMS
HLDATA                           5     1     P     1
PACK
HLDATA                           6     5     I     0
ITEMS
HLDATA                           7     1     0     1
ORDER
HLDATA                           8     7     I     0
```

```
ITEMS
HLDATA                                    9    1    I    0
ITEMS                                     9    1    I    0
```

You can also use combinations of these basic formats. For example, you might want to provide an HLDATA structure for all but the last branch of the hierarchy. The application data format would be:

```
SHIPBASE
    HLDATA
        SHIPMENT
        ORDER
        PACK
        ITEMS
SHIPMENT2
        ITEMS2
```

HL segment Literal keywords for Send and Receive Maps

Your application data may not contain the information the translator needs to create the HL segment, but you can use the following special literals to supply the values for the HL segment with a Send or Receive Map. For Data Transformation Mapping see "Data Transformation Mapping for HL Loops".

&HLID
>Supplies a sequential number for each HL segment created.

&HLPID
>Supplies the HLID value for the parent of the current HL.

&HCODE
>Supplies the hierarchical code associated with the current HL segment.

&HCHILD
>Supplies the value 1 if the current HL segment has subordinate segments.

When you map the HL segment, you can use these literal keywords by typing them in the Literal field of the Map Data Element panel (TP10) when you map the data elements. For send, they will be automatically mapped with their corresponding special literal when the HL loop is qualified. You must remap these elements if you want different mappings.

- HL element 628 will be mapped with &HLID
- HL element 734 will be mapped with &HLPID
- HL element 735 will be mapped with &HCODE
- HL element 736 will be mapped with &HCHILD

Data Transformation Mapping for HL Loops

WebSphere Data Interchange provides special mapping support for the HL loop which allows you to define Hierarchical levels and specify unique mapping instructions for each identifiable group of structures in a hierarchical level (loop). For EDI target messages the HL segment can be automatically created eliminating the need for the application to supply the hierarchical information.

Creating a Data Transformation Map

Source based mapped is required to use special HL mapping support when EDI is the source message.

Target based mapped is required to use special HL mapping support when EDI is the target message.

For information about creating a data transformation map, see Chapter 26, "Data transformation mapping" on page 201.

Defining HL loop levels

To create the base HL level:

1. Go to the mapping details tab, and right-click the HL loop in the command window. The Qualification Selection Window displays.

2. Select **Add HL Qualification**.

3. Select an HL level code from the drop-down list. The list contains valid codes from the code list for HL03 (element 735).

An HLLevel command is created in the command window as follows:

```
HLLevel(char "level code")
HLLevel('S')
```

To create the underlying HL nesting levels or children for this HL level, right-click the HLLevel command and select **HL Qualification**. The ″AddChild HL Qualification″ and ″AddPeer HL Qualification″ commands display.

- To create a child node, select the ″AddChild″ command.

  After selection of AddChild, select an HL level code from the drop-down list. The list contains valid codes from the code list for HL03 (element 735). The HL loop is inserted as a child of the current HL loop with an HLLevel command. The HL segment is automatically selected and all elements mapped with an HLAutoMapped command. This command can not be removed or modified. Additional mapping for these elements can be accomplished using normal mapping methods.

- To create a sibling node, select the ″AddPeer″ command.

  After selection of AddPeer, select an HL level code from the drop-down list. The list contains valid codes from the code list for HL03 (element 735). The HL loop is inserted as a sibling of the current HL loop with an HLLevel command. The HL segment is automatically selected and all elements mapped with an HLAutoMapped command. This command can not be removed or modified. Additional mapping for these elements can be accomplished using normal mapping methods.

With each HL loop level created, the HL loop is copied to its hierarchical locaion (as a child or peer) with an HLLevel command, and displayed in the command window as if the hierarchy were explicitly defined in the standard.

Qualifying HL loop levels in an EDI source message

To qualify the HL level, right click on the HLLevel command and select the Qualify command to select qulaification by occurrence, value, or expression. For multiple occurrence qualification, drag and drop the sourch path from the source window to the target path in the target window or drag the target path from the target window to the HL loop in the mapping command window. This creates a MapTo() command under the HLLevel command in the command window.

Qualifying HL loop levels in an EDI target message

To qualify the HL level, use the drag and drop method or right click on the HLLevel command and select the ″ForEach″ command for multiple occurrence qualification.

Hierarchical loops

This creates a ForEach() command under the HLLevel command in the command window. For additional qualification using occurrence, value, or expression, right click on the ForEach command and select the Qualify command.

If you do not want a ForEach command (for multiple occurrence qualification), occurrence is assumed. For more information, see "CloseOccurrence" on page 308.

To add qualifications using value or expression, right click the HLLevel command, select "Insert Within", and use conditional commands.

For information about commands not described in this section, see Appendix B, "Data Transformation mapping commands and functions" on page 301.

Appendix E. Notices

This information was developed for products and services offered in the United States. IBM may not offer the products, services, or features discussed in this information in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this information. The furnishing of this information does not give you any license to these patents. You can send license inquiries, in writing, to:

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

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

    IBM World Trade Asia Corporation
    Licensing
    2-31 Roppongi 3-chome, Minato-ku
    Tokyo 106, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law: INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the information. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this information at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs

and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

    IBM United Kingdom Laboratories,
    Mail Point 151,
    Hursley Park,
    Winchester,
    Hampshire,
    England
    SO21 2JN.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

The licensed program described in this information and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Programming License Agreement, or any equivalent agreement between us.

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

Trademarks

The following terms are trademarks of the IBM Corporation in the United States or other countries or both:

AIX	CICS	Crossworlds
DB2	DB2 Connect	DB2 Universal Database
IBM	IBM Global Network	OS/390
z/OS	WebSphere	

Lotus Notes is a trademarks of Lotus Development Corporation in the United States, or other countries, or both.

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

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

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

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

Glossary of terms and abbreviations

This glossary defines WebSphere Data Interchange terms and abbreviations used in this book. If you do not find the term you are looking for, see the index or the *IBM Dictionary of Computing*, New York: McGraw-Hill, 1994.

This glossary includes terms and definitions from the *American National Dictionary for Information Systems*, ANSI X3.172-1990, copyright 1990 by the American National Standards Institute. Copies may be ordered from the American National Standards Institute, 11 West 42 Street, New York, New York 10036. Definitions are identified by the symbol (A) after the definition.

A

AAR. Association of American Railroads. Represents the railroad industry in areas such as standards, public relations, and advertising.

acknowledgment. See *functional acknowledgment*, *network acknowledgment*.

action bar. The area at the top of a panel that contains choices currently available in the application that is running. Compare to the *function key area*, which contains actions common to all programs.

Activity Log. A WebSphere Data Interchange Client equivalent of the ACTLOGS profile on the WebSphere Data Interchange Host.

ADF. *See data format*.

ANSI. American National Standards Institute.

ANSI ASC X12. ANSI Accredited Standards Committee X12, which develops and maintains generic standards for business transactions for EDI.

application. A program that processes business information. An application that requests services from WebSphere Data Interchange is an enabled application.

application data. The actual data in an application data file.

application data format. See *data format*.

application default profile. Identifies business applications, such as purchasing and accounts receivable, to WebSphere Data Interchange and sets specific WebSphere Data Interchange processing defaults for an application.

B

base structure. The data structure that contains all the data structures and data fields that define the application data for a single transaction.

binary format (BIN). Representation of a decimal value in which each field must be 2 or 4 bytes long. The sign (+ or -) is in the far left bit of the field, and the number value is in the remaining bits of the field. Positive numbers have a 0 in the sign bit. Negative numbers have a 1 in the sign bit and are in twos complement form.

C

CICS. Customer Information Control System.

CD-ROM. Compact Disk-Read Only Memory; a storage medium for large amounts of data needed external to the personal computer.

client-server. A computing environment in which two or more machines work together to achieve a common task.

CLIST. See Command list.

code list. A table, supplied by WebSphere Data Interchange or defined by the user, that contains all acceptable values for a single data field.

command line. The line at the bottom of the panel that provides an alternate way of requesting services rather than using the Action column of the panel body.

Command list (CLIST). A list of commands and statements designed to perform a specific function for the user.

composite data element. In EDI standards, a group of related subelements, such as the elements that make up a name and address.

compound element. An item in the source or target document that contains child items. Examples are EDI segments and composite data elements, data format records and structures, and XML elements.

Config. The WebSphere Data Interchange Client database that stores the information necessary for running WebSphere Data Interchange Client, including messages, queries, reports, and preferences.

control number. Numbers (or masks used to create numbers) that are used to identify an interchange, group, or EDI transaction.

Glossary

control string. An object compiled from a map, data format, and EDI standard transaction; it contains the instructions used by the translator to translate a document from one format to another.

control structure. The beginning and ending segments (header and trailer) of standard enveloped transmissions.

conversion. The WebSphere Data Interchange Client process of transforming Host Standards, ADFs, and Trading Partner Transactions (TPTs) into WebSphere Data Interchange Client format Standards, Data Formats, and Maps.

Crystal Reports. A product used by WebSphere Data Interchange Client to format reports.

Customer Information Control System (CICS). An IBM licensed program that enables transactions entered at remote terminals to be processed concurrently by user-written application programs.

customize. To alter to suit the needs of a company, such as removing from an EDI standard the segments and data elements that the company does not use.

Customization data. Data not used directly by the translator, such as data formats, EDI standards, and maps.

D

DASD. Direct access storage device.

data dictionary. A file containing the definitions of all the data elements of an EDI standard.

data element. A single item of data in an EDI standard, such as a purchase order number. Corresponds to a data field in a data format.

data element delimiter. A character, such as an asterisk (*), that follows the segment identifier and separates each data element in a segment. See also *element separator* and *segment ID separator*.

data field. A single item of data in a data format, such as a purchase order number. Corresponds to a data element in an EDI standard.

data format. A description of the application data for a particular transaction. A data format is composed of loops, records, data structures, and fields.

data format dictionary. A file that contains data format components.

data format record. A group of logically related fields set up as a record in a data format.

data format structure. A group of related data fields in a data format, such as the fields making up the line item of an invoice. Corresponds to a composite data element in an EDI standard.

DataInterchange/MVS. The IBM DataInterchange product used on the host; pieces include a TSO parameter entry mechanism and a translator. The functionality available in this product is now available in WebSphere Data Interchange for z/OS.

DataInterchange/MVS-CICS. The CICS-based IBM DataInterchange product. The functionality available in this product is now available in WebSphere Data Interchange for z/OS.

data structure. A group of related data fields in a data format, such as the fields making up the line item of an invoice. Corresponds to a segment in a standard.

data transformation map. One of three supported map types. A data transformation map is a set of mapping instructions that describes how to translate data from a source document into a target document. Both the source and target documents can be one of several support document types.

DB2. Database 2, an IBM relational database management system.

ddname. Data definition name.

decimal notation. The character that represents a decimal point in the data.

delimiter. A character that terminates a string of characters, such as the value contained in a data element.

DI Client. WebSphere Data Interchange Client; the Windows-based, client/server interface for WebSphere Data Interchange.

dictionary. See *data dictionary*.

direct access storage device (DASD). A device in which access time is effectively independent of the location of the data.

distribution libraries. Supplied partitioned data sets on tape containing one or more components used to transfer data to a new system.

distribution tape. A magnetic tape that contains the distribution libraries for installing a new system.

DLL. Dynamic Link Library; an executable module that is linked into the main WebSphere Data Interchange executable module.

DLL/VBX. Dynamic Link Library for Visual Basic; a DLL which adheres to the conventions of the Visual Basic programming language.

document.   A business document that is exchanged between two enterprises as part of a business process, such as a purchase order or invoice. A document within WebSphere Data Interchange is singular. For example, it cannot contain multiple purchase orders. A document can also be represented in any syntax. For example, an XML purchase order and an EDI purchase order are both documents.

Document Type Definition (DTD).   A list of all components included in the XML document and their relationship to each other. This defines the structure of an XML document.

domain.   The data structure or group of data structures in a data format to and from which you should restrict the mapping of EDI repeating segments and loops.

drivers.   See *DLL* and *DLL/VBX*.

DTD.   See *Document Type Definition*.

E

EDI.   Electronic data interchange.

EDIA.   Electronic Data Interchange Association.

EDI administrator.   The person responsible for setting up and maintaining WebSphere Data Interchange.

EDI message.   See message.

EDI standard.   The industry-supplied, national, or international formats to which information is converted, allowing different computer systems and applications to interchange information.

EDI transaction.   A single business document, such as an invoice.

EDI transaction set.   A group of logically related data that make up an electronic business document, such as an invoice or purchase order.

EDIFACT.   Electronic Data Interchange for Administration Commerce and Transport. See UN/EDIFACT.

electronic data interchange (EDI).   A method of transmitting business information over a network, between business associates who agree to follow approved national or industry standards in translating and exchanging information.

electronic transmission.   The means by which information is transferred between parties, such as over a public network.

element.   See *data element*.

element separator.   A character that separates the data elements in a segment. See also *data element delimiter*.

encryption.   The encoding and scrambling of data. Data is encrypted by the sender and decrypted by the receiver using a predetermined program and unique electronic key.

event.   An occurrence that is important to a user's computer tasks, such as a software error, sending a transaction, or acknowledging a message.

Extensible Markup Language (XML).   A standard metalanguage for defining markup languages that was derived from, and is a subset of SGML. It is used to represent structured documents and data.

F

field.   See *data field*.

floating segment.   A segment of an EDI standard that may exist in many positions relative to other segments.

forward translation table.   A user-defined table that translates data values that differ between trading partners. For example, if a manufacturer and supplier have different part numbers for the same item, each company can use its own part number and have it converted to the other company's part number during translation. Forward translation tables translate local values to standard values.

functional acknowledgment.   An electronic acknowledgment returned to the sender to indicate acceptance or rejection of EDI transactions.

functional group.   One or more transaction sets of a similar type transmitted from the same location, enclosed by functional group header and trailer segments.

function key.   A key that causes a specified sequence of operations to be performed when it is pressed. Generally used to refer to keys labelled Fn, where n is a number from 1 to 24.

function key area.   Two lines at the bottom of the panel that list the active function keys for the panel.

G

global variable..   A variable that is shared among all instances of all documents within a translation session.

H

header.   A control structure that indicates the start of an electronic transmission.

Glossary

hierarchical loop. A technique for describing the relationship of data entities which are related in a parent/child manner, like a corporate organization chart. Used in mapping to group related data elements and segments such as trading partner address.

HL. *See hierarchical loop.*

I

IBM Global Network. The IBM communications network that provides products and services to IBM customers.

ICS. International Control Segments.

import. The process of taking WebSphere Data Interchange objects exported on another WebSphere Data Interchange system and incorporating them into the receiving system.

International Control Segments.

Interactive System Productivity Facility (ISPF). An IBM-licensed program that serves as a full-screen editor and dialog manager.

Information Exchange. A commerce engine of IBM Interchange Services for e-business that permits users to send and receive information electronically.

interchange. The exchange of information between trading partners.

ISPF. Interactive System Productivity Facility.

J

JCL. Job Control Language.

K

key. In a profile member, the field that identifies the member. For example, the key for members of the trading partner profile is the trading partner nickname.

L

Link Pack Area (LPA). In z/OS, an area of main storage containing reenterable routines from system libraries. Their presence in main storage saves loading time.

literal. In mapping, a value that is constant for each occurrence of the translation. If you provide the literal value during mapping, the translator does not have to refer repeatedly to the source to obtain the value.

local variable. A variable that is specific to the instance of the document in which it is being used.

log file. A file in which events are recorded.

logging. The recording of events in time sequence.

loop. A repeating group of related segments in a transaction set or a repeating group of related records and loops in a data format.

loop ID. A unique code identifying a loop and the number of times the group can be repeated.

loop repeat. A number indicating the maximum number of times a loop can be used in a transaction set.

LPA. Link pack area.

M

mailbox. If you use a mail type protocol to exchange messages with your trading partners, you will have one or more registered mailboxes. The mailbox profile is used in WebSphere Data Interchange to define your mailboxes and any associated preferences.

map. A set of instructions that indicate to WebSphere Data Interchange how to translate data from one format to another.

map rule. An association between a data transformation map and a trading partner.

maximum use. A number indicating the maximum number of times a segment can be used in a transaction set or the maximum number of times that a data format loop or record can repeat.

member. A collection of data for one entry in a profile. For example, a member of the trading partner profile contains data about one trading partner.

message. A free-form, usually short, communication to a trading partner. In UN/EDIFACT standards, a group of logically related data that make up an electronic business document, such as an invoice. A message is equivalent to a document.

message log. The file in which WebSphere Data Interchange Client logs messages about errors that occur within the client.

multiple-occurrence mapping. A form of mapping in which all occurrences of a loop or repeating segment are mapped to the same repeating structure in the data format.

N

network acknowledgment. A response from the network indicating the status of an interchange envelope, such as sent or received.

network commands. The commands that you want WebSphere Data Interchange to pass to your network, defined in the network commands profile. In the host product, this file is named NETOP.

network profile. The WebSphere Data Interchange Client terminology for NETPROF members on WebSphere Data Interchange Host.

O

ODBC. Open Data Base Connectivity. ODBC is an industry standard for making connections between a variety of software products and databases on different hardware platforms.

ODETTE. Organization for Data Exchange through Teletransmission in Europe.

Open Data Base Connectivity. See *ODBC*.

P

panel body. The area in the middle of the panel that contains entry fields, lists of selectable items, menu choices, and scrollable text.

parse. To break down into component parts.

path qualified mapping. A form of mapping in which all occurrences of a repeating compound or simple data element are mapped to a repeating compound or simple data element in another document.

PDF. Program Development Facility.

PDS. Partitioned data set.

PDS members. Groups of related information stored in partitioned data sets.

profile. Descriptive information about trading partners, network connections, and so on. Each profile can contain one or more objects or members. For example, the trading partner profile contains members for your trading partners (one member for trading partner address).

program directory. A document shipped with each release of a product that describes the detailed content of the product.

Q

qualifier. A data element which gives a generic segment or data element a specific meaning. Qualifiers are used in mapping single or multiple occurrences.

quiesce. To end a process by allowing operations to complete normally.

quiescing. The process of bringing a device or a system to a halt by rejection of new requests for work.

R

RACF. Resource access control facility.

receive map. One of three supported map types. A receive map is a set of mapping instructions that describe how to translate an EDI standard transaction into a proprietary application data document.

receive usage. An association between a receive map and a trading partner.

record. A logical grouping of related data structures and fields.

release character. The character that indicates that a separator or delimiter is to be used as text data instead of as a separator or delimiter. The release character must immediately precede the delimiter.

repository data. A group of data definitions, formats, and rules/usages, that WebSphere Data Interchange uses to process your data.

requestor. See *mailbox*.

Resource Access Control Facility (RACF). An IBM-licensed program that provides for access control by identifying and verifying the users to the system, authorizing access to protected resources, logging the detected unauthorized attempts to enter the system, and logging the detected accesses to protected resources.

reverse translation table. A user-defined table that translates data values that differ between trading partners. For example, if a manufacturer and supplier have different part numbers for the same item, each company can use its own part number and have it converted to the other company's part number during translation. Reverse translation tables translate standard values to local values.

rule. See *map rule*.

runtime data. Data used by the WebSphere Data Interchange translator, such as control strings, code lists, translation tables and profiles.

S

SAF. System Authorization Facility.

SAP. (1) A German company named Systeme, Anwendungen, and Produkte specializing in application software. A major product, SAP R/3, is a component-based architecture/application that integrates business processes, such as sales, materials management, and distribution. (2) SAP R/3 supports an

Glossary

EDI interface subsystem. SAPþR/3 generates application data in the SAP R/3 Intermediate Document (IDOC) layout. This data is then sent to the EDI subsystem via a file transfer product, such as FTP or TCP/IP.

security administrator. The person who controls access to business data and program functions.

segment. A group of related data elements. A segment is a single line in a transaction set, beginning with a function identifier and ending with a segment terminator delimiter. The data elements in the segment are separated by data element delimiters.

segment directory. A file containing the format of all segments in an EDI standard.

segment identifier. A unique identifier at the beginning of each segment consisting of two or three alphanumeric characters.

segment ID separator. The character that separates the segment identifier from the data elements in the segment.

segment terminator. The character that marks the end of a segment.

send map. On of three supported map types. A send map is a set of mapping instructions that describe how to translate a proprietary application data document into an EDI standard transaction.

send usage. An association between a send map and a trading partner.

simple element. An item in the source or target document that does not contain child items, only data. Examples are EDI data elements, data format fields, XML attributes, and PCDATA values.

single-occurrence mapping. A form of mapping in which each occurrence of a loop or repeating compound or simple data element in a document is mapped to a different compound or simple data element in another document.

SMP/E. System Modification Program Extended.

source document definition. A description of the document layout that will be used to identify the format of the input document for a translation.

special literal. The send and receive Mapping Data Element Editors include the Literal or Mapping Command field. Literals are constant values you enter in this field, such as 123. Special literals are values you enter in this field that begin with an ampersand (&) and are command to WebSphere Data Interchange, rather than constant values. For example, to use today's date, you enter &DATE.

SQL. Structured query language.

standards. See *EDI standard*.

structure. See *data structure* or *data format structure*.

subelement. In UN/EDIFACT standards, a data element that is part of a composite data element. For example, a data element and its qualifier are subelements of a composite data element.

subelement separator. A character that separates the subelements in a composite data element.

System Modification Program Extended (SMP/E). An IBM-licensed program used to install software and software changes on OS/VS1 and OS/VS2 systems.

T

tag. In UN/EDIFACT standards, the segment identifier. In export/import, a code identifies each field in the export record. Such export/import files are known as "tagged" files.

target document definition. A description of the document layout that will be used to create an output document from a translation.

TD queue. See *transient data queue*.

TDCC. Transportation Data Coordinating Committee.

TDQ. Transient data queue.

temporary storage queue (TS). Storage locations reserved for immediate results in CICS. They are deleted after the task that created them is complete and they are no longer necessary.

Time Sharing Option (TSO). A component of the z/OS operating system that allows users full access to z/OS functionality, but shares machine resources across users.

Time Sharing Option Extensions (TSO/E). The base for all TSO enhancements. It provides z/OS users with additional functions, improved usability, and better performance.

TPT. Trading partner transaction. See *map*.

trading partner profile. The profile that defines your trading partners, including information about network account numbers, user IDs, who pays for network charges, etc.

trading partners. Business associates, such as a manufacturer and a supplier, who agree to exchange information using electronic data interchange.

trading partner transaction. See *map*.

trailer. A control structure that indicates the end of an electronic transmission.

transaction.   A single business document, such as an invoice. See also *EDI transaction*.

transaction set.   A group of standard data segments, in a predefined sequence, needed to provide all of the data required to define a complete transaction, such as an invoice or purchase order. See also *EDI transaction set*.

Transaction Store.   The file that contains the results of translations and a history of translation activity.

transform.   The process of converting a document from one format to another.

transient data queue (TD).   A sequential data set used by the Folder Application Facility in CICS to log system messages.

translation.   The process of converting a document from one format to another.

translation table.   A user-defined table that translates data values that differ between trading partners. For example, if a manufacturer and supplier have different part numbers for the same item, each company can use its own part number and have it converted to the other company's part number during translation.

TSO.   Time Sharing Option.

TSO/E.   Time Sharing Option.

TSQ.   See *temporary storage queue*.

U

UCS.   Uniform Communication Standard.

unary operator.   An operator that changes the sign of a numeric value.

UN/EDIFACT.   United Nations Electronic Data Interchange for Administration Commerce and Transport.

Uniform Communication Standard (UCS).   The EDI standard used in the grocery industry.

UN/TDI.   United Nations Trade Data Interchange.

Usage.   An association between a send or receive map and a trading partner.

V

validation table.   A table, supplied by WebSphere Data Interchange or defined by the user, which contains all acceptable values for a single data field.

variable.   The entity in which a value may be stored based on data received; as opposed to a constant value.

W

WebSphere Data Interchange.   A generic term for the WebSphere Data Interchange products, WebSphere Data Interchange for z/OS and WebSphere Data Interchange for Multiplatforms. WebSphere Data Interchange is a translator of data from one document format to another; the pieces of this product include a TSO parameter entry mechanism, a CICS parameter entry mechanism, a Windows-based parameter entry mechanism (WebSphere Data Interchange Client), and a translator.

WebSphere Data Interchange Client.   A Windows-based product for entry of parameters needed by the WebSphere Data Interchange translator.

WebSphere MQ.   An IBM product that is used to implement messaging and queueing of data groups. Earlier releases of this product were known as MQSeries.

WebSphere MQ queue profile.   Represents a relationship between a logical name and a physical WebSphere MQ queue name.

WINS.   Warehouse Information Network Standard.

Windows®.   Microsoft's graphical operating system under which WebSphere Data Interchange Client runs.

X

X12.   A common EDI standard approved by the American National Standards Institute.

XML.   See *Extensible Markup Language*.

Glossary

Bibliography

This section describes the documentation available for the WebSphere Data Interchange product.

WebSphere Data Interchange publications

The WebSphere Data Interchange publications are:

- *WebSphere Data Interchange for z/OS Administration Guide*, SC34-6214
- *WebSphere Data Interchange User's Guide*, SC34-6215
- *WebSphere Data Interchange Messages and Codes*, SC34-6216
- *WebSphere Data Interchange Programmer's Reference*, SC34-6217
- *WebSphere Data Interchange for z/OS Installation Guide*, SC34-6999

You can order publications from the IBMLink™ Web site at:

`http://www.ibm.com/ibmlink`

In the United States, you can also order publications by dialing 1-800-879-2755.

In Canada, you can order publications by dialing 1-800-IBM-4YOU (1-800-426-4968).

For further information about ordering publications contact your IBM authorized dealer or marketing representative.

Softcopy books

All the WebSphere Data Interchange books are available in softcopy formats.

Portable Document Format (PDF)

The library is supplied as stand-alone PDFs in US English in the `DOC` directory on the product CD. The contents of the `DOC` directory can be viewed without installing the product.

PDF files can be viewed and printed using the Adobe Acrobat Reader. You will need Adobe Acrobat Reader with Search Version 4.05 on Windows NT, or Adobe Acrobat Reader with Search Version 4.5 on UNIX systems.

If you need to obtain the Adobe Acrobat Reader, or would like up-to-date information about the platforms on which the Acrobat Reader is supported, visit the Adobe Systems Inc. web site at:

`http://www.adobe.com/`

If you cut and paste examples of commands from PDF files to a command line for execution, you must check that the content is correct before you press `Enter`. Some characters might be corrupted by local system and font settings.

WebSphere Data Interchange information available on the Internet

The WebSphere Data Interchange product Web site is at:

`http://www.ibm.com/websphere/datainterchange/`

By following links from this Web site you can:

- Obtain latest information about the WebSphere Data Interchange products.
- Access the WebSphere Data Interchange books in PDF format.

WebSphere Data Interchange on the Internet

Index

Numerics

841 transaction set   289

A

accumulators   233
  actions of   235
  adding to maps   234
  types of   234
acknowledgment image   63
acknowledgments
  EDIFACT   143
  functional   143
  interchange   147
  receiving   145
  requesting   143
  sending   143, 144
  UCS 999   143
  X12 997   143
activity log profiles   81
  creating   82
  purpose   81
  setup overview   82
AIX server
  processor license units   30, 32
any-to-any
  data   55
  performing setup for   55
  translation   55
application control key, setting   226
application data
  components   154
    fields   154
    loops   155
    records   155
    structures   155
  obtaining   152
  raw data records   152
  structuring   152
application defaults profile dialog box   72
application defaults profiles   85
  creating   86
  setup overview   85
assignment   307
associated types   72
audits and reports
  acknowledgement image   63
  activity summary   63
  event log   63
  query   64
  status summary   63
  status summary2   63
  transaction details   63
  transaction image   63

B

Between operator   270
BIN segment ID   289
binary segment receive processing   296

C

C&   138
CICS performance profiles   91
  creating   92
  purpose   91
  setup overview   91
client requirements   9
client-server mode. See configuring WebSphere Data
  Interchange Client   13
code list   187
code lists
  creating   194
command chaining   121
commands   308
  CloseOccurrence   308
  conditional   307
  Default   316
  Error   309
  FAError   309
  ForEach   310
  HLLevel   312
  MapCall   312
  MapChain   313
  MapFrom   314
  MapSwitch   314
  MapTo   314
  PERFORM ENVELOPE   139
  PERFORM REENVELOPE   139
  PERFORM SEND   140
  PERFORM TRANSFORM   7, 55, 56
  PERFORM TRANSLATE   139
  Qualify   316
  SetProperty   316
commands, Data Transformation mapping   301
comments   303
conditional commands   307
config database   15
configuration
  recommended   19
    multi-user, multi-server   20
configuration alternatives
  multi-user, client-server   17
  multi-user, stand alone   16
  single user, stand alone   16
configure database connections   12
configuring systems   17
configuring WebSphere Data Interchange Client
  in client-server mode   13
  in stand-alone mode   13
connections to server databases   11
contacts   142

399

Sending your comments to IBM

If you especially like or dislike anything about this book, please use one of the methods listed below to send your comments to IBM.

Feel free to comment on what you regard as specific errors or omissions, and on the accuracy, organization, subject matter, or completeness of this book.

Please limit your comments to the information in this book and the way in which the information is presented.

To make comments about the functions of IBM products or systems, talk to your IBM representative or to your IBM authorized remarketer.

When you send comments to IBM, you grant IBM a nonexclusive right to use or distribute your comments in any way it believes appropriate, without incurring any obligation to you.

You can send your comments to IBM in any of the following ways:
- By mail, to this address:

  User Technologies Department (MP095)
  IBM United Kingdom Laboratories
  Hursley Park
  WINCHESTER,
  Hampshire
  SO21 2JN
  United Kingdom
- By fax:
  - From outside the U.K., after your international access code use 44–1962–842327
  - From within the U.K., use 01962–842327
- Electronically, use the appropriate network ID:
  - IBM Mail Exchange: GBIBM2Q9 at IBMMAIL
  - IBMLink: HURSLEY(IDRCF)
  - Internet: idrcf@hursley.ibm.com

Whichever method you use, ensure that you include:
- The publication title and order number
- The topic to which your comment applies
- Your name and address/telephone number/fax number/network ID.