WebSphere® WebSphere Data Interchange for MultiPlatforms

IBM

Version 3.3

Administration and Security Guide

SC34-6214-01

WebSphere® WebSphere Data Interchange for MultiPlatforms

Version 3.3

Administration and Security Guide

SC34-6214-01

Note!

Before using this information and the product it supports, read the information in “Notices” on page 73.

March 2007

This edition applies to IBM WebSphere Data Interchange for MultiPlatforms, V3.3. and to all subsequent releases and modifications until otherwise indicated in new editions.

To send us your comments about this documentation, email doc-comments@us.ibm.com. We look forward to hearing from you.

Contents

Figures

Tables

About this book

The *WebSphere Data Interchange Security and Administration Guide* provides guidance to electronic data interchange (EDI) administrators, programmers, and support personnel in performing security and administrative tasks for WebSphere® Data Interchange, V3.3.

Changes after publication

The README file on the product CD or tape can contain additional information or changes made after this book was published.

Who should read this book

This book is intended to guide electronic data interchange (EDI) administrators, programmers, and support personnel in performing security and administrative tasks for WebSphere Data Interchange, V3.3. The personnel should be familiar with, or have a working knowledge of:

- Customer Information Control System (CICS®)
- WebSphere Data Interchange, V.3.2 or before
- A programming language such as COBOL, C, or Assembler
- An IBM® relational database management system such as Virtual Storage Access Method (VSAM) or DB2
- Microsoft Windows or XP operating system

How this book is organized

This book contains the following chapters:

- Chapter 1, About WebSphere Data Interchange, which contains system requirements, server installation and setup, and information about using plug-ins and shared configuration databases
- Chapter 2, Client security, which provides detailed descriptions and instructions for using the new security feature
- Chapter 3, RACF security, which has detailed descriptions and instructions for applying security using RACF
- Chapter 4, Event logging, which explains event logging
- Chapter 5, Common event handler, which describes the functions performed by the common events handler,
- Appendix, which has a variety of scenarios of how you might use the security feature in WebSphere Data Interchange

Related books

The following books complete the WebSphere Data Interchange library and contain information related to the topics covered in this book. You can view these documents, and download them, from the library page of the WebSphere Data Interchange Web site:

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

- *WebSphere Data Interchange for MultiPlatforms Quick Start Guide*, CF0YREN

  This document provides a brief overview of how to use WebSphere Data Interchange.
- *WebSphere Data Interchange for MultiPlatforms Administration and Security Guide*, SC34-6214-01

  This document provides information on administrative tasks you will use in WebSphere Data Interchange.
- *WebSphere Data Interchange for MultiPlatforms Messages and Codes Guide* , SC34-6216-01

  This book provides information to assist you in diagnosing errors.
- *WebSphere Data Interchange for MultiPlatforms User's Guide*, SC34-6215-01

  This book provides information on the WebSphere Data Interchange Client/Server user interface.
- *WebSphere Data Interchange for MultiPlatforms Programmer's Reference Guide*, SC34-6217-01

  This document provides detailed technical information about WebSphere Data Interchange.
- *WebSphere Data Interchange for MultiPlatforms Mapping Guide*, SC23-5874-00

  This document provides instructions for your WebSphere Data Interchange mapper.
- *WebSphere Data Interchange for MultiPlatforms Utility Commands and File Formats Reference Guide*, SC23-5873-00

  This document provides the commands and file formats necessary to use WebSphere Data Interchange.
- *WebSphere Data Interchange for z/OS V3.3 Installation Guide*, SC34-6269-01

  This book provides information for the electronic data interchange (EDI) administrator about entering, sending, and receiving EDI transactions and other documents interactively.

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 http://www-306.ibm.com/software/integration/wdi/
- WebSphere Data Interchange training http://www-128.ibm.com/developerworks/websphere/education/enablement/
- WebSphere Data Interchange library http://www-306.ibm.com/software/integration/wdi/library/

- WebSphere Data Interchange Technical Support http://www-306.ibm.com/software/integration/wdi/support/

Chapter 1. About WebSphere Data Interchange

New in this release

V3.3 of WebSphere Data Interchange offers many enhancements to meet the ever-changing world of e-business:

- Enhancements to transaction/document store to store flat-file, XML and globalized data, and related processing and management functionality.
- Migration of reporting function to browser-based interface, and the removal of dependency upon Crystal Reports viewer software.
- Improved error handling and related email based notification of translation errors.
- Extension of management reporting function to data transformation maps.
- Generation of CBE/CEI-compliant event information.
- Improved role and group-based access and management for the WebSphere Data Interchange Client.
- Improved audit trail functionality for modifications to WebSphere Data Interchange objects.
- Support for submitting transaction replay and resend commands from the WebSphere Data Interchange Client.
- Support for ad-hoc and interactive submission of transformation commands from service profiles from within the WebSphere Data Interchange Client.
- Support for multiple, and expanded, data format record IDs, which provides greater flexibility in defining and using records within input data files.
- Support for new data elements in enveloping segments in ANSI X12 and UN/EDIFACT standards.
- Support for the ANSI X12 999 Functional Acknowledgement.
- Support for Java API for service-oriented architecture (SOA) implementations, which allows you to start WebSphere Data Interchange from Java applications and for invoking WebSphere Data Interchange as a Web service in a SOA environment.
- Streaming support for handing large messages, including the addition of steaming support for handling large messages (those greater than 200 MB) by being able to parse data from disk instead of from memory, and by adding support for Pageable Abstract Message Model (PAMM).
- Support for directory triggering of WebSphere Data Interchange server, to complement current WebSphere MQ triggering capability.
- Improved support for database globalization, including enable storage of UTF-8 and UTF-16 characters and Unicode data.
- Support for IBM Support Assistant (ISA), for improved customer technical support.
- Support for IBM Education Assistant (IEA), for improved knowledge base access, peer discussions, and problem resolution.

Getting help

Online Help is available by clicking Help on the upper-right side of each window.

System requirements

The Media Package includes the following materials:

- Quick Start CD
- WebSphere Data Interchange for MultiPlatform V3.3 Server CD
- WebSphere Data Interchange Client V3.3 CD
- DB2 UDB Workgroup Server Edition V8.2
- DB2 Connect Personal Edition V8.2
- WebSphere MQ V6.0

Table 1. WebSphere Data Interchange server system requirements

Software Requirements	Hardware Requirements
• AIX, V5.2, Windows 2000 or Windows XP • IBM DB2 Workgroup Edition, V8.2 with Open Database Connectivity (ODBC) **Note:** Provided with product for use with WebSphere Data Interchange only.	Standard requirements: • Any server capable of running the IBM AIX or Microsoft Windows 2000 or XP operating system • Any Intel® workstation capable of running Windows 2000, Windows XP or AIX, V5.2 • CD-ROM drive for installing the distributed material • Required network hardware and communication connection • Additional requirements for Windows 2000 or XP: – Intel Pentium® III processor at 933MHz, or faster – 1024MB memory – Storage device with a minimum of 8GB available space

WebSphere Data Interchange Client is a 32-bit application designed to run on hardware that meets the following requirements:

Table 2. WebSphere Data Interchange Client requirements

	Software Requirements	Hardware Requirements
Windows or Windows XP **Note:** Provided with product for use with WebSphere Data Interchange only.	• 32-bit Windows 2000 or Windows XP operating system with a minimum of 128MB RAM • IBM DB2 Connect Personal Edition, V8.2 **Note:** Provided with product for use with WebSphere Data Interchange only. • Client/server architecture that has Open Database Connectivity (ODBC) to the server • IBM WebSphere MQ, V6.0 **Note:** Provided with product for use with WebSphere Data Interchange only. Complimentary products: • IBM WebSphere Message Broker for MultiPlatform • GXS Expedite Base and AIX for RS/6000®; V4.5 or GXS Expedite Base for Windows, V4.6.2; or GXS Expedite for Windows, V6.2	• Any workstation with an Intel Pentium processor and a storage device with a minimum of 8GB available space

Server installation and setup

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

Instructions for uninstalling the Server are also provided.

Installing the WebSphere Data Interchange server

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

As an alternative, the install can be run in console mode by adding the **–console** switch to the install command.

By default, WebSphere Data Interchange Server is installed to the /opt/IBM/WDIServer/V3.3 directory. You must have at least 150 MB of free space on this file system. User data will take additional space.

1. If you are installing from a CD-ROM, insert the Server CD into the CD-ROM drive.
   a. Change to the directory containing the WDIServerV33.bin executable.
   b. Run the WDIServerV33.bin executable to start the InstallShield Wizard. The Welcome screen opens as the InstallShield Wizard prepares to install the WebSphere Data Interchange server.
2. Click Next. The license agreement opens. Read the information and license terms on the panel.
3. 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.
4. Click Next.
5. The Installer dialog displays a listing of the installation directory and the total size requirement.
6. Click Next.

   Note: If needed the install will expand the file system, provided there is enough space.
7. The InstallShield Wizard will begin copying program files. To stop this process at any point, click Cancel. At the end of the install, the screen displays the successful installation message.
8. Click Finish. The installation of the files is complete.

   The following directories are created within the installation directory:

   ```
   bin
   bind
   ddl
   eif
   include
   license
   mig32
   runtime
   samples
   ```

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

9. At this time, you will need to log out and log back in again. This will reset the environment to include some of the resources needed.

For information about setting the PATH and setting up a TCP/IP alias to enable remote access to AIX databases, see the WebSphere Data Interchange V3.3 readme.txt file.

Notes:

1. There is a script in the samples directory that can be used to set up the WebSphere Data Interchange environment variables. To use this script, enter the command: **. setdienv.sh**.
2. This command can also be run from within your .profile when you login. By using this at the command prompt, this will set the variables in the current environment and will be lost when you log out.

Windows

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

By default, the WebSphere Data Interchange server is installed to the C:\Program Files\IBM\WDIServer\V3.3 directory. You must have at least 150 MB of free space on this drive. Remember, user data will require additional space.

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 WDIServerV33.exe executable.
   c. Run the executable to start the InstallShield Wizard.

   The Welcome screen opens as the InstallShield Wizard prepares to install the WebSphere Data Interchange Server.
2. Click Next. The license agreement opens.
3. 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. If you want to print the license agreement, click Print. If you do not agree to the license agreement, the installer will exit.
4. To continue with the install, Click Next. The Installer dialog box displays the installation directory.
5. If you want to install the Server into another directory than the default, C:\Program Files\IBM\WDIServer\V3.3, change the location by typing a different location or use the Browse button to navigate to the location you prefer.
6. Click Next. The installer dialog then displays a listing of the installation directory and the total size of the install required.
7. Click Next. The InstallShield wizard displays the choices for the install.
8. Click Next.
9. The InstallShield Wizard begins copying program files. To stop this process at any point, click Cancel. When the installation is complete, the screen displays the successful installation message.

10. Click Finish. The installation of the files is complete. The following directories are created within the installation directory:

```
bin
bind
ddl
eif
include
license
mig32
runtime
samples
```

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

Note: For Windows installations, the bin directory for WebSphere Data Interchange V3.3 is added to the end of your current path. This directory must be completely within the first 255 characters of the PATH to allow DB2 to execute any of the Client's new Submission features. If you have another version of WebSphere Data Interchange installed on your system, you need to change your path to make sure that the WebSphere Data Interchange V3.3 bin directory comes before any other versions of WebSphere Data Interchange in your PATH. If other versions of WebSphere Data Interchange executables are found in your path, you may receive database errors such as a -818, since the executables from other versions are not bound to the EDIEC33E database.

Setting up the WebSphere Data Interchange database

There are new features in WebSphere Data Interchange V3.3 that require changes to the DB2 instance used for the WebSphere Data Interchange database. The user environment from which the DB2 database starts must contain information that enables the database to access WebSphere Data Interchange. 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 Line Tools —> Command Window** to open the DB2 Command window. If you are installing on AIX, log in as a DB2 user with administrator authority.

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

   Note: DB2 and WebSphere Data Interchange work best when installed on the same drive.
2. Change to the ddl directory under the installation directory.
   a. Issue the following command: db2 –tf createdb.ddl > createdb.out

   When this process has successfully completed, the database has been built. Review the log file named createdb.out. It should look like:

```
DB20000I  The CREATE DATABASE command completed successfully.
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
```

```
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
DB20000I  The UPDATE DATABASE CONFIGURATION command completed successfully.
```

3. Change to the DB2 directory that contains the bind files for the DB2 utilities:
   - If you are installing on Windows, this is typically `C:\Program Files\IBM\SQLLIB\bnd`.

     For more information, see the section Binding Database Utilities in the appropriate DB2 Quick Beginnings book.
   - If you are installing on AIX , this is typically `/u/<db2_instance>/sqllib/bnd`, where *<db2_instance>* is the userID of the instance owner.

     Issue the following commands:

     ```
     db2 connect to ediec33e
     db2 bind @db2ubind.lst messages bind.msg grant public
     db2 bind @db2cli.lst messages clibind.msg grant public
     db2 connect reset
     ```

     Or:

     From the DB2 installation directory, enter the following command where WDIServerInstalldir is the directory where you installed WebSphere Data Interchange Server V3.3. :

     ```
     \WDIServerInstalldir\ddl\binddbut.bat
     ```

     This batch file will run the aforementioned commands. 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 ediec33.ddl –l ec33.log
      ```

      This creates the tables, indexes, views, stored procedures, and other files necessary to run WebSphere Data Interchange. The resulting log file, named *ec33.log*, is about 264 Kb.

   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 grntec33.ddl –l grntec33.log
      ```

      The resulting log file, named *grntec33.log*, is about 35 Kb.

   Change to the bind directory under the installation directory.
5. Invoke the command to bind the DB2 packages and issue the GRANT statements that are required to grant access to the newly created tables for the Server. The default `in bindgrnt.fi l` 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
   ```

   The resulting log file, named *bind.log*, is about 11 Kb.

6. Change to the eif directory under the installation directory and issue the following command: `loadeif.bat`

   This process will use WDI PERFORM IMPORT to load additional data into the DB2 tables. This includes data such as functional acknowledgment maps, validation maps, a sample data transformation map, and various profiles and tables.
7. To use the remote execution function of the client connecting to a Windows database, the database will need to be stopped and restarted once all the previous setup is complete. This will enable the stored procedures to work correctly. For use of the remote execution function of the Client connecting to an AIX database, you will need to add the *setdienv.sh* to the DB2 profile of the user starting DB2 and then restart DB2.
8. You have now completed the set up of the database.

Note: If you are currently using WebSphere Data Interchange V3.2, and would like to migrate your transaction store, management reporting, and SAP status data from your WebSphere Data Interchange V3.2 database, see the Appendix of this guide. Other data such as maps, profiles, etc. can be migrated from previous versions of WebSphere Data Interchange using normal export and import procedures.

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.

Migrating data from previous releases

If you are currently using an earlier version/release of WebSphere Data Interchange, you can migrate maps, trading partners, profiles, and most other database objects by exporting the objects from your existing WebSphere Data Interchange database, then importing them into your WebSphere Data Interchange V3.3 database. You can export and import individual objects using the WebSphere Data Interchange Client release migration feature, which is explained in the *WebSphere Data Interchange User's Guide*, or use PERFORM EXPORT and PERFORM IMPORT commands on WebSphere Data Interchange Server, as explained in the *WebSphere Data Interchange Commands and File Formats Reference Guide*. Certain operational data including Transaction Store, Management Reporting, and SAP Status data cannot be imported or exported. Special programs are included with WebSphere Data Interchange V3.3 to migrate this data from a V3.2 database to WebSphere Data Interchange V3.3.

Some restrictions on the use of these programs:

- These programs only support the migration from V3.2 databases to V3.3 databases. Migration of Transaction Store, Management Reporting, and SAP status data from V3.1 is not supported.
- These programs only support migration to the same type of operating system. For example, you can use them to migrate from a V3.2 database on Windows to a V3.3 database on Windows, but not from V3.2 on Windows to V3.3 on AIX.
- These programs are intended to REPLACE ALL existing Transaction (Document) Store, Management Reporting, and SAP status information in the new database. Although it is possible modify the scripts to insert records into the V3.3 database without deleting the existing records first, this should only be used in special situations, and is generally not recommended. If the migrated records conflict with records that were generated by V3.3 processing, it can put the database into an invalid state, and later cause problems during translation and/or reporting functions. To avoid this situation, the script to load the data will use the “clean” option to delete all existing records from the V3.3 table before it loads the migrated data.
- There are a number of interdependencies between Transaction Store tables, and also between Management Reporting tables. You should migrate these as a set, and not try to migrate individual tables. Migrating individual tables can put the database into an invalid state, and later cause problems during translation and/or reporting functions.

These migration steps can be done either immediately after you create your V3.3 database and load the default data, or they can be done later after you have done additional setup and testing with V3.3. If you do the migration after you have done additional testing with V3.3, remember that any existing V3.3 Transaction Store, Management Reporting, and SAP status data will be deleted during the migration of the V3.2 data. Other data such as maps, profiles, etc. will not be affected by these migration steps.

Unloading 3.2 tables

V3.2 database is on the SAME machine where V3.3 is installed

To unload the Transaction Store, Management Reporting, and SAP status tables from a V3.2 database that is on the same machine where V3.3 is installed, follow these steps:

1. On the machine where V3.3 is installed (and contains the V3.2 database), open a command window which will allow you to enter DB2 commands. 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 DB2 user with administrator authority. The remaining database setup steps use this command window or login session.
2. Change to the *mig32* directory under the V3.3 install directory. Invoke the command to bind the DB2 package and issue the GRANT statement that is required to run the edimunl migration program. The default in bindmunl.ddl GRANTs access to PUBLIC. You might want to change PUBLIC to specific user IDs or a group of authorized users.

   ```
   db2 -tf bindmunl.ddl
   ```
3. Run the unload32 script to read the V3.2 database tables, and write the data to files. For each table that is unloaded, the number of records is displayed and a separate file with a .mig extension is created.

   ```
   unload32
   ```

Unloading V3.2 tables

V3.2 database is on a DIFFERENT machine than V3.3 is installed

To unload the Transaction Store, Management Reporting, and SAP status tables from a V3.2 database that is NOT on the machine where V3.3 is installed, follow these steps:

1. 1. Copy the contents of the *mig32* directory from the V3.3 machine to the machine that has the V3.2 database.
2. On the machine that contains the V3.2 database, open a command window which will allow you to enter DB2 commands. 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 DB2 user with administrator authority. The remaining database setup steps use this command window or login session.
3. Change to the directory where you copied the *mig32* files. Invoke the command to bind the DB2 package and issue the GRANT statement that is required to run the edimunl migration program. The default in bindmunl.ddl GRANTs access to PUBLIC. You might want to change PUBLIC to specific user IDs or a group of authorized users.

   ```
   db2 -tf bindmunl.ddl
   ```
4. Run the unload32 script to read the V3.2 database tables, and save them to files. For each table that is unloaded, the number of records is displayed and a separate file with a .mig extension is created.

   ```
   unload32
   ```
5. Copy the *.mig* files to the mig32 directory where V3.3 is installed. These files contain binary data, so you should make sure the files are copied as binary, to make sure they do not get corrupted during the copy.

Loading migrated data into V3.3 tables

Once you have successfully unloaded the data from the V3.2 database, you can load them into the V3.3 tables. Remember, that this will replace all Transaction Store, Management Reporting, and SAP status information that currently exists in the V3.3 database. To load the V3.2 data into the V3.3 database, follow these steps:

1. On the machine where V3.3 is installed , open a command window which will allow you to enter DB2 commands. 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 DB2 user with administrator authority. The remaining database setup steps use this command window or login session.
2. Change to the *mig32* directory under the V3.3 install directory. Invoke the command to bind the DB2 package and issue the GRANT statement that is required to run the edimlod migration program. The default in bindmlod.ddl GRANTs access to PUBLIC. You might want to change PUBLIC to specific user IDs or a group of authorized users.

   ```
   db2 -tf bindmlod.ddl
   ```
3. Run the load33 script to run the migration program edimlod for each table. This will delete existing entries from the V3.3 database tables (Transaction Store, Management Reporting, and SAP status), and insert the records from the files created during the unload process. For each table that is loaded, the number of records inserted is displayed.

   `load33` The edimlod program will do any necessary conversions on the V3.2 records so it can insert the data into the V3.3 database.

Notes:

1. Since the V3.2 database did not support Unicode, data is assumed to be in the system default codepage. Values are converted to Unicode during the load process as required.
2. The sender and receiver qualifier values are included in the Transaction Store tables for V3.3, but were not saved in V3.2. For records that are migrated from V3.2, these will be set to a value of "*". If you use the qualifiers as part of your filter criteria for queries and extracts, you will need to specify an "*" to see the migrated data.

   Or, you can omit the qualifier from your filter criteria. When server updates the Transaction Store status to reflect received functional acknowledgements or network communications, it has special logic to check for entries with the "*" qualifiers if it is unable to find the Transaction Store entries that match the specific qualifiers in the data.

Uninstalling the WebSphere Data Interchange server

The WebSphere Data Interchange server also provides programs that help you to uninstall the product on AIX and Windows.

AIX

To uninstall the WebSphere Data Interchange server on AIX:

1. Change to the `_uninst` directory under the installation directory.

2. Issue the command:
   ```
   uninstall.bin
   ```
3. Once completed, logout and log back in to clear all files.

Windows

To uninstall the WebSphere Data Interchange server on Windows :

1. Close all windows related to the WebSphere Data Interchange Server prior to attempting to uninstall the Server.
2. Click **Start—>Settings—>Control Panel**.
3. Select Add/Remove Programs.
4. Select the WebSphere Data Interchange server product entry and click Remove.
5. Reboot your machine to clear all files.

If there are any issues with the uninstall check the log file in the installation directory.

Starting the server

WebSphere Data Interchange server always reads commands from STDIN and writes the results to STDOUT–these are treated as streams. When invoked from the command line, the command line processor automatically opens the STDIN and STDOUT, piping them wherever the user requests. You typically prepare a file of PERFORM commands for input and redirect the input from that file. You would probably redirect the STDOUT to a file.

An example of this usage is:

```
ediservr<commands.txt>results.txt
```

Where *commands.txt* is the input file and has PERFORM commands, and *results.txt* contains the output from *ediservr.exe*.

The command files consist of a set of WebSphere Data Interchange commands separated by semicolons. Every command is terminated with a semicolon.

The first command is always SET, and the second command is always INIT. These are followed by a series of SET FILE commands that specify the input and output files for the following PERFORM commands. The SET FILE PERFORM sequence can be repeated as many times as necessary.

A typical command file would look like this:

```
SET plan(EDIEC33E) userid(xxxxxx) password(xxxxxx);
INIT;
SET FILE(APPFILE,c:\ne62\test\receive\input\appfile.txt);
SET FILE(EDIFILE,c:\ne62\test\receive\input\edifile.txt);
SET FILE(PRTFILE,c:\ne62\test\receive\input\prtfile.txt);
SET FILE(EXPFILE,c:\ne62\test\receive\input\expfile.txt);
SET FILE(TRKFILE,c:\ne62\test\receive\input\trkfile.txt);
SET FILE(ENVEXCP,c:\ne62\test\receive\input\excpfile.txt);
```

```
SET FILE(RPTFILE,c:\ne62\test\receive\input\rptfile.txt);
SET FILE(FAKFILE,c:\ne62\test\receive\input\fakfile.txt);
SET FILE(QRYFILE,c:\ne62\test\receive\input\qryfile.txt);
PERFORM DEENVELOPE AND TRANSLATE WHERE FILEID(EDIFILE)
    APPFILE(APPFILE) RAWDATA(Y); TERM;
```

Table 3. Server command descriptions

Command	Description
`commands.txt`	
SET	Sets up the environment • plan(EDIEC33E): To point to theWebSphere Data Interchange • databaseuserid(xxxxxx): User ID for the database • password(xxxxxx):
INIT	Loads the startup information and connect to the database using the parameters defined in the SET command.
SET FILE *(LogicalFileName,RealFileName)*	Defines various INPUT and OUTPUT files needed for translation. The LogicalFileName of each file is assigned with a RealFileName that includes the complete path. Depending on the type of PERFORM command used, some files are mandatory, while others are optional.
PERFORM	Issues standard perform commands. This command uses LogicalFileNames for various files used.
TERM	Disconnects from the database and frees all allocated memory.
results.txt (STDOUT)	If translation is successful, then contents of file will be: • DI Translator Started, build Date MMDDYYYY • DI Translator processed your request • DI Translator shut down If translation is not successful, then contents of file will be: • DI Translator Started, build Date MMDDYYYY • DI Translator Error.RC="*errorcode*", ERC="*extended return code*" • DI Translator shut down

Triggering from a message queue

The message queue adapter program is installed as part of WebSphere Data Interchange. The configuration scripts provided set up the necessary queues and definition objects. The adapter uses WebSphere MQ or other message triggering to know when messages need processing.

When a message is put into an application queue, a trigger message is created. The trigger monitor receives the message and executes the adapter. The adapter then passes the information needed to process the application message to the WebSphere Data Interchange server or translator. Application messages are committed, rolled back, or moved to the queue manager's dead letter queue, as determined by the return codes from the WebSphere Data Interchange server. The adapter will wait the time interval you set during setup for any successive messages, and then terminate. The trigger monitor then restarts WebSphere Data Interchange adapter upon receipt of another trigger message.

Base message support architecture uses six message queues:

Table 4. Message queues

Input queues	Output queues
EDI_IN	**EDI_OUT**
ADF_IN	**ADFF_OUT**
XML_IN	**XML_OUT**

The wdi.commands file in the samples directory contains all MQ Service Command (MQSC) and other messaging instructions for creating the needed queues.

Note: Necessary queues file and definition objects are created in the default queue manager.

To create these six queues and configure the input queues for triggering, run the wdicommand script. The specific configuration scripts for integrating with other WebSphere products are available at http://www.ibm.com/websphere/datainterchange/.

Each step of the trigger program is coordinated with a message exits dll called msgExits that must reside somewhere in the binaries path at runtime.

The message exits dll can instruct the trigger program to skip messages, terminate, take or skip a syncpoint, and so on. It can be used to customize the behavior of the adapter to route failed messages to a special queue, or to notify someone if a failure in translation occurs.

When triggered, the adapter follows this sequence:

1. Reads the wdi.properties file for runtime directories.
2. Calls the trigger startup exit msgTrigger( ) if present, and proceeds based on the return code from the exit.

3. Initializes WebSphere Data Interchange. If WebSphere Data Interchange does not initialize, the adapter turns triggering off for the queue and terminates.
4. Sets the name of the file that the message will be received into, which is *datadirectory*(from propertyfiles/*rcvdirectory*(from property files)/*MQSeries messageID*(from MQMD).rcv.
5. For each message on the queue:
   a. Browses the data queue to get the information about the next available message.
   b. Calls the message tracking exit if present, and passes it to browse the data. The message exit can return the batch ID to be used and an indicator of whether to proceed or to skip this message.
   c. If OK to proceed, calls WebSphere Data Interchange with a PERFORM RECEIVE AND PROCESS ONEMESG(Y) WHERE REQID(mq_queue_name[1-16]) BATCHSET(batchid).
   d. Upon returning from WebSphere Data Interchange, calls the msgTransform( ) exit with the return codes, If the return code from the exit instructs the trigger program not to proceed normally, then do what the return code is documented to mean in the adapter user exits, otherwise complete the following process:
      - If translation is acceptable (rc=0), then execute a syncpoint.
      - If translation is not acceptable (r<>0), the adapter posts the message to the dead letter queue defined within the message system or MQSeries. Then execute a syncpoint.
   e. Moves on to the next message (restarts the process at step a).
6. When no more messages arrive within the specified interval, call the msgTerminate user exit, if one exists. If the exit exists, proceed with termination, then terminate WebSphere Data Interchange, and then the adapter itself.

Adapter user exits

To modify or monitor the behavior of the adapter, you can implement the adapter user exits. The WebSphere Data Interchange adapter loads the library, if found in the bin directory, and calls the exit functions. The shared library must be named msgExits.dll and should be compiled using the native compiler for the target platform. For example, Microsoft Visual C++ for Windows.

msgExits.dll interface:

- **bool msgTrigger(const char*pszTriggerMessage, void*pvExitContext);**–Called when the trigger program is started. Passes the trigger message TQTMC2. Accepts a context that will be passed into all subsequent calls. The return value indicates whether to continue or terminate.
- **bool bSkip msgArrival(void*pbExitContext,char*pszSessionID)**–Message tracking exit that will be called just before attempting to get the next message. It can browse the queue for any information required, and then pass back a session ID for WebSphere Data Interchange to use as the batch ID. The return value indicates whether to process the message or skip it.
- **bool bProceed msgTransform(void*ExitContext,long rc,long ccbrc, long ccberc)**–Results of the transformation. Return values from msgTransform:
  - SYNC_CONTINUE–syncpoint, then continue
  - SYNC_TERM–syncpoint, then terminate
  - CONTINUE–do not syncpoint, but continue
  - TERMINATE–terminate without taking a syncpoint
- **bool bOK msgTerminate(void*pvExitContext)**–Okay to terminate. Return values from msgTerminate:
  - #define SYNC_CONTINUE 0x0000
  - #define SYNC_TERM 0x0001
  - #define CONTINUE 0x0002
  - #define TERMINATE 0x0003

A configuration file is installed in the `wdi/bin` directory: `wdi.properties`

The WebSphere Data Interchange server runtime information is stored in a properties file located in the `wdi/bin` directory. The installation default values for windows are listed below. The default values for AIX are similar.

```
runtimedirectory=C:\Program Files\IBM\wdi\bin
datadirectory=C:\Program Files\IBM\wdi\RunTime
dtddirectory=DTD
prtdirectory=PRT
appdirectory=APP
edidirectory=EDI
xmldirectory=XML
aexdirectory=AEX
rptdirectory=RPT
fakdirectory=FAK
```

```
qrydirectory=QRY
xexdirectory=XEX
wrkdirectory=WRK
rcvdirectory=RCV
plan=EDIEC33E
userid=user*
userpassword=password*
Languagecode=ENU
waitinterval=10000
```

Note: *Only required if you want to connect to the database using a different authorization ID than the one the adapter process is running under.

All files created and used during run time are created under the data directory. If you want WebSphere Data Interchange to create files in a different directory, then change the data directory value in wdi.properties to be the new directory you created. This directory can be another drive on Windows or another file system on AIX.

The waitinterval value specifies the number of milliseconds the adapter waits for messages on the Application queue before terminating. Once the adapter terminates, another trigger message restarts the adapter.

Calling from a C++ program

This section provides an example of how to use the WebSphere Data Interchange C++ API. This example includes all the source code necessary to build a C++ program to send several PERFORM commands to the WebSphere Data Interchange product.

Elements of the C++ API

The C++ API is made up of several classes that are all defined in the diapi.h header file shipped with the WebSphere Data Interchange product. The classes that make up the API are:

CSyncTranslator
CASyncTranslator
CRemoteTranslator
CDIEnvironment
CDIRequest

Once a program includes the diapi.h header file, it can use these objects to interact with the WebSphere Data Interchange translator by passing in PERFORM commands to either a CSyncTranslator, CASyncTranslator or CRemoteTranslator object. The three different types of translator objects that can be created are:

- **CSyncTranslator**–Provides access to the translator in a synchronous manner. The process waits for each command to complete before enabling the next command to be performed.

  Method and
  Function

CSyncTranslator(void) (Constructor)
Instantiates a new CSyncTranslator object. This method takes no arguments.

enum eResult Initialize(CDIEnvironment&env)
Causes the translator to be initialized. This method must be called before any transactions can be processed. This method takes a CDIEnvironment object that contains information about the system, such as database information (plan, user ID, password) and system information (language). This method returns an enumerated type that contains success or failure information about the method invocation.

enum eResult Terminate( )
Terminates the translator and causes it to free any memory that was allocated during the translation process. This method takes no arguments and returns an enumerated type with information about the success or failure of the method invocation.

virtual enum eResult ProcessRequest(CDIRequest&req)
Initializes a PERFORM command to be processed by the translator. This method takes a CDIRequest object that has been initialized with a PERFORM command. The enumerated type returned by the function can be used to determine the success or failure of the PERFORM command.

virtual long GetRetCode(void)
Gets the return code from the last translator action performed. This method takes no arguments.

virtual long GetExtRetCode(void)
Accesses the extended return code from the last translator action performed. This method takes no arguments.

- **CASyncTranslator**–Provides asynchronous access to the translator. This method enables your program to begin processing on several transactions at once without waiting for the previous PERFORM command to complete.

Method and
Definition

CASyncTranslator(short sMaxReqs=10, short nNice=0)
Takes two arguments: sMaxReqs and NNice.sMaxReqs specifies the maximum number of requests that can be made. nNice specifies a nice value any process created by the translator during this session.

enum eResult Initialize(CDIEnvironment&Env)
Initializes the translator using the CDIEnvironment argument. The CDIEnvironment object contains the system environment information such as the database plan, user ID, and password, as well as the system language setting.

enum eResult Terminate ( )
Terminates the translator and causes it to free any memory that was allocated during the translation process. This method takes no arguments and returns an enumerated type with information about the success or failure of the method invocation.

enum eResult ProcessRequest(CDI Request& req)
: Passes the perform command contained in the CDIRequest object to the translator to be executed.

short GetMaxRequests( )
: Returns the maximum number of requests that can be executed at one time. This value limits the number of translators that can be started by this object.

short GetCurrentRequests( )
: Returns the number of requests being processed.

enum eResult UpdateCurReq(void)
: Causes the current request count to be updated.

- **CRemoteTranslator**–Provides access to a WebSphere Data Interchange translator running on a remote system. The cRemoteTranslator is like the CAsyncTranslator, except its constructor takes the host name of the room system as an additional argument to its constructor.

Method and
Description

CRemoteTranslator(char*psz,Host,short, sMaxReqs=10)
: Creates a new instance of the CRemoteTranslator class that can communicate with a remote server TCP/IP sockets.

enum eResult Initialize(CDIEnvironment&env)
: Initializes the translator using the CDIEnvironment argument. The CDIEnvironment object contains the system environment information, such as the database information (plan, user ID, password) and system information (language).

enum eResult Terminate( )
: Terminates the translator and causes it to free any memory that was allocated during the translation process. This method takes no arguments and returns an enumerated type with information about the success or failure of the method invocation.

enum eResult ProcessRequest(CDIRequest&req)
: Passes the PERFORM command contained in the CDIRequest object to the translator to be executed.

short GetMaxRequest( )
: Returns the maximum number of requests that can be executed at one time. This value limits the number of translators that can be started by this object.

short GetCurrentRequests( )
: Returns the number of requests being processed.

enum eResult UpdateCurReqCnt(void)
: Causes the current request count to be updated.

- **CDIEnvironment**–Encapsulates all the system settings needed by the CSyncTranslator, CAsyncTranslator, and CRemoteTranslator during their initialization. The CDIEnvironment class must be instantiated and then passed to the initialize method of one of the translator objects.

Method and
Description

void SetSys(char*pszVal)
Identifies the installation-defined Data Interchange for MVS systems used to run the EDIUTILV utility. The default is DIENU.

void SetAppl(char*pszVal)
Identifies the application ID to run the Data Interchange utility. This keyword also identifies the logfile specified by the ACTLOGS profile. If you specify this parameter, the activity log profile must contain a matching entry to define which log file is used for recording errors and events pertaining to the application. The two APPLID values shipped with WebSphere Data Interchange are:

- EDIFFS (default)–Associated with the LOGFFS ddname and the default APPLID and log when using the utilities.
- EDIMP–Associated with the LOGEDI ddname and the APPLID and log used during online Data Interchange processing.

void SetLang(char*pszVal)
Identifies the language profile to use as specified in the LANGPROF profile. The value you specify with the SetLang method must match one of the values in the LANGPROF profile. The LANGPROF that ships with WebSphere Data Interchange is ENU.

void SetPlan(char*pszVal)
Identifies the DB2 plan that WebSphere Data Interchange is to use to access its database tables.

void SetEdiDataQueueName(char*pszVal)
Identifies the routing queue for completion message from the translator. When the CASyncTranslator completes the processing of its EDI data, it will send a completion message to this queue.

void SetAppDataQueueName(char*pszVal)
Identifies the routing queue for completion message for ADF data coming from the translator. When the CAsyncTranslator completes the processing of an ADF, it will send a completion message to this queue.

void SetHostName(char*pszVal)
Sets the host name of a remote WebSphere Data Interchange translator. The value set in this method is only used with the CRemote Translator.

void SetHostPort(char*pszVal)
Identifies the port number of a remote WebSphere Data Interchange translator for network communication. The value set in this method is only used with the CRemote Translator.

void SetUser(char*pszVal)
Sets the database user ID needed to access the DB2 database. This only needs to be set if the userID of the person running the program does not have the necessary authority to access the database.

void SetPassword(char*pszVal)
: Sets the database password needed to access the DB2 database. This only needs to be set if the userID of the person running the program does not have the necessary authority to access the database.

void SetRouterType(enum CDIMsgQueue::qtype enVal)
: Sets the type of routers (queue) for the completion messages. This method can accept the following values:
  - file //File
  - pipe //Named Pipe
  - socket //TCP/IP socket
  - email //Email address

void SetUnitOfWork(enum eUnitOfWork enVal)
: Defines a unit of work to the translator. This method enables the application programmer to define the point at which commits should be done. Possible values for this function are:
  - eTransaction–Commits should be done after every transaction
  - eEnvelope–Commits should be done after every envelope is encountered
  - eNoCommit–No commits are performed.

void SetInterfaceType(enum eInterfaceType enVal)
: Possible values for this function are:
  - eITCmdLine–//Invoked by command line
  - eITApi–//Invoked by API
  - eITWeb–//Invoked by Web server (not supported).

- **CDIRequest**–Represents a request for translation that can be submitted to a translator to be processed. The CDIRequest object names the files necessary to perform a translation as well as the perform statement to be executed. This is a list of the files that can be associated with a CDI request object:
  - Application file
  - EDI file
  - Tracking file
  - Exception file
  - EDI Except file
  - Print file
  - Report file
  - Query file
  - Work file
  - Functional Acknowledgement file

  Method and
  : **Description**

  CDIRequest(void)
  : Constructor for CDIRequest builds an instance of the CDIRequest object. This method does not take any arguments.

void ClearOutput(void)
: Clears all the output fields of the request object.

void SetAppFile(char*pszFile)
: Sets the name of the application file for this request.

void SetEdiFile(char*pszFile)
: Ses the name of the EDI file for this request.

void SetTrackingFile(char*pszFile)
: Sets the name of the tracking file for this request.

void SetExceptionFile(char*pszFile)
: Sets the name of the exception file where ADF can be written if a fatal error is encountered during the processing of the PERFORM command.

void SetPrintFile(char*pszFile)
: Sets the name of the print file where status information about a completed translation can be written.

void SetReportFile(char*pszFile)
: Sets the name of the report file where reports and printouts that you have requested can be stored.

void SetQueryFile(char*pszFile)
: Sets the name of the file where results from a QUERY DATA EXTRACT can be stored.

API Return Codes

When a message is processed, return codes are generated by WebSphere Data Interchange. Return codes are written to an event log that can be queried using a PRINT EVENT LOG command. You can review a single transaction by specifying the transaction ID or review a batch of transactions by specifying the batch ID. You can also query the document store for specific transactions by using a PRINT TRANSACTION DETAILS command and specifying the transaction ID.

Although all successful processing codes are written to the event log, only exception codes for serious errors are written to the event log. You can find additional information about C++ and Java API return codes in the *WebSphere Data Interchange Programmer's Reference Guide*.

WebSphere Data Interchange API Example

This section provides an example program that uses C++ API for WebSphere Data Interchange. This example is made up of one source file that initializes two CDIRequest objects with two PERFORM statements. The sample script provided at the end of this section contains a listing of the main items for this example.

This program begins by creating CSyncTranslator, CDIEnvironment, and two CDIRequest objects (EDI2ADFRequest and ADF2EDIRequest) to handle the two different requests. Next the two requests were initialized with the file names needed to process the requests, and the setPerformCommand method was used to set the required PERFORM commands to execute.

Once the PERFORM commands are created and the translator is initialized, the ProcessRequest method is called to execute the perform. When that action completes, the return codes are checked, the translator is terminated, and the program is exited.

Building the example

This example can be built on different platforms supported by WebSphere Data Interchange.

AIX:

1. Copy the files *apiexamp.cpp* and *apiexamp.mk* from /usr/wdi/DIV3.3/samples to your home directory or any other work directory where you have permissions to write files.
2. Issue the command make -f apiexamp.mk. This will build an executable file.
3. Import the demo maps from *demo850clrecvsuw.eif* and *demo850clsendsuw.eif* using the WebSphere Data Interchange Client.

Windows: On the Microsoft Window platform, you have several choices of compilers. In this example, the Microsoft Visual C++ compiler is used to build the API example:

1. Create an empty win32 console project within Microsoft Visual C++.
2. Add apiexamp.cpp to te project.
3. Build the project you created.
4. Execute the newly built apiexamp executable.

Sample Script:

```
#include
#include "diapi.h"
/*-----------------------------------------------------------*/
/* Main program             */
/*-----------------------------------------------------------*/
void pause(void);

int main ()

{
   CDIEnvironment  aCDIEnvironment;
   CDIRequest       anADF2EDIRequest;
   CDIRequest       anEDI2ADFRequest;
   CSyncTranslator  aCSyncTranslator;
   enum eResult  rc;

   //Define the Data Interchange Environment
   aCDIEnvironment.SetPlan("EDIEC33E");
   aCDIEnvironment.SetLang("ENU  ");

   // Initialize the translator:
   rc = aCSyncTranslator.Initialize(aCDIEnvironment);
   cout << endl << endl << "TestInitialize:  rc=" << rc
   << ", zccbrc=" << aCSyncTranslator.GetRetCode()
   << ", zccberc=" << aCSyncTranslator.GetExtRetCode()
   << endl;

   pause();

   // Name the input and output files for an EDI to ADF request
   anEDI2ADFRequest.SetAppFile("demo850.app");
   anEDI2ADFRequest.SetEdiFile("demo850.edi");
   anEDI2ADFRequest.SetTrackingFile("demo850.trk");
   anEDI2ADFRequest.SetExceptionFile("demo850.aex");

5XQQLQJ_WKH_:HE6SKHUH_'DWD_,QWHUFKDQJH_6HUYHU
anEDI2ADFRequest.SetPrintFile("demo850.prt");

anEDI2ADFRequest.SetFunAckFile("denvtr.fak");

anEDI2ADFRequest.SetWorkFile("denvtr.wrk");

anEDI2ADFRequest.SetReportFile("denvtr.rpt");

anEDI2ADFRequest.SetQueryFile("denvtr.qry");

// Name the input and output files for an ADF to EDI request

anADF2EDIRequest.SetAppFile("denvtr.app");

anADF2EDIRequest.SetEdiFile("denvtr.edi");

anADF2EDIRequest.SetTrackingFile("denvtr.edi");

anADF2EDIRequest.SetExceptionFile("denvtr.aex");

anADF2EDIRequest.SetPrintFile("denvtr.prt");

anADF2EDIRequest.SetFunAckFile("denvtr.fak");

anADF2EDIRequest.SetWorkFile("denvtr.wrk");

anADF2EDIRequest.SetReportFile("denvtr.rpt");

// Set the perform commands to be executed:

 //  ADF-TO-EDI TEST CASE:
********************************************

anADF2EDIRequest.SetPerformCmd

("PERFORM TRANSLATE AND ENVELOPE WHERE APPFILE(APPFILE) "
     "FILEID(EDIFILE) RAWTEST(Y) RAWFMTID(MMTHL7) PURGINT(-1)");

 // EDI-TO-ADF TEST CASES:
********************************************

anEDI2ADFRequest.SetPerformCmd

("PERFORM DEENVELOPE AND TRANSLATE WHERE FILEID(EDIFILE) "

      "APPFILE(APPFILE) RAWDATA(Y) PURGINT(-1)");

// Ask the synchronous translator to process the EDI to ADF Request:
cout << "EDI to ADF translation" << endl;
rc = aCSyncTranslator.ProcessRequest(anEDI2ADFRequest);

// Confirm the input and output names and print the return codes:

cout << endl << endl <<"App File        : "

     << anEDI2ADFRequest.GetAppFile() << ", "

     << anEDI2ADFRequest.GetAppFileLen() << endl;

   cout << "EDI File        : " << anEDI2ADFRequest.GetEdiFile() << ", "
     << anEDI2ADFRequest.GetEdiFileLen() << endl;
cout << "Report File     : "
  << anEDI2ADFRequest.GetReportFile() << ", "
     << anEDI2ADFRequest.GetReportFileLen() << endl;
cout << "Exception File  : "
  << anEDI2ADFRequest.GetExceptionFile() << ", "
     << anEDI2ADFRequest.GetExceptionFileLen() << endl;

cout << "Print File      : "

     << anEDI2ADFRequest.GetPrintFile() << ", "

     << anEDI2ADFRequest.GetPrintFileLen() << endl;

   cout << "Tracking File  : "
   << anEDI2ADFRequest.GetTrackingFile() << ", "
   << anEDI2ADFRequest.GetTrackingFileLen() << endl;

   cout << "Query File  : "
   << anEDI2ADFRequest.GetQueryFile() << ", "
   << anEDI2ADFRequest.GetQueryFileLen() << endl;

   cout << "Work File  : " << anEDI2ADFRequest.GetWorkFile() << ", "
   << anEDI2ADFRequest.GetWorkFileLen() << endl;
   cout << "FunAck File  : "
   << anEDI2ADFRequest.GetFunAckFile() << ", "
   << anEDI2ADFRequest.GetFunAckFileLen() << endl;

   cout << endl << endl << "ProcessRequest:  rc=" << rc
   << ", zccbrc=" << aCSyncTranslator.GetRetCode()
   << ", zccberc=" << aCSyncTranslator.GetExtRetCode();

   cout << "EDI to ADF request complete" << endl;
   pause();

// Ask the synchronous translator to process the ADF to EDI

Request:
   cout << "ADF to EDI translation" << endl;
   rc = aCSyncTranslator.ProcessRequest(anEDI2ADFRequest);

   // Confirm the input and output names and print the return codes:

   cout << endl << endl <<"App File   : "
   << anADF2EDIRequest.GetAppFile() << ", "
   << anADF2EDIRequest.GetAppFileLen() << endl;

   cout << "EDI File        : "
   << anADF2EDIRequest.GetEdiFile() << ", "
   << anADF2EDIRequest.GetEdiFileLen() << endl;

   cout << "Report File  : "
   << anADF2EDIRequest.GetReportFile() << ", "
   << anADF2EDIRequest.GetReportFileLen() << endl;

   cout << "Exception File  : "
   << anADF2EDIRequest.GetExceptionFile() << ", "
   << anADF2EDIRequest.GetExceptionFileLen() << endl;

   cout << "Print File  : "
   << anADF2EDIRequest.GetPrintFile() << ", "
   << anADF2EDIRequest.GetPrintFileLen() << endl;

   cout << "Tracking File  : "
   << anADF2EDIRequest.GetTrackingFile() << ", "
   << anADF2EDIRequest.GetTrackingFileLen() << endl;

   cout << "Query File  : "
   << anADF2EDIRequest.GetQueryFile() << ", "
   << anADF2EDIRequest.GetQueryFileLen() << endl;

   cout << "Work File       : "
   << anADF2EDIRequest.GetWorkFile() << ", "
   << anADF2EDIRequest.GetWorkFileLen() << endl;
```

```
5XQQLQJ_WKH_:HE6SKHUH_'DWD_,QWHUFKDQJH_6HUYHU
cout << "FunAck File     : "
     << anADF2EDIRequest.GetFunAckFile() << ", "
     << anADF2EDIRequest.GetFunAckFileLen() << endl;

cout << endl << endl << "ProcessRequest:  rc=" << rc
     << ", zccbrc=" << aCSyncTranslator.GetRetCode()
     << ", zccberc=" << aCSyncTranslator.GetExtRetCode()
     << endl;

// Terminate the translator:
cout << "Now terminating the translator to free up any resources"

     << endl;
rc = aCSyncTranslator.Terminate();
cout << endl << endl << "Terminate:  rc=" << rc

     << ", zccbrc=" << aCSyncTranslator.GetRetCode()
     << ", zccberc=" << aCSyncTranslator.GetExtRetCode();

// Terminate and go home:
cout << endl << endl;
return(0);

}

void pause()

{
char achar;
cout << "Hit enter to continue" << endl;
cin.get(achar);

}
```

Native Java

WebSphere Data Interchange V3.3 has Java Native Interface implementation for its C++ API. The following are the Java classes.

1. Configuration: `(Native C++ Class CDIEnvironment)`

Purpose: The Configuration class encapsulates all the system settings needed by the Translator class during their initialization. The Configuration class must be instantiated and then passed to the initialize method of Translator objects.

Methods:

`void setSystemID(String ID)`

Identifies the installation-defined WebSphere Data Interchange systems used to run the EDIUTILV utility. The default is DIENU.

`int setAppDefsProf(String Profile)`
Identifies the Application ID to run the Data Interchange utility.

`int setLangProf(String Profile)`
Identifies the language profile to use as specified in the Language (LANGPROF) profile.

`int setPlan(String PlanName)`
Identifies the DB2 plan that WebSphere Data Interchange is to use to access its database tables.

`int setUserID(String ID)`
Sets the database user ID needed to access the DB2 database.

`void setPassword(String pswd)`
Sets the password to be used by the translator to access the WebSphere Data Interchange database.

`int setUnitOfWork(int UOW)`
Defines a unit of work to the translator. This method enables the application programmer to define the point at which Commits should be done. Possible values for this function are 0 (Transaction), 1 (Envelope) and 2 (No Commit)

`int setInterfaceType(int type)`
The possible values are 0 (Invoked by command line), 1 (invoked by API) and 2 (Invoked by Web server)

```
String getSystemID()
String getAppDefsProf()
String getLangProf()
String getPlan()
String getUserID()
String getPassword()
int getUnitOfWork()
int getInterfaceType()
```

Note: See the *WebSphere Data Interchange Programmer's Reference Guide*: Calling from a C++ program under class CDIEnvironment for further description of these functions.

2. Request: `(Native C++ Class CDIRequest)`

Purpose: Request for translation that will be submitted to a translator. The Request object names the files necessary to perform a translation as well as the perform statement to be executed. This is a list of the files that can be associated with a Request object.

Methods:

```
//Functions to set various Files
void SetAppFile(String pszFile);
long GetAppFileLen();
void SetEdiFile(String pszFile);
long GetEdiFileLen();
void SetTrackingFile(String pszFile);
long GetTrackingFileLen();
void SetExceptionFile(String pszFile);
long GetExceptionFileLen();
void SetEdiExceptFile(String pszFile);
long GetEdiExceptFileLen();
void SetPrintFile(String pszFile);
long GetPrintFileLen();
void SetReportFile(String pszFile);
long GetReportFileLen(); SetQueryFile(String pszFile);
long GetQueryFileLen();
void SetWorkFile(String pszFile);
long GetWorkFileLen(); SetFunAckFile(String pszFile);
long GetFunAckFileLen();

//Other Set functions
void SetPerformCmd(String pszCommand);
void SetOptionString(String pszCommand);
void SetTimeStamp();
void SetIRec(boolean bSet);
void SetERec(boolean bSet);
void SetGRec(boolean bSet);
void SetTRec(boolean bSet);
void SetURec(boolean bSet);
void SetGenFuncAcks(boolean bYesNo);
void SetRequestType(int eType);
void SetUserData(String pszData, short sLen);
void SetOptRecs(boolean bI, boolean bE, boolean bG, boolean bT, boolean bU)
void SetUnitOfWork(int enVal);

//Get methods
String GetAppFile();
String GetEdiFile();
String GetTrackingFile();
String GetExceptionFile();
String GetEdiExceptFile();
String GetPrintFile();
String GetReportFile();
String GetQueryFile();
String GetWorkFile();
String GetFunAckFile();
String GetOptionString();
String GetUserData();
boolean GetGenFuncAcks();
int GetRequestType();
```

```
boolean GetIRec();
boolean GetERec();
boolean GetGRec();
boolean GetTRec();
boolean GetURec();
String GetPerformCmd(boolean bMake)
```

(Note: See the *WebSphere Data Interchange Programmer's Reference Guide*: Calling from a C++ program under class CDIRequest for further description of these functions).

3. Translator: `(Native C++ Class CSyncTranslator)`

Purpose: The Translator class provides access to the translator in a synchronous manner. The process waits for each command to complete before enabling the next command to be performed. Methods:

`int Initialize(CDIEnvironment& env)`
: Causes the translator to be initialized. This method must be called before any transactions can be processed. This method takes a Configuration object that contains information about the system, such as database information (plan, user ID, and password) and system information (language). This method returns an enumerated type that contains success or failure information about the method invocation.

`int Terminate ()`
: Terminates the translator and causes it to free any memory that was allocated during the translation process. This method takes no arguments and returns an enumerated type with information about the success or failure of the method invocation.

`int ProcessRequest(CDIRequest& req)`
: Initializes a PERFORM command to be processed by the translator. This method takes a Request object that has been initialized with a perform command. The enumerated type returned by the function can be used to determine the success or failure of the PERFORM command.

`int SetFileName(char* pKey, char* pVal)`
: Set a physical file name to a real file name.

`int GetFileName(char** pRealName, char* pKey, long* lpDeltaLen)`
: Get the real file name of a logical name.

`long GetRetCode(void)`
: Gets the return code from the last translator action performed. This method takes no arguments.

`long GetExtRetCode(void)`
: Accesses the extended return code from the last translator action performed. This method takes no arguments.

Note: See the : Calling from a C++ program under class CSyncTranslator for further description of these functions.

Sample program: run.java

The following program is the Java version of the sample program C++ program provided with WebSphere Data Interchange installation. To compile this program, the CLASSPATH environment variable needs to point to the directory where Request, Configuration and Translator Class files are stored. Also, WDI.DLL should be stored in a directory that is pointed to by PATH environment variable before executing this program.

Sample Java API program

```
package com.ibm.wdi;
public class Run {
  public static void main(String[] args) {
  String file = "tmp";
  Request req = new Request();
  Configuration cfg = new Configuration();
  Translator tr = new Translator();

  String logicalfl = "OUTFILE";
  cfg.setPlan("EDIEC33E");
  cfg.setLangProf("ENG     ");
  tr.initialize(cfg);
  tr.setFileName("XMLFILE", "poxml5sr.dat");
  tr.setFileName("OUTFILE", "sample.out");
  tr.setFileName("FFSEXCP", "sample.aex");
  tr.setFileName("PRTFILE", "sample.prt");

  req.SetPerformCmd("PERFORM TRANSFORM WHERE INFILE(XMLFILE)

OUTFILE(OUTFILE) SYNTAX(X) CLEARFILE(Y) XMLEBCDIC(N) TRACELEVEL(A3)");
  tr.processRequest(req);
  System.out.println("XMLFILE is "+tr.getFileName("XMLFILE")+" and length is
"+tr.getFileLen("XMLFILE"));
  System.out.println("OUTFILE is "+tr.getFileName("OUTFILE"+" and length is
"+tr.getFileLen("OUTFILE"));
  System.out.println("FFSEXCP is "+tr.getFileName("FFSEXCP")+" and length is
"+tr.getFileLen("FFSEXCP"));
  System.out.println("Return code : " + tr.GetRetCode() +"\n"+ "Ext Return Code :
"+tr.GetExtRetCode());

  tr.terminate();     } }
```

Working with the Configuration database

WebSphere Data Interchange stores common information, such as preferences, queries, the Message Log, Systems, and serialized information in the Configuration database. The WebSphere Data Interchange Client connects to the Configuration database when the application is started, and that connection is maintained until you close the Client. The Configuration database is not used by the WebSphere Data Interchange Server.

A default Configuration database is installed on the local machine with WebSphere Data Interchange Client. WebSphere Data Interchange will automatically connect to the default Configuration database without any additional action taken. A shared Configuration database can be created in any database management system that supports ODBC 3.0 or higher.

You can direct the WebSphere Data Interchange Client to access the shared Configuration database by changing the associated settings in configuration options, and then restarting the Client. In addition, if there is an issue with connecting to the Configuration database when the WebSphere Data Interchange Client is started, you are given an opportunity to correct the Configuration database information or select an alternate Configuration database. Configuration database information, such as the ODBC Data Source Name (DSN) and qualifier, are stored in the Windows registry. The information from the registry can be overridden by specifying the Data Source Name and database qualifier as command line parameters when invoking WebSphere Data Interchange Client.

The Configuration database contains three menu options:

- Audit Trail, where you can set up and execute queries for database configuration object types
- Security, where you set up and maintain User ID Definitions and Roles
- Options, where you set up Configuration database connection information, the use of folders in the Configuration database, message logging, and where you configure the performance characteristics of the Audit Trail.

To make changes to the Configuration database:

1. Click **View–>Administration–>Configuration Database–>Options**. The Configuration editor displays.
2. Click the tab where you want to make changes.
3. Complete your changes.
4. Click Save.

WebSphere Data Interchange is installed with a local configuration database. However, it can be run with a shared configuration database. Using a shared configuration database enables multiple users of WebSphere Data Interchange Client to share queries, system definitions, and it can ease maintenance considerations. The following assumptions are made:

- You have the ODBC drivers necessary to access any database used by WebSphere Data Interchange Client.
- You are able to install these ODBC drivers.
- You are able to setup the ODBC Data Source Names (DSN) needed to access any database used by WebSphere Data Interchange Client.

Perform the following steps to setup a shared configuration database:

1. 1. Install the configuration database to a database server, such as DB2, using the CONFIG33.DDL provided with this document.

2. If needed, install the ODBC drivers needed on your machine to access the shared configuration database and those that will be needed to access other WebSphere Data Interchange databases.
3. Create an ODBC DSN that can be used to access the new shared configuration database. Use the *Data Source Administrator* tool found within the Control Panel of most Windows operating systems to define a data source name. Some database tools can automatically establish an ODBC Data Source Name for when you identify the remote database to your system using the database tool. For instance, IBM's *DB2 Configuration Assistant* can be used to identify a remote DB2 database. It will define the ODBC Data Source Name if you select the appropriate option.
4. Follow the standard instructions to install WebSphere Data Interchange Client.
5. Install the latest WebSphere Data Interchange Client Fix Pack.
6. Use WebSphere Data Interchange Client to perform a release migration export of the data in the configuration database.
7. Open preferences in WebSphere Data Interchange Client by clicking **View->Preferences**.
8. In preferences, change the ODBC DSN to the data source name associated with the new shared configuration database. Add a database qualifier for the configuration database, if needed.
9. Click Okay.
10. Close WebSphere Data Interchange Client.
11. Start WebSphere Data Interchange Client. It should open using the new shared configuration database.
12. Use the Release Migration function in WebSphere Data Interchange to import the configuration data you exported in an earlier step.
13. You might want to delete the default system now is the users accessing this shared configuration database will not be using the default database installed with WebSphere Data Interchange Client.
14. Create the ODBC data source names for the other WebSphere Data Interchange databases (such as development, test, and production databases) that will be accessed by users of this shared configuration database.
15. Define the systems in WebSphere Data Interchange Client associated with each database that will be access by the users of this shared configuration database.
16. Delete the default WDIClient33CFG ODBC data source name if necessary. This machine should now be ready to run WebSphere Data Interchange Client.

Perform the following steps for each machine that will have WebSphere Data Interchange Client installed with the shared configuration database:

1. If needed, install the ODBC drivers needed on the user's machine to access the shared configuration database and those that will be needed to access the other WebSphere Data Interchange databases.
2. Create the ODBC data source name (DSN) that will be used to access the configuration database. This name can be any unused data source name, however it is recommended that each user define the same name. It might aid in simplifying support issues.

3. Create the ODBC data source names for the other WebSphere Data Interchange databases (such as development, test, and production databases) that will be accessed by the user of the shared configuration database. These data source names must match those named in the system definitions created during the setup of the shared configuration database.
4. Install WebSphere Data Interchange Client.
5. Start WebSphere Data Interchange Client and then open preferences by clicking **View->Preferences**.
6. In preferences, change the ODBC DSN to the data source name associated with the new shared configuration database. Add a database qualifier for the configuration database, if needed.
7. Click Okay.
8. Close WebSphere Data Interchange Client
9. Start WebSphere Data Interchange Client. It should open using the new shared configuration database.
10. Test access to each of your defined systems.
11. Delete the default WDIClient33CFG ODBC DSN if required. The machine is now ready to run WebSphere Data Interchange Client.

Be mindful of the following considerations:

- Each user utilizing the shared configuration database will have the same systems defined in WebSphere Data Interchange Client. It is important that each database that the user will access be properly defined within ODBC on the user's machine.
- Each user of the shared configuration database will be able to update certain items in the shared configuration database, such as systems, if permitted. Every user will be affected by any change made by a single user. Queries support the concept of *public, protected,* and *private*, so they can be protected if required. The user can only purge entries he has created in the message log.
- Each user will have read access to all messages contained in the message log, regardless of the user that originated the message.

Chapter 2. Client security

Overview

WebSphere Data Interchange V3.3 offers two new security features for the Client: *Role Based Access Control* and *Access Groups*. Both features are optional and can be implemented independent of one another. You can use one without the other, or use them together. Three security related objects make up the two security features: *User IDs*, *Roles*, and *Access Groups*. All of the security objects can be imported between Systems, and exported out of the Configuration database. They might or might not be imported by the Server, depending on how you set up your Server. All security objects are supported in the Configuration database. Role Based Access Control, enables you to:

- Restrict what object types a user can view
- Restrict what functions a user can perform
- Decide which system–any, some, or all–will use Role Based Access
- Use the System Editor or Configuration database options to enable or disable

A *user* is anyone who accesses the WebSphere Data Interchange Client. The *User ID* is the unique system logon identification that you assign to each user. The implementation of a user is in the security facility the customer uses on the system that contains the WebSphere Data Interchange database that will be accessed. On the z/OS platform, this would typically be RACF, as described in the next chapter. Role Based Access Control is a method for administrators to grant one or more users certain access to specific object types. It is never enabled on the WebSphere Data Interchange Client supplied database. You must enable it manually through either the Client or Server. List windows will not appear for objects the user is not authorized to view, and users will not be able to open a functional area if they are not authorized to view any of the objects within the functional area. Additionally, menu actions will be disabled when the user is not authorized to use them for the object.

An *Access Group* is a collection of objects in the WebSphere Data Interchange system. The Access Group is actually a subset of an object type that can be segregated and assigned to an Access Group. For example, an Access Group might contain a specific set of map types for XML or EDI. A user who works exclusively with XML data might not see the EDI maps during a query because the user has not been granted access to the Access Group that contains the EDI maps. You can grant access to one or more Access Groups for each specific user, and users with one or more assigned Access Groups can access objects contained in any of the assigned Access Groups.

Access Group support enables you to:

- Group objects based on your needs–geographic region, industry, or other discriminator
- Restrict which specific objects a user can see in list windows
- Decide which system–any, some, or all–will use Access Groups

Client security

The default database supplied with the WebSphere Data Interchange Client enables a local user to update all objects within the Client-supplied database. Role-based user access is not available within the Client-supplied database–you cannot set up Permissions by user Role in a local setting. However, *security objects* are available to the user in a Client-supplied database. This is provided so the user can create and maintain the various security objects so they can be exported to other systems.

Sometimes, you might want to limit a user's access to only certain objects within a specific object type. For instance, some users are considered *mappers* and they should not be permitted access to the *trading partners* functional area or objects; however, users who are designated as trading partner administrators must have access to the mapping object.

WebSphere Data Interchange's security controls are flexible, yet secure:

- Users can be affiliated with one or more Roles, and given access to individual objects and specific functions within an object.
- Access to specific Permissions will override any Role access granted to the user so that even if a user has permission to work within an Access Group of objects, some of those objects might be secured at a higher permission level. Access to an object can provide a user:
  - Read only access for an object
  - Update access for an object
  - Permission to create a new object
  - Permission to delete or remove objects.
  - Permission to submit objects
  - No access to the object type
- Security administration is performed through a GUI available on the WebSphere Data Interchange Client, and is accessible only by users who have been assigned the WebSphere Data Interchange Administrator Role. If role based access control is not enabled, then security objects are available to everyone. If it has been enabled, then the security objects are available to anyone that has been given access to those objects (it might or might not be the Administrator Role).
- Objects being created or updated might want to provide a list of some related object type to the user. In these cases, the list will always show all related objects.

Implementing security in WebSphere Data Interchange Client

When implementing the security controls in the WebSphere Data Interchange Client, proceed through the implementation in this order:

1. Create the administrator user IDs.
2. Grant the administrators access to all security related objects, Systems, and Configuration database options.
3. Define or update the Roles.
4. Define the Access Groups.

   Note: Access Group default values can also help to assign Access Groups to a large number of objects.
5. Define the user IDs
6. Assign Access Groups to all existing objects.

   Note: Once you have the security controls configured, ***remember to turn on security***.

To implement security for one group of users at a time:

1. Provide all users access to all objects.
2. Restrict selected groups of users as required.

   Note: Users who do not have any Access Groups assigned to them are not restricted by Access Groups. You should use Roles to restrict users performing similar tasks to the needed objects types.
3. Secure WebSphere Data Interchange Client related data according to business requirements as required.
4. Implement Access Groups to restrict users to the specific objects they should be working with.

The following sections describe the security objects in greater detail.

Roles

A Role is a collection of *Permissions* on a set of object types. For example, a mapper might be granted access to maps, standards, data formats. The user might be able to create, read, update, and delete maps and standards, but might only read data formats. Roles:

- Define an area in which one or more users might be assigned.
- Include specific object Permissions and other Roles.

Object Permissions within nested Roles are merged, and the highest access granted is used. Default Roles are provided by WebSphere Data Interchange Client. The default Roles can be altered or deleted as required. Users can fulfill one or more Roles for the product. You define and maintain Roles within the Client. WebSphere Data Interchange provides a set of default Roles. You can also override the Role specified access for a user by indicating levels of access to specific object types.

Client security

The default database supplied with the WebSphere Data Interchange Client enables the user to update all objects within that database. The Configuration database also supports Role-based access when it is defined as a shared Configuration database. The default Configuration database supplied with the Client enables the user to update all objects within that database. Shared versions of the Configuration database can use Role Based Access to determine who can update what object types in the User ID definition.

When setting up WebSphere Data Interchange, identify at least one administrator. Have the administrator maintain:

- System options
- Security objects
- Audit trail
- Shared Configuration database options

You should always ensure someone has authority to update Systems and Configuration database options.

You might create an unlimited number of Roles. Some examples might include:

- Administrator, who manages users and Permissions and Systems
- Configuration, who maintains the Configuration database
- Data Format Administrator, who manages the Data Formats
- Document Store, who handles inquiries for items stored in the Document Store
- EDI Administrator, who manages the EDI standards,
- Mapper, who manages the maps that transform documents
- Operator, who manages all profiles
- SuperUser, who has the highest level of access to the WebSphere Data Interchange system
- Trading Partner Administrator, who manages the trading partner community
- XML Administrator, who handles schemas and DTDs

Roles and Permissions

WebSphere Data Interchange gives you the ability to allow or disallow a user to access to a specific object type or functions for a specific object type. This access is called a permission. Permissions are granted to users and to Roles. Since a Role can include other Roles, and a user can be assigned multiple Roles, conflicting Permissions in Roles will result in the highest specified level of access being used. Permissions assigned directly to a user will override any permission assigned in a Role. There are several levels of Permissions:

- **None**–User has no Permission.
- **Read**–User is permitted to bring up an instance of the object type in a read-only editor, or perform other ″read only″ type of operations, such as export
- **Update**–User is permitted to update instances of the object type
- **Create**–User is permitted to create new instances of the object type

- **Delete**–User is permitted to delete instances of the object type
- **Submit**–User is permitted to use submit based actions on this object type for processing by the Server.

It should be noted that WebSphere Data Interchange Client security only protects the data through the WebSphere Data Interchange Client.

To create User ID Permissions, complete these steps:

1. On the WebSphere Data Interchange tool bar, click **View–>Administration–>Security**.

   This displays the security objects for the System selected on the navigator bar. Each System has its own set of security related objects.
2. Click the User ID Definitions tab.

   Note: To edit an existing User ID Definition, double click on the name of the User ID Definition you want to edit.
3. To create a new User ID Definition, click New. The User ID Definitions editor displays.
4. Complete the fields on the User ID Definitions editor. If you want to know what Permissions are assigned to a User ID object, click Summary on the User ID Definitions editor.

The levels are based on a hierarchy and are as follows:

Level	Access Permissions
None	The user is not given any access to functions associated with the object, except possibly as a list of names from another object the user does have access to. This is the lowest level of access
Read-only	The user can list and view the object. The user cannot update, create or delete the object, or perform other read only type of operations, such as export. The read-only access level is higher in precedence than the None access level, but lower than the update access level.
Update	The user can list, view, and update the specified object. The user cannot create or delete the object. The Update access level is higher in precedence than the read-only access level but lower than the create access level.
Create	The user can list, view, update, and create objects. The user cannot delete objects. The Create access level is the highest access level.
Delete	The user can delete the object. Deleting an object will delete associated objects even if the user has no authority to delete the associated objects. Read, update, or create access must be granted before delete can be granted.
Submit	The user can submit the object for processing by the WebSphere Data Interchange Server. The user cannot view, delete, update, or create the object unless one of those specific access levels is also specified. Read, update, or create access must be granted before submit can be granted.

Assigning and editing Roles and Permissions

To assign Roles in the WebSphere Data Interchange Client, follow these steps:

1. On the WebSphere Data Interchange tool bar, click **View–>Administration–>Security**.

   The Security Functional Area displays.
2. Click the Roles tab.
3. Click New on the Functional Area tool bar. The Roles editor displays.

   Note: To edit an existing permission, double click on the name of the permission you want to edit.
4. Complete the fields on the Roles editor.
5. If you want to assign a sub-Role to the Role you are creating, click the Roles tab and select the Role or Roles you want to add as sub-Roles.
6. Click Save.
7. Click the Permissions tab.
8. If you want to assign specific additional Permissions to this Role, that is, Permissions that are not already included in the sub-Roles, Click New. The Add Permission dialog box displays.
9. Choose the Object type for the new permission. The Permissions dialog box displays and you must choose the level of permission for this object type and Role.
10. Click Insert to save the permission.
11. Repeats steps 9 and 10 for each permission you want to add, then press the cancel button to close the dialog box, or press the OK to enter the last permission.
12. When you are finished, click Save to save the Role.

Access Groups

When Access Groups are enabled, users who do not have any assigned Access Groups will have access to **all** Access Groups. Each object supporting Access Groups can be assigned to one Access Group. User IDs can participate in multiple Access Groups. Any object not assigned to an Access Group is automatically assigned to Access Group *Global* unless another default is specified. The user must also be authorized to the Access Group assigned to the object. By default user IDs can access all groups.

To create Access Groups, complete these steps:

1. On the WebSphere Data Interchange tool bar, click **View–>Administration–>Security**.

   The Security Functional Area displays.
2. Click the Access Groups tab.
3. Click New. The Access Groups editor displays.
4. Enter a name and short description for the Access Group you are creating.
5. Click Save.

Assigning objects to Access Groups

Once you have created a new Access Group within a system, you can assign objects to that Access Group. This feature is used most appropriately by companies whose employees might want to access objects within a specific global or service region, such as North America or the European Union. To assign objects to Access Groups, complete the following steps:

1. Ensure Access Groups are enabled for the system where you have created the Access Group. If this is not already done, complete steps 2 through 6. Otherwise, skip to step 7.
2. From the WebSphere Data Interchange Client Navigator bar, click **View–>Administration–>Systems**.
3. Double-click the name of the system where your Access Group resides.
4. Click on the Access Groups tab.
5. Click on the Allow Use of Security Access Groups check box.
6. Click Save and type any comments as prompted by the Client.
7. Right-click on the object you want to assign to the Access Group. For example, you might want to assign Trading Partner profiles to this Access Group. You would right-click on the Trading Partner profile.
8. Select Object Properties from the pop-up menu.

   Note: You can also do this by highlighting the object you want to change and clicking **Actions–>Object Properties**.
9. Select the name of the Access Group you want to assign this object to from the drop-down menu.
10. Click OK to assign the object to the Access Group.

Server access control by role scenarios

A user assigned to the role of mapper opens the WebSphere Data Interchange client. The user notices that they can access the mapping functional area, XML functional area, EDI standards functional area, and data formats functional area because the functional area buttons for these areas are enabled. All other functional area buttons are disabled. When opening the open functional area dialog box, the user notices that only the functional areas the user is permitted to access are listed. The user opens the mapping functional area and notices that there is no list window for the send maps and receive maps.

After consulting the administrator, the user discovers that their access to these objects is specifically removed because the organization no longer is using those map types. When running a query on the data transformation maps list window, the user sees that only the maps the user is involved with are displayed. This occurs because the maps have been assigned to *groups* and the user can only see maps that are contained in groups assigned to the user.

User ID objects

A generic userid called **&WDIUSER** is predefined in the WebSphere Data Interchange Server database. When WebSphere Data Interchange Client security is enabled (either Access Groups or Role Based Access Control), an attempt is made to locate a User ID object for the user. When it is found, the Roles and Permissions assigned to that User ID object are used to determine access for the user. If a corresponding User ID object is not found, an attempt is made to locate the **&WDIUSER** object. If it is found, then the Roles and Permissions assigned to the **&WDIUSER** object are used for the undefined user. If the **&WDIUSER** object is not found, then the user will not have access to any objects within WebSphere Data Interchange Client. The access levels for the generic userid will be used when the logged on user has not been defined to the System.

A defined user will not inherit privileges from the generic userid. The administrator can update and delete the generic userid, and can modify the generic userid as required. To create or edit a User ID object:

1. On the WebSphere Data Interchange tool bar, click **View–>Administration–>Security**.

   The Security list window displays.
2. Click the User ID Definitions tab.
3. Click New on the Security list window tool bar. The User ID Definitions editor displays.
4. Complete the fields on the User ID Definitions editor tabs: General, Roles, and Permissions.
5. When you have completed the fields on the last tab, click Save to save the User ID Definition.

Chapter 3. RACF security

Overview

Security for WebSphere Data Interchange Server is provided by the Resource Access Control Facility (RACF**) or an equivalent product that is consistent with System Authorization Facility (SAF) interfaces. To protect WebSphere Data Interchange programs and data, use RACF (or an equivalent) and the resource names described later in this appendix. WebSphere Data Interchange provides the level of security that RACF or an equivalent can provide in the MVS/TSO environment. You can control access to the WebSphere Data Interchange systems and files during WebSphere Data Interchange execution. WebSphere Data Interchange provides control over the record-level access through predefined resource names. However, because RACF or an equivalent provides security only to the dataset level, users have access to the entire dataset. If you have installed more than one WebSphere Data Interchange system, have multiple system-related resource names, and have a user that needs to have different authority levels based on the system-related resource names, the only way to guarantee the authority levels is to use a different MVS/TSO user ID for each system-related resource name.

RACF class for WebSphere Data Interchange

To protect WebSphere Data Interchange resources, create a new RACF class called EDIR. Add this class to the class descriptor table (CDT) using the ICHERCDE macro. For details, see your access control facility documentation. For EDIR, specify the macro as follows. Values not shown are chosen locally.

```
ICHERCDE CLASS=EDIR,
  MAXLNTH=39,
  FIRST=ANY,
  OTHER=ANY
```

You must also add class EDIR to the RACF router table using the ICHRFRTB macro with

```
ACTION = RACF:
 ICHRFRTB CLASS=EDIR,
   ACTION=RACF
```

Customers using TopSecret^{TM} instead of RACF should add a RESCLASS, EDIR, to their Resource Descriptor Table (RDT) and manage it as any resource.

Resource names

The following resource names are RACF profile names. Each resource name consists of two or more qualifiers connected by periods. Qualifiers that appear in uppercase are to be used as shown. Those in lowercase are variables. The variable *sys*, for system name, is required only if you have installed more than one copy of WebSphere Data

Interchange on the same system. For example, you would use *sys* to distinguish between test and production versions. You can use two types of system-related resource names:

- **SYSTEM.sys**, where *sys* is the system name for one copy of WebSphere Data Interchange, such as the copy used for testing or the copy used for production. Using system names, you can provide separate protection for each copy of WebSphere Data Interchange. If using a CLIST to start WebSphere Data Interchange, the CLIST must define the system name, which can be up to 8 characters. If a system name is not provided when WebSphere Data Interchange is started, DIENU is used by default.
- **.SYSTEM**, which determines how the system name is used to grant users access to the WebSphere Data Interchange product.
  - If the SYSTEM resource is defined, users must be specifically granted access under the SYSTEM.sys resource name, or they are denied access to the product.
  - If the SYSTEM resource is not defined, users are granted access to the product unless they are specifically excluded under the SYSTEM.sys resource name.
  - For all resource names which have variables at the end of the name, you can customize your access to the resource.

Defining access levels in RACF

You can grant the following access levels to resource names:

- None–Users cannot access the resource.
- Read–Users can view the resource.
- Update–Users can view and update the resource. User cannot copy or create the resource.
- Alter–User can view, update, create, copy, or delete the resource.

Translation and communication

WebSphere Data Interchange provides services (function calls) that are available through an application program interface (API). The Interactive Entry Facility, Transaction Store Facility, and WebSphere Data Interchange Utility also use this interface. The resource names used to protect these functions are:

sys.FUNCTION.TRANSLATE.SEND
Translate for sending function. Only users who have access under this resource can translate documents for sending. In addition, users must have access under the appropriate sys.TRANSACTION.tptname resource, or the translation is not permitted.

sys.FUNCTION.TRANSLATE.RECEIVE
Translate Received Transactions function. Only users who have access under this resource can translate documents that have been received. In addition, users must have access under the appropriate sys.TRANSACTION.tptname resource, or the translation is not permitted.

sys.FUNCTION.SEND
Send Network function. Only users who have access under this resource can

send documents to the network. In addition, users must have access under the appropriate sys.PROF.REQPROF.mbrname and sys.PROF.TPPROF.mbrname, or the send is not permitted. sys.FUNCTION.ENVELOPE Envelope and Deenvelope functions Only users who have access under this resource name can envelope or deenvelope documents.

sys.FUNCTION.CANCEL
Network Cancel function. Only users who have access under this resource can cancel documents from the network or use the RECALL function from the Interactive Entry Facility. In addition, users must have access under the appropriate sys.PROF.REQPROF.mbrname and sys.PROF.TPPROF.mbrname, or the cancel is not permitted.

sys.FUNCTION.RECEIVE
Receive Network function. Only users who have access under this resource can receive documents from the network. In addition, users must have access under the appropriate sys.PROF.REQPROF.mbrname and sys.PROF.TPPROF.mbrname, or the receive is not Allowed.

Resource name verification

All the WebSphere Data Interchange profiles are protected with the following resource name: *sys.PROF.profname.mbrname*

where:

mbrname is the name of a profile member

profname is one of these variable names:

Table 5. Resource variable names

Variable name:	For:
ADAMCTL	User Exits
APPDEFS	Application Defaults
CONTRECV	Continuous Receive Profile
E	EDIFACT standard Envelope Data
I	ICS Standard Envelope Data
LANGPROF	Language Profile
WEBSPHEREMQ	Queue Profile
NETOP	Network Commands
NETPROF	Network Profile
REQPROF	Mailboxes
SECUPROF	Network Security
T	UN/TDI Standard Envelope Data
TPPROF	Trading Partner Profile
U	UCS Standard envelope data

Table 5. Resource variable names (continued)

Variable name:	For:
X	X12 Standard envelope data

Only users with ALTER access to the resource sys.PROF.profname can create or copy profile members within that profile. For example, ALTER authority on *sys.PROF.TPPROF* permits the user to add and copy trading partner profile members.

Only users with ALTER access authority to the resource *sys.PROF.profname* can import profile members into that profile. The user must also have ALTER access authority to resource *sys.PROF.profname.mbrname* for any members that are to be imported. This also applies to profile members imported as associated objects.

Users with ALTER access to the resource *sys.PROF.** can create or copy profile members in any profile unless specifically excluded under the resource name for the specific profile.

Using RACF general resources services

The following steps show an example of defining a RACF profile for the general resource SYSTEM.DIENU and then granting access to the resource.

1. From the RACF–Services Option Menu, select option 2 to add a profile for a general resource.
2. Complete the fields as follows:
   a. Type **1** in the **Option** field to add a profile.
   b. Type EDIR in the **RESOURCE CLASS** field.
   c. Type SYSTEM.DIENU in the **RESOURCE NAME** field.
   d. Press Enter.

      The RACF - Add General Resource Profile panel displays.
3. Press Enter to accept the information supplied on the Add panel.

   The RACF - General Resources Services panel displays again.
4. Select option 4 to grant users access to the resource you just defined.

   The RACF - Maintain General Resource Access List - Add panel displays.
5. Grant ALTER access rights for the resource SYSTEM.DIENU. The owner, USERA, always has access.
6. Enter the IDs of all other users you want to grant ALTER access to this resource.
7. Press Enter to accept the information supplied.

Where resource names are checked

Table 6 lists the DataInterchange resource names and the services that check the user' authorization.

Table 6. DataInterchange resource name verification

Resource Name:	**Checked by:**
sys.PROF.profname sys.PROF.profname.mbrname sys.PROF.profname.mbrname	• Profile Maintenance and Profile Services before access is given to profname.mbrname • Export before exporting member mbrname from profname • Import before importing member mbrname into profname
sys.PROF.TPPROF.mbrname	Communications before actions involving mbrname
sys.PROF.REQPROF.mbrname	Communications before actions involving mbrname
sys.FUNCTION.TRANSLATE.SEND	• Translator before a send translate function • Interactive Entry Facility before sending or translating documents
sys.FUNCTION.TRANSLATE.RECEIVE	• Translator before a receive translate function • Interactive Entry Facility before receiving documents, messages, or files
sys.FUNCTION.ENVELOPE	Translator before an envelope or deenvelope function
sys.FUNCTION.SEND	• Communications before a send function • Interactive Entry Facility before sending documents, messages, or files
sys.FUNCTION.RECEIVE	• Communications before a receive function • Interactive Entry Facility before receiving documents, messages, or files
sys.FUNCTION.EXPORT	Export Utility before a batch export
sys.FUNCTION.IMPORT	Import Utility before a batch import

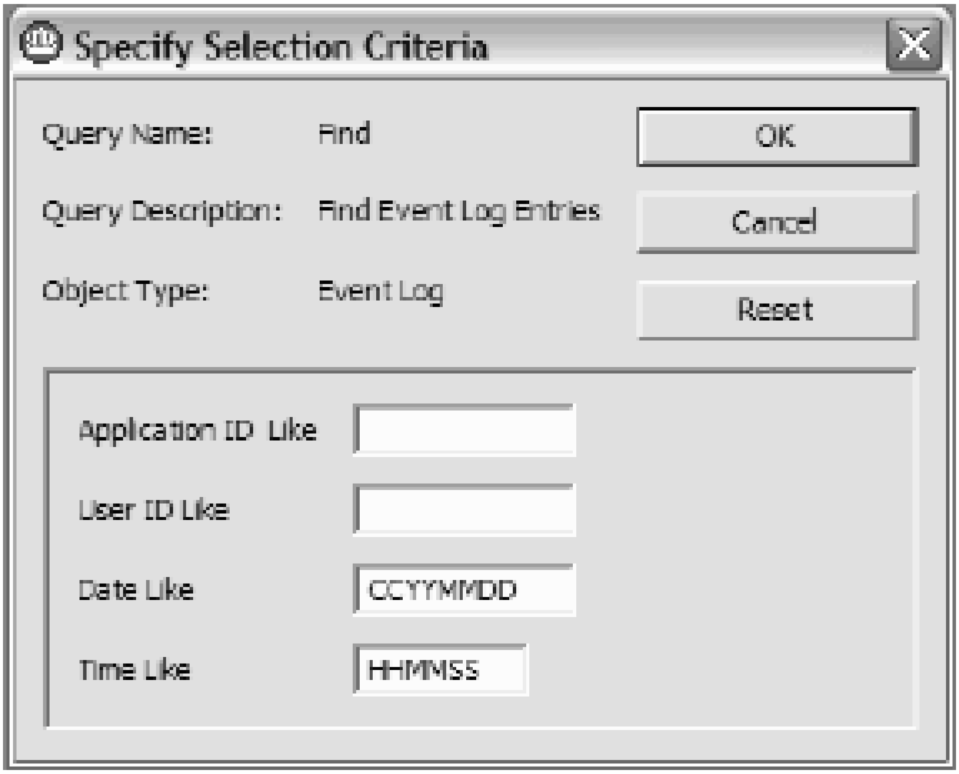
Chapter 4. Event logging

Event Logs are identified using the Application Defaults Profile. The name of the Event Log will be the same as the Application Defaults Profile. When using the WebSphere Data Interchange Utility, the Application Defaults Profile is not specifically identified. Instead, the Application Defaults Profile is used to identify which Application Defaults Profile is associated with a given use of WebSphere Data Interchange. An Application Defaults Profile can be specifically identified whenever the WebSphere Data Interchange Utility is used. If an Application Defaults Profile does not specify an Application Defaults Profile, the associated Application Defaults Profile will default to the same name as the Application Defaults Profile. If that Application Defaults Profile does not exist, the Event Log will be EDIFFS.

An Event Log can be inactive or active at any given time. You control this feature in the Application Defaults Profile. Making an Event Log inactive using the Application Defaults Profile will prevent any message from being written to the Event Log, with one exception (see next paragraph). Setting it as inactive using the Application Defaults Profile will prevent messages from being written to the Event Log when that specific Application Defaults Profile is used. Other Application Defaults Profiles that reference the same Application Defaults Profile will be able to write messages to the Event Log, assuming the active flags enable it. Any error condition or event that changes a transaction's status is recorded in the Event Log even when it is marked as inactive in the Application Defaults Profile. Event Logs do not record events from WebSphere Data Interchange Client; that is, WebSphere Data Interchange Client does not write to an Event Log. However, they can be examined in WebSphere WebSphere Data Interchange Client using the Event Log list window.

When working with Event Logs, you can view or print Event Log entries; turn logging on and off for different events, such as writing EDI Standard Data; add Event Logs for individual applications; and archive and restore log entries.

Display the Event Log list window by clicking **View–>Event Log**. The Specify Selection Criteria dialog box displays, as shown in Figure 1.



Query Name: Find

Query Description: Find Event Log Entries

Object Type: Event Log

Application ID Like

User ID Like

Date Like CCYYMMDD

Time Like HHMMSS

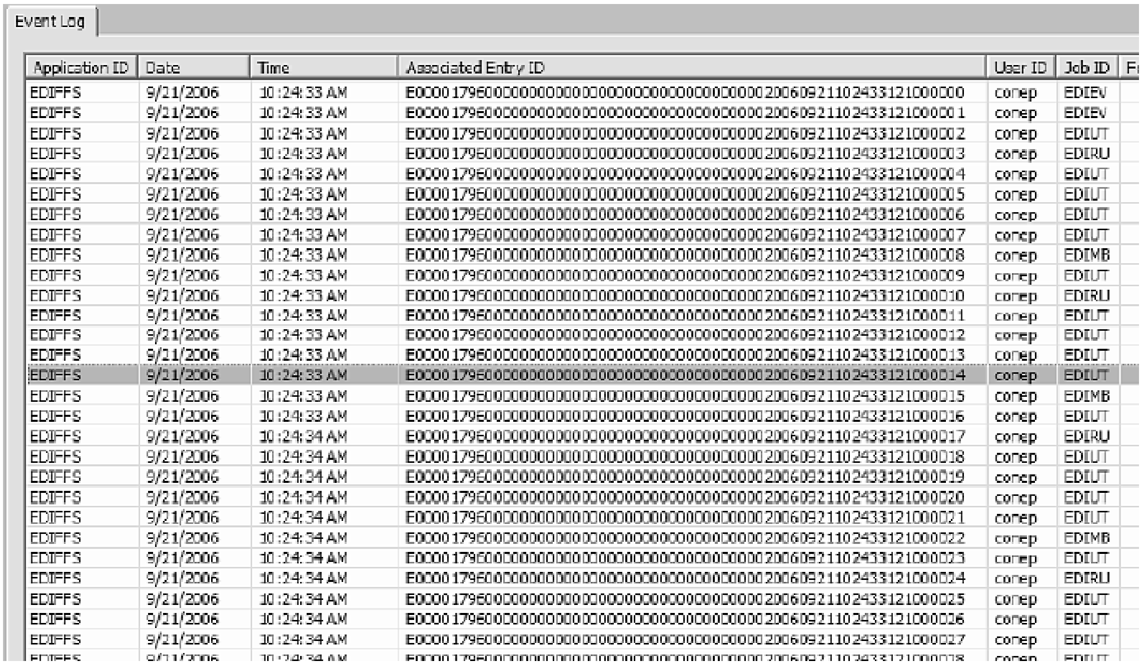
Figure 1. Specify selection criteria dialog box

If you want to see all the events, click Okay. If you want to filter the events you see, complete the associated fields for filtering on this dialog box.

Note: Entering filter values enables you to reduce the number of entries you will see in the event log list window.

Once you have typed any filter values you want, click Okay.

The query executes and the results display in the Event Log list window. Figure 2 on page 51 shows a sample Event Log list window.



Event Log

Application ID	Date	Time	Associated Entry ID	User ID	Job ID
EDIFFS	9/21/2006	10:24:33 AM	E0000 17960000000000000000000000000000002006092110243312100000	corep	EDIEV
EDIFFS	9/21/2006	10:24:33 AM	E0000 17960000000000000000000000000000002006092110243312100001	corep	EDIEV
EDIFFS	9/21/2006	10:24:33 AM	E0000 17960000000000000000000000000000002006092110243312100002	corep	EDIUT
EDIFFS	9/21/2006	10:24:33 AM	E0000 17960000000000000000000000000000002006092110243312100003	corep	EDIRU
EDIFFS	9/21/2006	10:24:33 AM	E0000 17960000000000000000000000000000002006092110243312100004	corep	EDIUT
EDIFFS	9/21/2006	10:24:33 AM	E0000 17960000000000000000000000000000002006092110243312100005	corep	EDIUT
EDIFFS	9/21/2006	10:24:33 AM	E0000 17960000000000000000000000000000002006092110243312100006	corep	EDIUT
EDIFFS	9/21/2006	10:24:33 AM	E0000 17960000000000000000000000000000002006092110243312100007	corep	EDIUT
EDIFFS	9/21/2006	10:24:33 AM	E0000 17960000000000000000000000000000002006092110243312100008	corep	EDIMB
EDIFFS	9/21/2006	10:24:33 AM	E0000 17960000000000000000000000000000002006092110243312100009	corep	EDIUT
EDIFFS	9/21/2006	10:24:33 AM	E0000 17960000000000000000000000000000002006092110243312100010	corep	EDIRU
EDIFFS	9/21/2006	10:24:33 AM	E0000 17960000000000000000000000000000002006092110243312100011	corep	EDIUT
EDIFFS	9/21/2006	10:24:33 AM	E0000 17960000000000000000000000000000002006092110243312100012	corep	EDIUT
EDIFFS	9/21/2006	10:24:33 AM	E0000 17960000000000000000000000000000002006092110243312100013	corep	EDIUT
EDIFFS	9/21/2006	10:24:33 AM	E0000 17960000000000000000000000000000002006092110243312100014	corep	EDIUT
EDIFFS	9/21/2006	10:24:33 AM	E0000 17960000000000000000000000000000002006092110243312100015	corep	EDIMB
EDIFFS	9/21/2006	10:24:33 AM	E0000 17960000000000000000000000000000002006092110243312100016	corep	EDIUT
EDIFFS	9/21/2006	10:24:34 AM	E0000 17960000000000000000000000000000002006092110243312100017	corep	EDIRU
EDIFFS	9/21/2006	10:24:34 AM	E0000 17960000000000000000000000000000002006092110243312100018	corep	EDIUT
EDIFFS	9/21/2006	10:24:34 AM	E0000 17960000000000000000000000000000002006092110243312100019	corep	EDIUT
EDIFFS	9/21/2006	10:24:34 AM	E0000 17960000000000000000000000000000002006092110243312100020	corep	EDIUT
EDIFFS	9/21/2006	10:24:34 AM	E0000 17960000000000000000000000000000002006092110243312100021	corep	EDIUT
EDIFFS	9/21/2006	10:24:34 AM	E0000 17960000000000000000000000000000002006092110243312100022	corep	EDIMB
EDIFFS	9/21/2006	10:24:34 AM	E0000 17960000000000000000000000000000002006092110243312100023	corep	EDIUT
EDIFFS	9/21/2006	10:24:34 AM	E0000 17960000000000000000000000000000002006092110243312100024	corep	EDIRU
EDIFFS	9/21/2006	10:24:34 AM	E0000 17960000000000000000000000000000002006092110243312100025	corep	EDIUT
EDIFFS	9/21/2006	10:24:34 AM	E0000 17960000000000000000000000000000002006092110243312100026	corep	EDIUT
EDIFFS	9/21/2006	10:24:34 AM	E0000 17960000000000000000000000000000002006092110243312100027	corep	EDIUT

Figure 2. WebSphere Data Interchange sample event log list window

Unless you have created and used another query for the Event Log list window, the dialog box will use the default find query. The Find query enables you to indicate:

- The name of the application in use when the event log entry was written. Specify this value if you want to see only entries from a specific application. The name of the application is the name of the application defaults profile that was is use when the entry was written to the event log.
- That you only want to see event log entries generated by a specific user ID.
- That you only want to see entries from a specific date. The date uses the format CCYYMMDD.
- That you only want to see entries from a specific time. The time uses the format HHMMSS. The filter values can be specified in any combination and none is required.

For instance, if no filter values are typed, you will see every entry from every Event Log. You can receive all of the entries from only one application that occurred on a specific date by filling in the **Application ID** and **Date** fields for filter values. You can also use a wild card character, for example the percent sign %, to use less specific values. For example, if you wanted to see all entries from a specific date that occurred in the 10 o'clock hour, you would fill in the date field and then specify the time field as *10%*. You can create your own queries to use to view the event log. You can have as many queries for the event log as you like.

Chapter 5. Common events handling

The Common Events Handler provides the following:

- An XML version of the Print File. This allows applications to easily parse the error messages and generate appropriate alerts or other notifications.
- An ADF (fixed record format) version of the Print File.
- The ability to route print files to a message queue, file, directory, or CICS transient data or temporary storage queue for post-WebSphere Data Interchange processing.
- By creating Event Destination Profiles, you can route print files to different destinations based on certain criteria, such as the maximum severity level or whether a particular error message occurred.
- A WebSphere Data Interchange supplied Java API handler or a user-written handler can then read these files from each destination and process them.
- The filtering messages to the Print File, the XML Print File, and the ADF Print File. Through filtering the you have the flexibility to eliminate messages that you consider unnecessary.
- The inclusion of more information about the documents that cause a particular error allowing you to more easily determine which input file or document caused a particular error.
- During DT translations, the Document Store handle is included in Event Log entries.
- The asynchronous option for Event Logging. This option reduces performance impacts on the translator.
- The option to write Print File messages during Data Transformation processing, instead of holding messages in memory until processing is complete. This reduces memory usage when many messages are logged, and enables you to see progress during large translations.

See the *WebSphere Data Interchange User's Guide* and *WebSphere Data Interchange Programmer's Reference Guide* for additional information on using Common Event Handling.

Common Event Handler Java API

You must verify that the Common Event Handling Java API is properly installed. The prerequisite is *Java 1.4 Standard Edition* or higher is installed and configured for your environment. To verify that the Common Event Handling Java API is properly installed, complete these steps:

1. Verify the Java run time environment.
   a. In a command window, type `java -version` , then press Enter. The output should provide you with the version of Java installed on your system:
      ```
      java version "1.4.2_12" Java(TM) 2 Runtime Environment, Standard Edition
      ```
      If you do not see a message similar to the one shown here, then you need to either install or configure Java before you proceed to the next step.
2. Change your current directory to the WebSphere Data Interchange samples directory.
3. On the windows command line, type: `cd C:\Program Files\IBM\WDIServer\V3.3\samples`, then press Enter. The verify program runs the WebSphere Data Interchange utility creating a prtfile, then the Common Event Handler Java API processes the prtfile and outputs the results for verification to the file WDITESTCEH.output
4. On the command line, type: `wditestceh.bat` then press Enter. The file WDITESTCEH.output is created.
5. Open the file and verify that it contains the message indicating successful execution:
   `WebSphere Data Interchange Common Event Handler Java API Received and processed the following data:` the normal prtfile data should follow the message.

Configuring for email notification

The Common Event Handling Java API can be used in conjunction with the sample Common Event Handler Java plug-in. This feature requires access to an SMTP compliant email server, either through an ISP or an in-house SMTP server. You also need to install the *mail.jar* and the *activation.jar* in addition to the Common Event Handler Java plug-in. You can download these jars from http://java.sun.com/products/javamail/. The mail.jar is part of the JavaMail 1.4 API and the activation.jar is part of the JavaBeans Activation Framework. You must add the path to the installed location of these jars to the CLASSPATH environment variable.

See the*WebSphere Data Interchange User's Guide* and *WebSphere Data Interchange Programmer's Reference Guide* for more information about setting the wdi.properties values and the Event Destination Profile to enable routing of events to email notification.

Appendix

Security scenarios

Server access control by role scenarios

Five possible scenarios that would use this role based access control implementation are described in this section. The first four scenarios describe situations where it is required that access to configuration artifacts be controlled based upon the user, and the type of artifact (trading partners, maps, etc.) that the user wants to access. In the fifth scenario, access is controlled based upon the user, the type of artifact and the members of the artifact. This last scenario demonstrates how to configure WebSphere Data Interchange and DB2 such that a user is restricted to a subset of an artifact, for example just those trading partners in North America.

Scenario 1

A customer wants to restrict the Mapping staff from updating Trading Partner Information. Mappers should be able to change Data Formats, Standards, XML definitions, code lists, and Maps. They could view, but should not be able to change, profiles. They should not be permitted to view Document Store.

Solution: Create a group called MAPPERS using the access control tool on the platform where the WebSphere Data Interchange database resides. For example, on z/OS you could use RACF to create the group.

1. Add the user IDs of the Mapping Staff to the group MAPPERS.
2. Copy the file **mapper_perms.ddl** to a name of your choosing.
3. Review the Mapper Class section and determine to which configuration artifacts mappers should be given access in your installation.
4. Change the SELECT, INSERT, UPDATE, DELETE clauses as required on the objects to which you want to restrict access. To prevent any access to a configuration artifact, comment out the appropriate GRANT statements. In this case, the Transaction Store objects should be commented out.
5. Do a global change from User01 to GROUP MAPPERS
6. Submit the DDL file to DB2 and verify that it ran successfully.

Scenario 2

The Implementation Group of a customer can migrate and maintain usages, maintain trading partners, and view document store and the event log. They can view but not change maps and mapping objects.

Solution: Create a group called IMPLMNTS using the access control tool on the platform where the WebSphere Data Interchange database resides. For example, on z/OS you could use RACF to create the group.

1. Add the user IDs of the Implementation Group to the group IMPLMNTS.
2. Copy the file **EDIAdmin_perms.ddl** to a name of your choosing.

3. Review the EDIAdminr Class section and determine to which configuration artifacts administrators should be given access in your installation.
4. Change the SELECT, INSERT, UPDATE, DELETE clauses as required on the objects to which you need to restrict access. To prevent any access to a configuration artifact, comment out the appropriate GRANT statements.
5. Do a global change from User01 to GROUP IMPLMNTS.
6. Submit the DDL file to DB2 and verify that it ran successfully.

Scenario 3

A special group has been asked to maintain code lists. They can view standards and nothing else. They need the ability to add, update, delete entries in a code list – including the creation of new code lists.

Solution: Create a group called CODELIST using the access control tool on the platform where the WebSphere Data Interchange database resides. For example, on z/OS you could use RACF to create the group.

1. Add the user ID's of the special group to the group CODELIST.
2. Copy the file **EDIAdmin_perms.ddl** to a name of your choosing.
3. Review the EDIAdmin Class section and determine to which configuration artifacts code list should be given access in your installation.
4. Change the SELECT, INSERT, UPDATE, DELETE clauses as required on the objects to which you need to restrict access. To prevent any access to a configuration artifact, comment out the appropriate GRANT statements. In this case, comment out all the GRANT statements except those under the heading Standards Tables.
5. Do a global change from User01 to GROUP CODELIST.
6. Submit the DDL file to DB2 and verify that it ran successfully.
7. Repeat the last 2 steps for each userid in the code list group

Scenario 4

1) The Operations group has view only access to all components in the WebSphere Data Interchange Client.

Solution: Create a group called OPERATOR using the access control tool on the platform where the WebSphere Data Interchange database resides. For example, on z/OS you could use RACF to create the group.

1. Add the user ID's of the Operations Group to the group OPERATOR.
2. Copy the file **operator_perms.ddl** to a name of your choosing.
3. Review the Operator Class section and determine to which configuration artifacts operators should be given access in your installation.
4. Change the SELECT, INSERT, UPDATE, DELETE clauses as required on the objects to which you need to restrict access. To prevent any access to a configuration artifact, comment out the appropriate GRANT statements.
5. Do a global change from User01 to GROUP OPERATOR.
6. Submit the DDL file to DB2 and verify that it ran successfully.

Scenario 5

A global company wants to consolidate all their EDI processing to a single data center and a single production database. They do not want to consolidate the EDI staffs into one staff because time zones, local customs and language barriers make it difficult to deal with trading partners all over the world from a single location. They are concerned that the EDI staff in one region might unintentionally damage the configuration for another region, so they would like to restrict the access of each of the regional offices to just those configuration artifacts that apply to them. For example, the Asia-Pacific region users should not be able to see or modify the trading partners for the North American and European regions, and vice versa. To address this concern, the global company can:

1. Copy the file **views.ddl** three times to names of your choosing. For this scenario we assume the names are **asia_config.ddl, namerica_config.ddl,** and **europe_config.ddl**.
2. Edit the file **asia_config.ddl** and make the following global changes:
   a. From ROLE. to ASIA..
   b. From **no_access_list** to NA, EU
   c. Submit the DDL file to DB2 and verify that it ran successfully.
3. Edit the file **na_config.ddl** and make the following global changes:
   a. From ROLE. to NAMERICA..
   b. From **no_access_list** to AS, EU
   c. Submit the DDL file to DB2 and verify that it ran successfully.
4. Edit the file na_config.ddl and make the following global changes:
   a. From ROLE. to EUROPE..
   b. From **no_access_list** to AS, NA.
   c. Submit the DDL file to DB2 and verify that it ran successfully.
5. Create a group called ASIA, a group called NAMERICA and a group called EUROPE using the access control tool on the platform where the WebSphere Data Interchange database resides. For example, on z/OS you could use RACF to create the group.
6. Add the user IDs of the EDI staff in the Asia-Pacific region to the group ASIA, the user IDs of the EDI staff in the North American region to the group NAMERICA and the user IDs of the EDI staff in the North American region to the group EUROPE.
7. For each region (Europe, North America and Asia):
   a. Determine the set of roles the region requires, such as mapper, operator, system administrator, etc.
   b. Copy the appropriate permission files (such as **sysadmin_perms.ddl** and **mapper_perms.ddl**) to names of your choosing, for example **europe_mapper.ddl**.
   c. Change the SELECT, INSERT, UPDATE, DELETE clauses as required on the objects to which you need to restrict access. To prevent any access to a configuration artifact, comment out the appropriate GRANT statements.
   d. Do a global change from User01 to GROUP *region_name*, where *region_name* is either ASIA, NAMERICA, or EUROPE.

   e. Submit the DDL file to DB2 and verify that it ran successfully.
8. For each client system in each region:
   a. Create a new system and name it the region name.
   b. Specify the same database parameters for the new system as the base EDIENU33 system, **except** for the Database Qualifier field.
   c. Enter the region name in that field.
   d. Set the client system user to use the system corresponding to their region name and role when connecting to the consolidated database.

Configuration artifacts overview

Introduction to client security

The ability to control the access of users according to role is a repeatedly requested customer requirement for the WebSphere Data Interchange V3.3 Client. Customers need the ability to restrict the components of the WebSphere Data Interchange Client to a set of users. Typical roles are Mapping, Trading Partner Maintenance, Configuration or Setup data Administration, and Operations (e.g. users of Transaction Store and Event Log).

In a typical EDI Installation, the roles that employees perform should determine the database access they are granted. To give access outside of the role domain is to unnecessarily expose the system to inadvertent or malicious changes.

In WebSphere Data Interchange V3.3 Client, a *View* functionality has been implemented in conjunction with a set of DB2 GRANTS to implement a role based security capability. View works exactly like *Open* except that the editors and related dialogs are opened in read-only mode.

The read-only mode is not new to the Client. The Client has supported read-only mode for all editors and related dialogs in the past. This applied when an object that was locked was opened. The client displayed the editor in read-only mode. The new View function simply provides a mechanism for the user to open the editors and related dialogs in read-only mode without the object being locked.

You can access the View function on the File menu. In addition, the preferences dialog now permits the user to indicate whether Open or View will be the default when you click Open on the toolbar. The setting of this preference also affects the action that occurs when you double clicks on an object that normally would produce an Open action. This preference setting permits users who only have read access to the database to work efficiently without the added steps of going to the File menu.

Access to tables is controlled using database authorizations (GRANT) and views.

Configuration artifacts

A **configuration artifact** is an object within the WebSphere Data Interchange client that can be edited or accessed via an editor. The implementation of the ConfigArtifact class in WebSphere Data Interchange is the set of DB2 tables used to represent the object in the WebSphere Data Interchange database.

Table 7. Server database tables

Configuration artifact	DB2 Table name
Application Defaults Profile	EDIENU33.EDIPSAP
Audit Trail	EDIENU33.AUDITHDR
CICS Performance Profile	EDIENU33.EDIPSSY
Commands	EDIENU33.EDIDBREQ EDIENU33.EDIDBFILE1 EDIENU33.EDIDBFILE2 EDIENU33.EDICMDDICT EDIENU33.EDICMD EDIENU33.EDICMDREF
Continuous Receive Profile	EDIENU33.EDIPSCR
Comment	EDIENU33.EDITPCM
Contact	EDIENU33.EDITPCN EDIENU33.EDITPCT
Control Strings	
EDI Envelope Control String	EDIENU33.EDICSTX EDIENU33.EDIMAPCSTHDR EDIENU33.EDIMAPCSTDETADF
EDI Envelope Standard	EDIENU33.EDISTDENV
Data Format Profiles	
Data Format Dictionary	EDIENU33.EDIADFDICT
Data Format Record ID Information	EDIENU33.EDIADFRECIDINFO
Data Format	EDIENU33.EDIADFHEADER EDIENU33.EDIADFHDRMEM
Data Format Loop	EDIENU33.EDIADFLOOP EDIENU33.EDIADFLOOPMEM
Data Format Record	EDIENU33.EDIADFRECORD EDIENU33.EDIADFRECMEM
Data Format Structure	EDIENU33.EDIADFSTRUCT EDIENU33.EDIADFSTRUCTMEM
Data Format Field	EDIENU33.EDIADFFIELD
Data References Profile	EDIENU33.EDIDATAREF
Document Store Profile–All objects	EDIENU33.EDIDSDOC EDIENU33.EDIDSEXTN EDIENU33.EDIDSATTR EDIENU33.EDIDSREL EDIENU33.EDIDSRELATTR EDIENU33.EDIDSACT EDIENU33.EDIDSACTATTR EDIENU33.EDIDSSTATUS EDIENU33.EDIDSDTL EDIENU33.EDIDSIMG

Table 7. Server database tables (continued)

Configuration artifact	DB2 Table name
EDI Standards Profiles	
EDI Standard Dictionary	EDIENU33.EDISTDSTH
EDI Standard Transaction	EDIENU33.EDISTDTXD EDIENU33.EDISTDTXH EDIENU33.EDISTDTXN
EDI Standard Segment	EDIENU33.EDISTDSGD EDIENU33.EDISTDSGH EDIENU33.EDISTDSGN
EDI Standard Data Element	EDIENU33.EDISTDDED EDIENU33.EDISTDDEH
EDI Standard Composite Data Element Notes	EDIENU33.EDISTDCDN
EDI Standard Code List	EDIENU33.EDIPSTV EDIENU33.EDIPSTT EDIENU33.EDIPSTD
Envelope Profiles	
E Envelope Profile	EDIENU33.EDIPSEE
I Envelope Profile	EDIENU33.EDIPSIE
T Envelope Profile	EDIENU33.EDIPSTE
U Envelope Profile	EDIENU33.EDIPSUE
X Envelope Profile	EDIENU33.EDIPSXE
Event Destination Profile	EDIENU33.EDIPSDP
Event Log	EDIENU33.EDIELOG
Language Profile	EDIENU33.EDIPSLP
Mailbox Profile	EDIENU33.EDIPSRQ
Management Reporting	EDIENU33.EDIMRCM EDIENU33.EDIMRPC EDIENU33.EDIMRPR EDIENU33.EDIMRPS EDIENU33.EDIMRRT EDIENU33.EDIMRST
Mapping Profiles	
Data Transformation Map Rule	EDIENU33.EDIRULE
Send Map Usage	EDIENU33.EDITPST
Receive Map Usage	EDIENU33.EDITPRT
Map Application Control Fields	EDIENU33.EDIMAPAPPLCNTL
Mapping Commands	EDIENU33.EDIMAPCMDS
Mapping Elements	EDIENU33.EDIMAPELE
Map Global Variables	EDIENU33.EDIMAPGBLVAR
Map Header	EDIENU33.EDIMAPHEAD

Table 7. Server database tables (continued)

Configuration artifact	**DB2 Table name**
Map Local Variables	EDIENU33.EDIMAPLCLVAR
Mapping Command Nodes	EDIENU33.EDIMAPNODES
Mapping Cross References	EDIENU33.EDIMAPREF
Mapping Segments	EDIENU33.EDIMAPSEG
Mapping Syntax	EDIENU33.EDIMAPSYNTAX
Map Report	EDIENU33.EDIMAPRPT
MCD Profile	EDIENU33.EDIPSMCD
Profile Messages	EDIENU33.EDIPSMS
Rule	EDIENU33.EDIRULE
Security Profiles	EDIENU33.EDISECROLE EDIENU33.EDISECROLEDETROLE EDIENU33.EDISECUID EDIENU33.EDISECUIDDETROLE EDIENU33.EDISECGRP EDIENU33.EDISECUIDDETGRP EDIENU33.EDISECUIDDETOB
Service Profile	EDIENU33.EDIPSSL
Trading Partner Profile	EDIENU33.EDIPROF EDIENU33.EDIPSTP EDIENU33.EDIPSBI
Transaction Store–All objects	EDIENU33.EDITSTH EDIENU33.EDITSEV EDIENU33.EDITSGP EDIENU33.EDITSTI EDIENU33.EDITSTO EDIENU33.EDITSAU EDIENU33.EDITSLT EDIENU33.EDIVTSTH EDIENU33.EDIVTSIC
User Exits Profile	EDIENU33.EDIPSAD
Networking Profiles	
Network Command Profile	EDIENU33.EDIPSNO
Network Security Profile	EDIENU33.EDIPSSP
Network Profile	EDIENU33.EDIPSNP
WebSphere MQ Profile	EDIENU33.EDIPSMQ
XML Profiles	
XML Dictionary	EDIENU33.EDIXMLDICT
XML DTD	EDIENU33.EDIDTDHDR EDIENU33.EDIDTD
XML Schema	EDIENU33.EDIDTDHDR EDIENU33.EDIDTD

Table 7. Server database tables (continued)

Configuration artifact	DB2 Table name
XML Namespace	EDIENU33.EDIXMLNS

A number of sets of DB2 Data Definition Language (DDL) are provided to show how different role based access scenarios can be implemented. Review the Conceptual Model section of this document to understand the relationship between permissions, WebSphere Data Interchange DB2 Tables, and Roles.

Views

A *view* defines a subset of instances of the configuration artifact that can be seen by the users. It is used to limit which instances of a particular configuration artifact that the user can affect. For example, if a user should be allowed to access the X12 standards, but not any other kind of standard, then a view could be created that shows only standards beginning with the three characters X12–assuming that all X12 standards begin with X12.

A view can be created on any configuration artifact. The implementation of the View class in WebSphere Data Interchange V3.3 is a DB2 view. There is a file called *views.ddl* provided with the product that the user can copy, rename and edit as required to create views. In the file there is a CREATE VIEW role_name.table_name AS SELECT * FROM EDIENU33.table_name statement for each table in the WebSphere Data Interchange database.

For example, to restrict a mapper role to only be able to see rows defining X12 standards, the following CREATE VIEW statement for the EDISTDSTH table might be used:

```
--Create a role specific view of the Standard Dictionary table;
 CREATE VIEW MAPPER.EDISTDSTH AS SELECT * FROM
  EDIENU33.EDISTDSTH
        WHERE STDID = 'X12%'
--Terminate the CREATE VIEW statement:    ;
--Create a VIEW  for every table. For those that do not need to be restricted,
  the WHERE clause is not needed.
```

The system administrator is able to restrict the instances of each configuration artifact that a particular role sees by adding an appropriate WHERE clause to the views that are used to represent the configuration artifact.

Groups

A Group is a collection of users who have the same role. An alternative to defining each user with a unique set of authorizations is to assign the authorizations to a GROUP and then assign individual users to that GROUP. This technique is not available on all platforms. With DB2, the technique is shown below:

```
--Grant authority for the group role_name to access this
--table by changing the following grant statment as
--required:   GRANT SELECT, INSERT, DELETE, UPDATE ON role_name.EDISTDSTH
TO GROUP role_name;  On z/OS, then add users to the RACF Group name role_name.
```

On Windows Server, use Windows Users and Passwords Administration to associate users with the Group.

On AIX, use SMIT or SMITTY to associate users with the Group.

On Windows 2000 Professional and Windows XP, use DB2 Connect to make the user assignments.

Classes

SysAdmin

The SysAdmin class represents the system administrator. The system administrator must have access to everything and be able to see all instances in order to do their job. The SysAdmin is also the person that manages the users in the security facility and must have authority to create views in the DB2 instance where the WebSphere Data Interchange database resides. The implementation of the SysAdmin class in V3.3 is as a role with full permissions on every table and no restrictions on any view. There is a file called sysadmin_perms.ddl provided with the product that the user can edit as required to create permissions and views for the SysAdmin role. In the file all the GRANT statements contain the full SELECT, INSERT, UPDATE, DELETE clause.

EDIAdmin

The EDIAdmin class represents the EDI administrator. The EDI administrator manages the trading partner community. They must have access to everything necessary to define new trading partners to the system and configure them such that messages flowing to or from the trading partner will be handled correctly. The implementation of the EDIAdmin class in WebSphere Data Interchange; V3.3 is as a role with permissions as shown above and no restrictions on any view. There is a file called ediadmin_perms.ddl provided with the product that the user can edit as required to create permissions and views for the EDIAdmin role. The default set of required permissions are:

Table 8. EDIAdmin Role permissions

Configuration artifact	Read	Insert	Update	Delete
Application Defaults Profile	Y	N	N	N
Audit Trail	Y	N	N	N
CICS Performance Profile	Y	N	N	N
Commands	Y	N	N	N
Continuous Receive Profile	Y	N	N	N
Comment	Y	Y	Y	Y
Contact	Y	Y	Y	Y

Table 8. EDIAdmin Role permissions (continued)

Configuration artifact	Read	Insert	Update	Delete
Control Strings				
EDI Envelope Control String	Y	N	N	N
EDI Envelope Standard	Y	N	N	N
Data Format Profiles				
YData Format Dictionary	Y	N	N	N
Data Format Record ID Information	Y	N	N	N
Data Format	Y	N	N	N
Data Format Loop	Y	N	N	N
Data Format Record	Y	N	N	N
Data Format Structure	Y	N	N	N
Data Format Field	Y	N	N	N
Data References Profile	Y	N	N	N
Document Store Profile–All objects	Y	N	N	N
EDI Standards Profiles				
EDI Standard Dictionary	Y	N	N	N
EDI Standard Transaction	Y	N	N	N
EDI Standard Segment	Y	N	N	N
EDI Standard Data Element	Y	N	N	N
EDI Standard Composite Data Element Notes	Y	N	N	N
EDI Standard Code List	Y	N	N	N
Envelope Profiles				
E Envelope Profile	Y	N	N	N
I Envelope Profile	Y	N	N	N
T Envelope Profile	Y	N	N	N
U Envelope Profile	Y	N	N	N
X Envelope Profile	Y	N	N	N
Event Destination Profile	Y	N	N	N
Event Log	Y	N	N	N
Language Profile	Y	N	N	N
Mailbox Profile	Y	N	N	N
Management Reporting	Y	N	N	N
Mapping Profiles				
Data Transformation Map Rule	Y	N	N	N
Send Map Usage	Y	Y	Y	Y
Receive Map Usage	Y	Y	Y	Y

Table 8. EDIAdmin Role permissions (continued)

Configuration artifact	**Read**	**Insert**	**Update**	**Delete**
Map Application Control Fields	Y	N	N	N
Mapping Commands	Y	N	N	N
Mapping Elements	Y	N	N	N
Map Global Variables	Y	N	N	N
Map Header	Y	N	N	N
Map Local Variables	Y	N	N	N
Mapping Command Nodes	Y	N	N	N
Mapping Cross References	Y	N	N	N
Mapping Segments	Y	N	N	N
Mapping Syntax	Y	N	N	N
Map Report	Y	N	N	N
MCD Profile	Y	N	N	N
Profile Messages	Y	N	N	N
Rule	Y	Y	Y	Y
Security Profiles	Y	N	N	N
Service Profile	Y	N	N	N
Trading Partner Profile	Y	Y	Y	Y
Transaction Store–All objects	Y	N	N	N
User Exits Profile	Y	N	N	N
Networking Profiles				
Network Command Profile	Y	N	N	N
Network Security Profile	Y	N	N	N
Network Profile	Y	N	N	N
WebSphere MQ Profile	Y	N	N	N
XML Profiles				
XML Dictionary	Y	N	N	N
XML DTD	Y	N	N	N
XML Schema	Y	N	N	N
XML Namespace	Y	N	N	N

SystemsIntegrator

The SystemsIntegrator class represents the role of the person that integrates WebSphere Data Interchange; into the customer's IT systems. The systems integrator manages the queues, networks, files, PERFORM command, service profiles etc. in order to move data between WebSphere Data Interchange, backend systems and trading partners. They must have access to everything necessary to implement integration of WebSphere Data Interchange; with their own systems. The implementation of the SystemsIntegrator class in V3.3 is as a role with permissions as shown above and no restrictions on any view. There is a file called sysintegrator_perms.ddl provided with the product that the user can edit as required to create permissions and views for the SystemsIntegrator role. The default set of required permissions are:

Table 9. SystemsIntegrator Role permissions

Configuration artifact	**Read**	**Insert**	**Update**	**Delete**
Application Defaults Profile	Y	Y	Y	Y
Audit Trail	Y	Y	Y	Y
CICS Performance Profile	Y	Y	Y	Y
Commands	Y	Y	Y	Y
Continuous Receive Profile	Y	Y	Y	Y
Comment	Y	Y	Y	Y
Contact	Y	Y	Y	Y
Control Strings				
EDI Envelope Control String	Y	N	N	N
EDI Envelope Standard	Y	N	N	N
Data Format Profiles				
Data Format Dictionary	Y	N	N	N
Data Format Record ID Information	Y	N	N	N
Data Format	Y	N	N	N
Data Format Loop	Y	N	N	N
Data Format Record	Y	N	N	N
Data Format Structure	Y	N	N	N
Data Format Field	Y	N	N	N
Data References Profile	Y	N	N	N
Document Store Profile–All objects	Y	N	N	N
EDI Standards Profiles				
EDI Standard Dictionary	Y	N	N	N
EDI Standard Transaction	Y	N	N	N
EDI Standard Segment	Y	N	N	N
EDI Standard Data Element	Y	N	N	N
EDI Standard Composite Data Element Notes	Y	N	N	N
EDI Standard Code List	Y	N	N	N
Envelope Profiles				
E Envelope Profile	Y	N	N	N
I Envelope Profile	Y	N	N	N
T Envelope Profile	Y	N	N	N

Table 9. SystemsIntegrator Role permissions (continued)

Configuration artifact	**Read**	**Insert**	**Update**	**Delete**
U Envelope Profile	Y	N	N	N
X Envelope Profile	Y	N	N	N
Event Destination Profile	Y	Y	Y	Y
Event Log	Y	Y	Y	Y
Language Profile	Y	Y	Y	Y
Mailbox Profile	Y	Y	Y	Y
Management Reporting	Y	N	N	N
Mapping Profiles				
Data Transformation Map Rule	Y	N	N	N
Send Map Usage	Y	Y	Y	Y
Receive Map Usage	Y	Y	Y	Y
Map Application Control Fields	Y	N	N	N
Mapping Commands	Y	N	N	N
Mapping Elements	Y	N	N	N
Map Global Variables	Y	N	N	N
Map Header	Y	N	N	N
Map Local Variables	Y	N	N	N
Mapping Command Nodes	Y	N	N	N
Mapping Cross References	Y	N	N	N
Mapping Segments	Y	N	N	N
Mapping Syntax	Y	N	N	N
Map Report	Y	N	N	N
MCD Profile	Y	N	N	N
Profile Messages	Y	N	N	N
Rule	Y	Y	Y	Y
Security Profiles	Y	N	N	N
Service Profile	Y	N	N	N
Trading Partner Profile	Y	Y	Y	Y
Transaction Store–All objects	Y	N	N	N
User Exits Profile	Y	N	N	N
Networking Profiles				
Network Command Profile	Y	Y	Y	Y
Network Security Profile	Y	Y	Y	Y
Network Profile	Y	Y	Y	Y
WebSphere MQ Profile	Y	N	N	N
XML Profiles				

Table 9. SystemsIntegrator Role permissions (continued)

Configuration artifact	Read	Insert	Update	Delete
XML Dictionary	Y	N	N	N
XML DTD	Y	N	N	N
XML Schema	Y	N	N	N
XML Namespace	Y	N	N	N

Mapper

The Mapper class represents the role of the person that manages the transformation maps. They must have access to everything necessary to implement integration of WebSphere Data Interchange; with their own systems. The implementation of the Mapper class in V3.3 is as a role with permissions as shown above and no restrictions on any view. There is a file called mapper_perms.ddl provided with the product that the user can edit as required to create permissions and views for the Mapper role. The default set of required permissions are:

Table 10. Mapper Role permissions

Configuration artifact	Read	Insert	Update	Delete
Application Defaults Profile	Y	N	N	N
Audit Trail	Y	N	N	N
CICS Performance Profile	Y	N	N	N
Commands	Y	N	N	N
Continuous Receive Profile	Y	N	N	N
Comment	Y	N	N	N
Contact	Y	N	N	N
Control Strings				
EDI Envelope Control String	Y	N	N	N
EDI Envelope Standard	Y	N	N	N
Data Format Profiles				
Data Format Dictionary	Y	Y	Y	Y
Data Format Record ID Information	Y	Y	Y	Y
Data Format	Y	Y	Y	Y
Data Format Loop	Y	Y	Y	Y
Data Format Record	Y	Y	Y	Y
Data Format Structure	Y	Y	Y	Y
Data Format Field	Y	Y	Y	Y
Data References Profile	Y	Y	Y	Y
Document Store Profile–All objects	Y	N	N	N
EDI Standards Profiles				

Table 10. Mapper Role permissions (continued)

Configuration artifact	Read	Insert	Update	Delete
EDI Standard Dictionary	Y	Y	Y	Y
EDI Standard Transaction	Y	Y	Y	Y
EDI Standard Segment	Y	Y	Y	Y
EDI Standard Data Element	Y	Y	Y	Y
EDI Standard Composite Data Element Notes	Y	Y	Y	Y
EDI Standard Code List	Y	Y	Y	Y
Envelope Profiles				
E Envelope Profile	Y	Y	Y	Y
I Envelope Profile	Y	Y	Y	Y
T Envelope Profile	Y	Y	Y	Y
U Envelope Profile	Y	Y	Y	Y
X Envelope Profile	Y	Y	Y	Y
Event Destination Profile	Y	N	N	N
Event Log	Y	N	N	N
Language Profile	Y	N	N	N
Mailbox Profile	Y	N	N	N
Management Reporting	Y	N	N	N
Mapping Profiles				
Data Transformation Map Rule	Y	N	N	N
Send Map Usage	Y	Y	Y	Y
Receive Map Usage	Y	Y	Y	Y
Map Application Control Fields	Y	N	N	N
Mapping Commands	Y	N	N	N
Mapping Elements	Y	N	N	N
Map Global Variables	Y	N	N	N
Map Header	Y	N	N	N
Map Local Variables	Y	N	N	N
Mapping Command Nodes	Y	N	N	N
Mapping Cross References	Y	N	N	N
Mapping Segments	Y	N	N	N
Mapping Syntax	Y	N	N	N
Map Report	Y	N	N	N
MCD Profile	Y	Y	Y	Y
Profile Messages	Y	N	N	N
Rule	Y	Y	Y	Y

Table 10. Mapper Role permissions (continued)

Configuration artifact	Read	Insert	Update	Delete
Security Profiles	Y	N	N	N
Service Profile	Y	N	N	N
Trading Partner Profile	Y	Y	Y	Y
Transaction Store–All objects	Y	N	N	N
User Exits Profile	Y	N	N	N
Networking Profiles				
Network Command Profile	Y	Y	Y	Y
Network Security Profile	Y	Y	Y	Y
Network Profile	Y	Y	Y	Y
WebSphere MQ Profile	Y	N	N	N
XML Profiles				
XML Dictionary	Y	Y	Y	Y
XML DTD	Y	Y	Y	Y
XML Schema	Y	Y	Y	Y
XML Namespace	Y	Y	Y	Y

Operator

The Operator class represents the role of the person that manages customer data problems. They must have access to everything necessary to do problem determination and correction of translations involving customer data. The implementation of the Operator class in V3.3 is as a role with permissions as shown above and no restrictions on any view. There is a file called operator_perms.ddl provided with the product that the user can edit as required to create permissions and views for the Operator role. The default set of required permissions are:

Table 11. Operator Role permissions

Configuration artifact	Read	Insert	Update	Delete
Application Defaults Profile	Y	N	N	N
Audit Trail	Y	N	N	N
CICS Performance Profile	Y	N	N	N
Commands	Y	N	N	N
Continuous Receive Profile	Y	N	N	N
Comment	Y	N	N	N
Contact	Y	N	N	N
Control Strings				
EDI Envelope Control String	Y	N	N	N
EDI Envelope Standard	Y	N	N	N

Table 11. Operator Role permissions (continued)

Configuration artifact	Read	Insert	Update	Delete
Data Format Profiles				
Data Format Dictionary	Y	N	N	N
Data Format Record ID Information	Y	N	N	N
Data Format	Y	N	N	N
Data Format Loop	Y	N	N	N
Data Format Record	Y	N	N	N
Data Format Structure	Y	N	N	N
Data Format Field	Y	N	N	N
Data References Profile	Y	N	N	N
Document Store Profile–All objects	Y	N	N	N
EDI Standards Profiles				
EDI Standard Dictionary	Y	N	N	N
EDI Standard Transaction	Y	N	N	N
EDI Standard Segment	Y	N	N	N
EDI Standard Data Element	Y	N	N	N
EDI Standard Composite Data Element Notes	Y	N	N	N
EDI Standard Code List	Y	N	N	N
Envelope Profiles				
E Envelope Profile	Y	N	N	N
I Envelope Profile	Y	N	N	N
T Envelope Profile	Y	N	N	N
U Envelope Profile	Y	N	N	N
X Envelope Profile	Y	N	N	N
Event Destination Profile	Y	N	N	N
Event Log	Y	N	N	N
Language Profile	Y	N	N	N
Mailbox Profile	Y	N	N	N
Management Reporting	Y	N	N	N
Mapping Profiles				
Data Transformation Map Rule	Y	N	N	N
Send Map Usage	Y	N	N	N
Receive Map Usage	Y	N	N	N
Map Application Control Fields	Y	N	N	N
Mapping Commands	Y	N	N	N
Mapping Elements	Y	N	N	N

Table 11. Operator Role permissions (continued)

Configuration artifact	Read	Insert	Update	Delete
Map Global Variables	Y	N	N	N
Map Header	Y	N	N	N
Map Local Variables	Y	N	N	N
Mapping Command Nodes	Y	N	N	N
Mapping Cross References	Y	N	N	N
Mapping Segments	Y	N	N	N
Mapping Syntax	Y	N	N	N
Map Report	Y	N	N	N
MCD Profile	Y	N	N	N
Profile Messages	Y	N	N	N
Rule	Y	N	N	N
Security Profiles	Y	N	N	N
Service Profile	Y	N	N	N
Trading Partner Profile	Y	N	N	N
Transaction Store–All objects	Y	N	N	N
User Exits Profile	Y	N	N	N
Networking Profiles				
Network Command Profile	Y	N	N	N
Network Security Profile	Y	N	N	N
Network Profile	Y	N	N	N
WebSphere MQ Profile	Y	N	N	N
XML Profiles				
XML Dictionary	Y	N	N	N
XML DTD	Y	N	N	N
XML Schema	Y	N	N	N
XML Namespace	Y	N	N	N

Notices

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 Corporation
Department DD40
P.O. Box 30021
Tampa, Florida 33630-3021 USA

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 document and all licensed material available for it are provided by IBM under terms of the IBM Customer Agreement, IBM International Program License Agreement or any equivalent agreement between us.

Any performance data contained herein was determined in a controlled environment. Therefore, the results obtained in other operating environments may vary significantly. Some measurements may have been made on development-level systems and there is no guarantee that these measurements will be the same on generally available systems. Furthermore, some measurements may have been estimated through extrapolation. Actual results may vary. Users of this document should verify the applicable data for their specific environment.

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.

All statements regarding IBM's future direction or intent are subject to change or withdrawal without notice, and represent goals and objectives only.

All IBM prices shown are IBM's suggested retail prices, are current and are subject to change without notice. Dealer prices may vary.

This information is for planning purposes only. The information herein is subject to change before the products described become available.

This information contains examples of data and reports used in daily business operations. To illustrate them as completely as possible, the examples include the names of individuals, companies, brands, and products. All of these names are fictitious and any similarity to the names and addresses used by an actual business enterprise is entirely coincidental.

Programming interface information

Programming interface information, if provided, is intended to help you create application software using this program.

General-use programming interfaces allow you to write application software that obtain the services of this program's tools.

However, this information may also contain diagnosis, modification, and tuning information. Diagnosis, modification and tuning information is provided to help you debug your application software.

Warning: Do not use this diagnosis, modification, and tuning information as a programming interface because it is subject to change.

Trademarks and service marks

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*.

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.

code list. A table, supplied by WebSphere Data Interchange or defined by the user, that contains all acceptable values for a single data field.

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.

control number. Numbers (or masks used to create numbers) that are used to identify an interchange, group, or EDI transaction.

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.

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.

D

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*.

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.

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.

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.

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.

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.

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

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.

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.

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.

O

ODETTE. Organization for Data Exchange through Teletransmission in Europe.

P

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.

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.

R

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*.

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

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.

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.

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.

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.

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.

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*.

Bibliography

This section describes the documentation available for the WebSphere Data Interchange product.

WebSphere Data Interchange publications

The WebSphere Data Interchange V3.3 publications are:

- *WebSphere Data Interchange for MultiPlatforms Quick Start Guide* CF0YREN
- *WebSphere Data Interchange for MultiPlatforms Administration and Security Guide* SC34-6214-01
- *WebSphere Data Interchange for MultiPlatforms Messages and Codes Guide* SC34-6216-01
- *WebSphere Data Interchange for MultiPlatforms User's Guide* SC34-6215-01
- *WebSphere Data Interchange for MultiPlatforms Programmer's Reference Guide* SC34-6217-01
- *WebSphere Data Interchange for MultiPlatforms Mapping Guide* SC23-5874-00
- *WebSphere Data Interchange for MultiPlatforms Utility Commands and File Formats Reference Guide* SC23-5873-00
- *WebSphere Data Interchange for z/OS V3.3 Program Directory* GI10-2561-01
- *WebSphere Data Interchange for z/OS V3.3 Installation Guide* SC34-6269-01
- *WebSphere Data Interchange for z/OS V3.3 License File* GC34-6270-02

Softcopy books

All the WebSphere Data Interchange books are available in softcopy format.

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.

Index

Printed in USA

SC34-6214-01