SeeBeyond ICAN Suite

ODETTE-FTP eWay Intelligent Adapter User's Guide

Release 5.0

Version 20050524125753.

Contents

Chapter 3

Chapter 4

Chapter 1

Introducing the ODETTE-FTP eWay

Welcome to the *ODETTE-FTP eWay*. This document includes information about installing, configuring, and using the SeeBeyond® Integrated Composite Application Network Suite™ (ICAN) ODETTE-FTP eWay Intelligent Adapter.

What's in This Chapter

1.1 About ODETTE-FTP

ODETTE-FTP was defined in 1986 by the Organization for Data Exchange by Tele Transmission in Europe (ODETTE) to address the electronic data interchange (EDI) requirements of the European automotive industry. It was designed in the spirit of the Open System Interconnection (OSI) model utilizing the Network Service provided by the CCITT X25.

ODETTE-FTP can be deployed on systems of all sizes, from personal computers to large mainframes. As a result of the wide scale deployment of internet technology and the trend towards global business practices, ODETTE has extended the scope of it's file transfer protocol to allow the use of TCP/IP and to make the protocol available to the Internet community.

The aim of the ODETTE-FTP is to facilitate the transmission of files between one or more locations in a way that is independent of the data communication network, system hardware and software environment.

In designing and specifying the protocol, the following factors are considered.

1. The possible differences of size and sophistication (file storage, small and large systems).
2. The necessity to work with existing systems (reduce changes to existing products and allow easy implementation).

3 Systems of different ages.

4 Systems of different manufacturers.

5 The potential for growth in sophistication (limit impact and avoid changes at other locations).

Information is always exchanged between ODETTE-FTP entities in a standard representation called a Virtual File. This allows data transfer without regard for the nature of the communicating systems.

Note: *The mapping of a file between a local and virtual representation varies from system to system.*

1.2 About the ODETTE-FTP eWay

The ODETTE-FTP eWay enables the ICAN system to exchange data with any other system using the ODETTE-FTP (OFTP) communication protocol.

The basic purpose of the ODETTE-FTP eWay is to transfer files over the OFTP using an Object Type Definition (OTD) specialized for this eWay. This OTD is called the ODEX OTD and is based on the eGate OTD file oftp.xsc.

Using this OTD, you can create eGate Collaboration Rules to customize the business logic and data transfers required by your system. The ODEX OTD's configuration allows you to control basic data-exchange operations in ODEX. The OTD's implementation is based on the ODEX batch interface.

Note: *For more information on OTDs and their use in ICAN, see the eGate Integrator User's Guide.*

The basic functions of the eWay are:

- Handle and operate data communication with ODEX.
- Interface with trading partners.

The eWay enables eGate Integrator Projects to exchange data with ODETTE-FTP. This document describes how to install and configure the eWay.

1.2.1 The eWay and the ODEX Application

To communicate with OFTP, the ODETTE-FTP eWay uses ODEX, a third-party standalone application. The eWay only launches and monitors the application, while ODEX does the actual OFTP file transfer.

ODEX is an end-user oriented software with graphical user interfaces (GUIs) for trading-partner configuration, electronic data interchange (EDI) job configuration, scheduling (automation), and data communication. ODEX also provides a batch interface allowing you to do these tasks using a batch program.

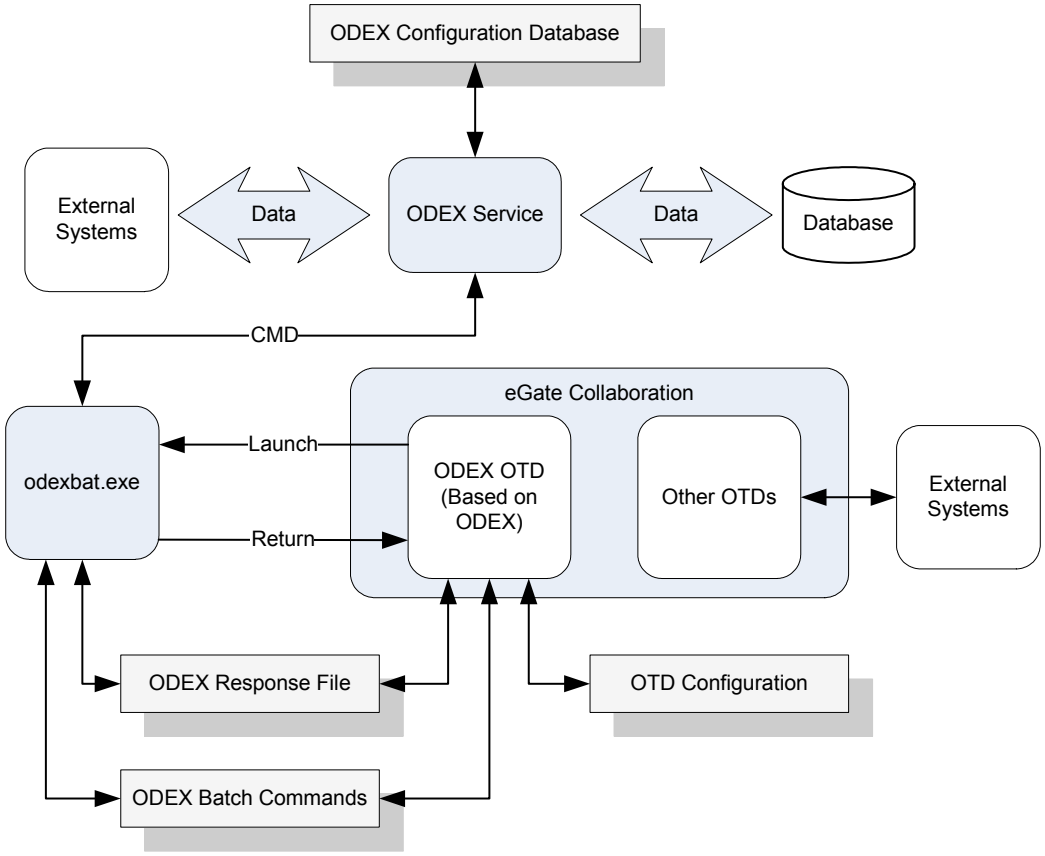
Because the eWay operates in conjunction with ODEX, its implementation is product specific. Correct implementation of this eWay assumes that ODEX is properly installed and configured on at least one eGate Logical Host. The eWay cannot configure ODEX. It can only keep necessary information available to communicate with and control the third-party application.

Note: *For complete details on how to set up and configure ODEX, see the appropriate documentation for that product.*

eWay Design Overview

Figure 1 shows a general diagram of how the ODETTE-FTP eWay, the ODEX OTD, and the ODEX application operate in supporting one another.

Figure 1 ODETTE-FTP eWay/ODEX Operation



The advantages of this setup are:

- An easy-to-use eWay
- Ability to operate ODEX using the eWay
- The robustness and versatility already available with ODEX

eWay Configuration

The configuration parameters for the ODETTE-FTP eWay, allow you to set necessary ODEX parameters of operation. In turn, these eWay parameters are adopted into the ODEX OTD's configuration.

When you configure the ODETTE-FTP eWay, its set configuration parameters must match the corresponding ODEX configurations. If the eWay and ODEX are sending and expecting different sets of instructions, communication cannot take place between the two.

See **Chapter 3** for details on how to set the ODETTE-FTP eWay configuration parameters.

1.3 About This Document

This guide explains how to install, configure, and operate the SeeBeyond® Integrated Composite Application Network Suite™ (ICAN) ODETTE-FTP eWay Intelligent Adapter.

1.3.1 What's in This Document

This document includes the following chapters:

- **Chapter 1 "Introducing the ODETTE-FTP eWay"**: Provides an overview description of the product as well as high-level information about this document.
- **Chapter 2 "Installing the ODETTE-FTP eWay"**: Describes the system requirements and provides instructions for installing the ODETTE-FTP eWay.
- **Chapter 3 "Configuring the ODETTE-FTP eWay"**: Provides instructions for configuring the eWay to communicate with ODETTE-FTP.
- **Chapter 4 "Implementing the ODETTE-FTP eWay"**: Provides instructions for installing and running the sample Projects.
- **Appendix A"The ODEX Application"**: This appendix explains the ODEX application and what is needed for it to operate in conjunction with the ODETTE-FTP eWay.

1.3.2 Scope

This document describes the process of installing, configuring, and running the ODETTE-FTP eWay.

This document does not cover the Java methods exposed by this eWay. For information on the Java methods, download and view the ODETTE-FTP eWay Javadoc files from the Enterprise Manager.

1.3.3 Intended Audience

This guide is intended for experienced computer users who have the responsibility of helping to set up and maintain a fully functioning ICAN Suite system. This person must also understand any operating systems on which the ICAN Suite is installed and must be thoroughly familiar with Windows-style GUI operations.

1.3.4 Document Conventions

The following writing conventions are observed throughout this document.

Table 1 Writing Conventions

Text	Convention	Example
Button, file, icon, parameter, variable, method, menu, and object names.	**Bold** text	▪ Click **OK** to save and close. ▪ From the **File** menu, select **Exit**. ▪ Select the **logicalhost.exe** file. ▪ Enter the **timeout** value. ▪ Use the **getClassName()** method. ▪ Configure the **Inbound** File eWay.
Command line arguments and code samples	`Fixed` font. Variables are shown in ***`bold italic`***.	`bootstrap -p` ***`password`***
Hypertext links	**Blue** text	**http://www.seebeyond.com**

1.3.5 Screenshots

Depending on what products you have installed, and how they are configured, the screenshots in this document may differ from what you see on your system.

1.4 Related Documents

The following SeeBeyond documents provide additional information about the ICAN product suite:

- *eGate Integrator User's Guide*
- *SeeBeyond ICAN Suite Installation Guide*

1.5 SeeBeyond Web Site

The SeeBeyond Web site is your best source for up-to-the-minute product news and technical support information. The site's URL is:

http://www.seebeyond.com

1.6 Feedback

We appreciate your feedback. Please send any comments or suggestions regarding this document to:

docfeedback@seebeyond.com

1.7 Related Documents

The following SeeBeyond documents provide additional information about the ICAN product suite:

- *eGate Integrator User's Guide*
- *SeeBeyond ICAN Suite Installation Guide*

Chapter 2

Installing the ODETTE-FTP eWay

This chapter explains how to install the eWay Intelligent Adapter for ODETTE-FTP.

What's in This Chapter

2.1 Supported Operating Systems

The ODETTE-FTP eWay is available on the following operating systems:

- Windows 2000, Windows XP, Windows Server 2003
- HP-UX 11.0 and 11i (PA-RISC)
- IBM AIX 5.1L and 5.2

2.2 System Requirements

The system requirements for the ODETTE-FTP eWay are the same as for eGate Integrator. Refer to the *SeeBeyond ICAN Suite Installation Guide* for a complete listing of system requirements. It is also helpful to review the **Readme.txt** file for additional requirements prior to installation.

Note: *To enable Web services, you must also install and configure eInsight.*

2.3 External System Requirements

The ODETTE-FTP eWay supports Data Interchange's ODEX high-speed asynchronous communications software application. The eWay uses this application to interface with OFTP. To use the eWay, you must have ODEX installed on an eGate Logical Host with the eWay. ODEX must also be operational and accessible to the eWay.

You can also visit Data Interchange's Web site at the following URL:

http://www.dip.co.uk/Display.aspx?page=products/Odex

Note: *Internet links may change after the date of publication. If Data Interchange has changed this URL, you can search their Web site for the appropriate Web pages.*

As explained earlier in this chapter, you can use either of two available ODEX products with this eWay. The ODETTE-FTP eWay supports the following external systems:

- **ODEX Plus** (version 3.4): UNIX
- **ODEX Professional** (version 3.4.1): Windows

See the appropriate ODEX user's guides for details on product differences, as well as explanations of system requirements, operation, and procedures on how to install and operate this application. The eWay interfaces with both ODEX products via the applications' batch interfaces.

2.3.1 Installing ODEX

See the appropriate ODEX user's guides for complete information on how to install ODEX. Be sure to correctly configure ODEX to meet your communication needs then configure the ODETTE-FTP eWay to conform to your ODEX parameters. It is helpful to keep a record of your ODEX configuration information. Enter configuration parameters for the ODETTE-FTP eWay according to your ODEX configuration.

2.4 Before You Install

- Open and review the **Readme.txt** for the ODETTE-FTP eWay for any additional information or requirements, prior to installation. The **Readme.txt** is located on the installation CD-ROM.
- Exit all Windows programs before running the setup program, including any anti-virus applications.
- You must have Administrator privileges to install this eWay.

2.5 Installing the eWay Product Files

The installation process includes:

- Installing the ICAN Repository.
- Uploading products to the Repository (including the ODETTE-FTP eWay, documentation, sample files, and Javadocs).
- Downloading components (including the Enterprise Designer and Logical Host) from the Repository.
- Updating products in the Enterprise Designer using the Update Center Wizard.

To install the ODETTE-FTP eWay

1. Follow the instructions for installing ICAN in the *SeeBeyond ICAN Suite Installation Guide*.
2. After uploading the **eGate.sar** file to the Repository, upload the following additional product files:
   - **ProductFileName.sar** (to install the ODETTE-FTP eWay)
   - **FileeWay.sar** (to install the File eWay, used in the sample Projects)
   - **ProductFileNameDocs.sar** (to install the ODETTE-FTP eWay documentation)

Note: *These files may not be located on the same installation disc as the **eGate.sar** file.*

To install the ODETTE-FTP eWay Samples and Javadocs

1. From the Documentation tab of the Enterprise Manager, click **ODETTE-FTP eWay** to view the list of files available for this product.
2. Click **Download Sample** to open the **ProductFileNameSample.zip** file.
3. Use WinZip to extract the sample files to the desired location.
4. Click Download Javadocs to open the **ProductFileNameJavadoc.zip** file.
5. Use WinZip to extract the Javadocs files to the desired location.

   After you complete the process of installing the Repository, Logical Host, and Enterprise Designer (as described in the *SeeBeyond ICAN Suite Installation Guide*), refer to **Chapter 4** for instructions on importing the sample Project into your Repository via the Enterprise Designer.

2.6 After You Install

Once the eWay is installed and configured, it must then be incorporated into a Project before it can perform its intended functions. See the eGate Integrator User's Guide for more information on incorporating the eWay into an eGate Project.

Chapter 3

Configuring the ODETTE-FTP eWay

This chapter describes how to set the properties of the ODETTE-FTP eWay.

What's in This Chapter

3.1 Properties of ODETTE-FTP eWay

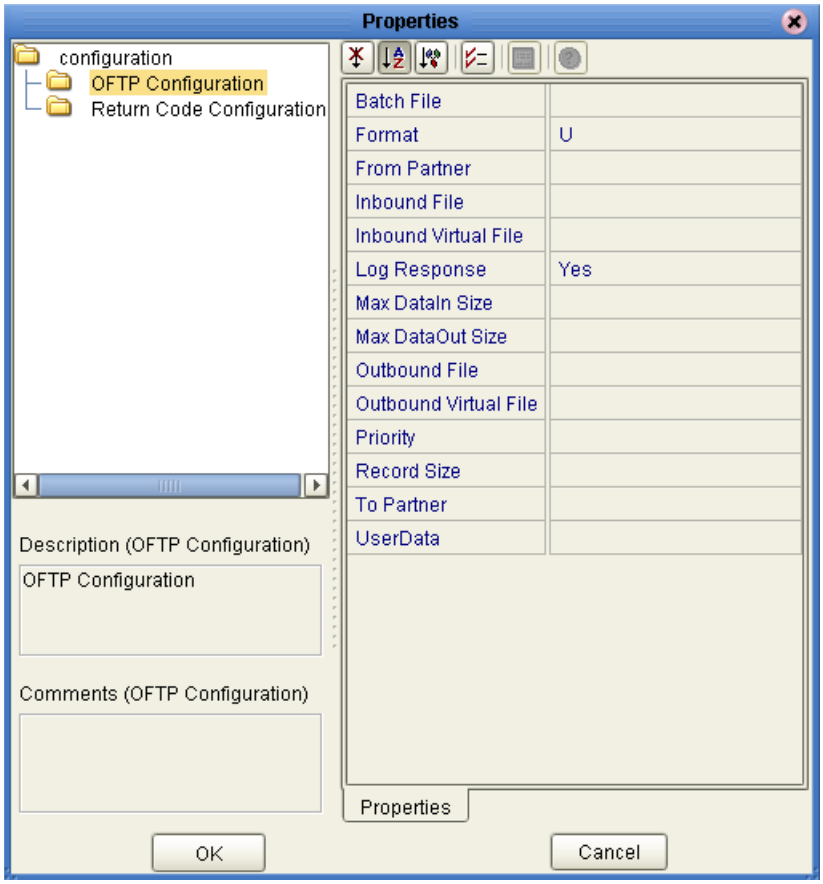
The following parameter descriptions are used for you to enter the necessary information, on the Properties window, for the eWay to establish a connection to the external application.

3.1.1 Setting the Properties of the ODETTE-FTP eWay for OFTP Configuration

The Property settings define the properties used to interact with the external database.

Note: *Not all parameters are supported in the current release, please contact SeeBeyond for more information.*

Figure 2 The eWay Properties - OFTP Configuration



Batch File

Description

Specifies the full path pointing to the ODEXBAT command file on the Logical Host. Because the format is platform dependent, it is important to pay attention to drive letters and path component separators. While it is not mandatory, it can be invoked by calling the ODEX OTD method **invoke()** from within a Collaboration.

Required Values

Optional; if used, enter the valid path location of the **odexbat.exe** file, on a Logical Host.

Format

Description

Specifies the format of a file to be sent. The file can contain unformatted, text, fixed-length, or variable-length records.

Required Values

U for unformatted, **T** for text, **F** for fixed length, and **V** for variable length.

From Partner

Description

Specifies the Local ID of your trading partner from whom you receive data. This ID must be a preconfigured trading partner in the ODEX User Directory, corresponding to LocalCode of ODEX. This is only used to refer to a trading partner in your Collaboration; it will never go out to your trading partner in any exchanged message.

Required Values

A valid ODEX Local Code.

Inbound File

Description

Specifies the name of a file inbound from ODEX. Note that it is a file name only and it does not include the path name of the parent directory. The parent directory must be given in the Inbound Box parameter.

Required Values

A valid file name.

Inbound Virtual File

Description

Specifies the default virtual file name (VFN) used for extracting received files from ODEX into the Inbound Box. For example, a newly received file from a particular trading partner can be extracted with a VFN, such as **INVOICE,** so that you know your trading partner sent an Invoice to you.

Required Values

A valid file name.

Log Response

Description

This flag indicates whether the response file is appended to the eWay log. If it is set to **Yes**, the response file information is added, if set to **No**, the file information for each batch execution is overwritten.

Note: *You must turn on the **TRACE** flag for the eWay to be sure the **odexbat.exe** response file is added to the eWay's log file.*

Required Values

Yes or **No**. The default is **Yes**.

Max DataIn Size

Description

Specifies the maximum size for incoming data, that is, the inbound payload node (DataIn) in the OTD. This node is used to hold the extracted inbound file from ODEX. The node can be copied to another destination, for example, to an Intelligent Queue (IQ). This parameter is a safeguard to prevent a huge payload from using up too much memory.

Required Values

An integer from 1024 to 864000.

Max DataOut Size

Description

Specifies the maximum size for outbound payload DataOut, when the maximum is violated, an error is generated. This is a safeguard to prevent huge payloads using up memory.

Required Values

An integer from 1024 to 864000.

Outbound File

Description

Specifies the name of the file outbound to ODEX (to be scheduled by ODEX and sent out to a trading partner). It does not include the path name of the parent directory. The parent directory must be given in the Outbound Box parameter. The Outbound Box is the full path pointing to a folder on the Logical Host, where all out going files are located. In ODEX, this corresponds to **PCFile**, see ODEX documentation for more information.

Required Values

A valid file name.

Outbound Virtual File

Description

Specifies the default OFTP virtual file name (VFN). Used for applying a name to the file that is scheduled to be sent out. This file name is usually agreed upon between trading partners. For example, INVOICE101 could be the file name your trading partner expects for a specific type of invoice data-bearing file. The names for virtual files must conform to OFTP specifications.

Required Values

A valid OFTP virtual file name.

Priority

Description

Specifies the ODEX priority code for sending a file. The codes range from 1 to 9, with 1 as the highest priority.

Required Values

An integer from 1 to 9.

Record Size

Description

If the format is fixed length (**F**), then this parameter gives the record size of the file transferred.

Required Values

An integer from 0 to 99999; optional, only use this parameter for fixed-length files.

To Partner

Description

Specifies a local ID for one of your trading partners to whom you send data. This ID must be a preconfigured trading partner in the ODEX User Directory. The ID must correspond to a Local Code in ODEX. This ID is used to refer to a trading partner in your Collaboration and never goes out to your trading partner in any exchanged message. See the appropriate ODEX user's guides for a detailed description of the Local Code.

Required Values

A valid ODEX Local Code ID.

User Data

Description

Refers to OFTP UserData, an eight-character string whose specific usage is agreed upon between the trading partners exchanging data. This is an optional parameter used when scheduling a file to ODEX.

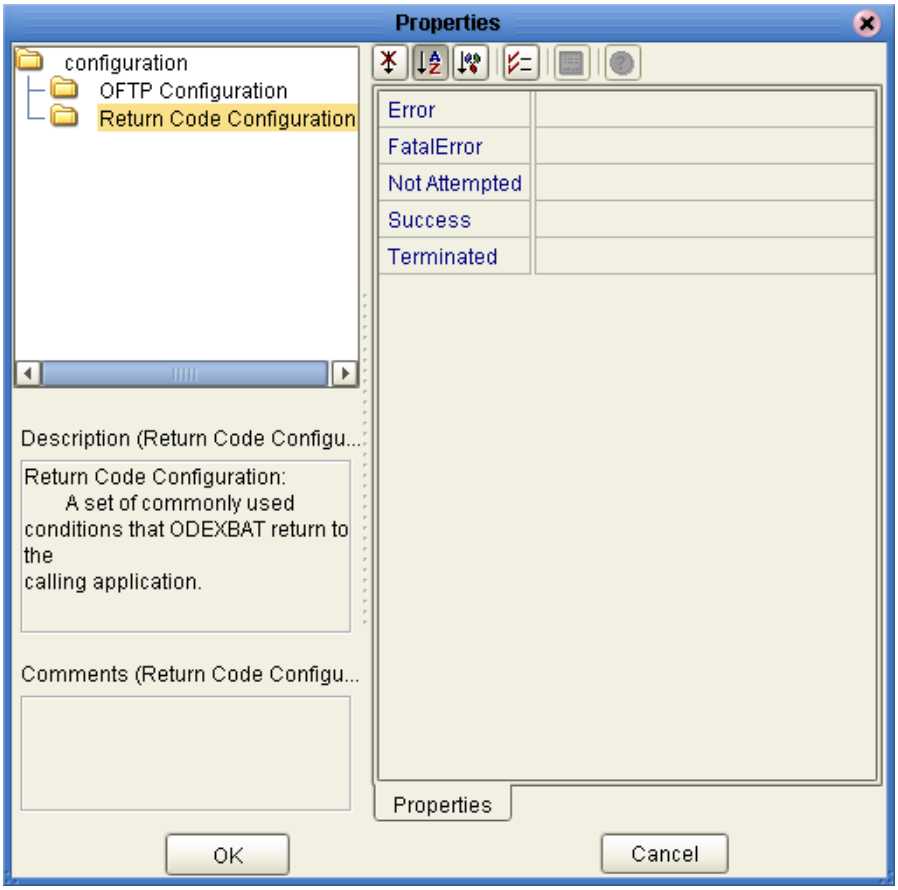
Required Values

An eight-character string; optional.

3.1.2 Setting the Properties of the ODETTE-FTP eWay for Return Code Configuration

The parameters in this section allow you to set the numbers for commonly used conditions that **odexbat.exe** returns to the calling application.

Figure 3 The eWay Properties - Return Code Configuration



Error

Description

Specifies the Return Code indicating that the batch command has been executed with an error or warning.

Required Values

A valid integer. For **ODEXBAT** configuration, this parameter must be set to 4.

FatalError

Description

Specifies the Return Code indicating that the batch command failed because of a fatal error.

Required Values

A valid integer. For **ODEXBAT** configuration, this parameter must be set to 8.

Not Attempted

Description

Specifies the return code indicating that the batch command was not attempted. Usually this return results from a file syntax error.

Required Values

A valid integer. For **ODEXBAT** configuration, this parameter must be set to 12.

Success

Description

Specifies the return code indicating that the batch command has been executed successfully.

Required Values

A valid integer. For **ODEXBAT** configuration, this parameter must be set to 0.

Terminated

Description

Specifies the return code indicating that the batch command processor has been terminated for some reason, for example, stopped by an operating-system level command.

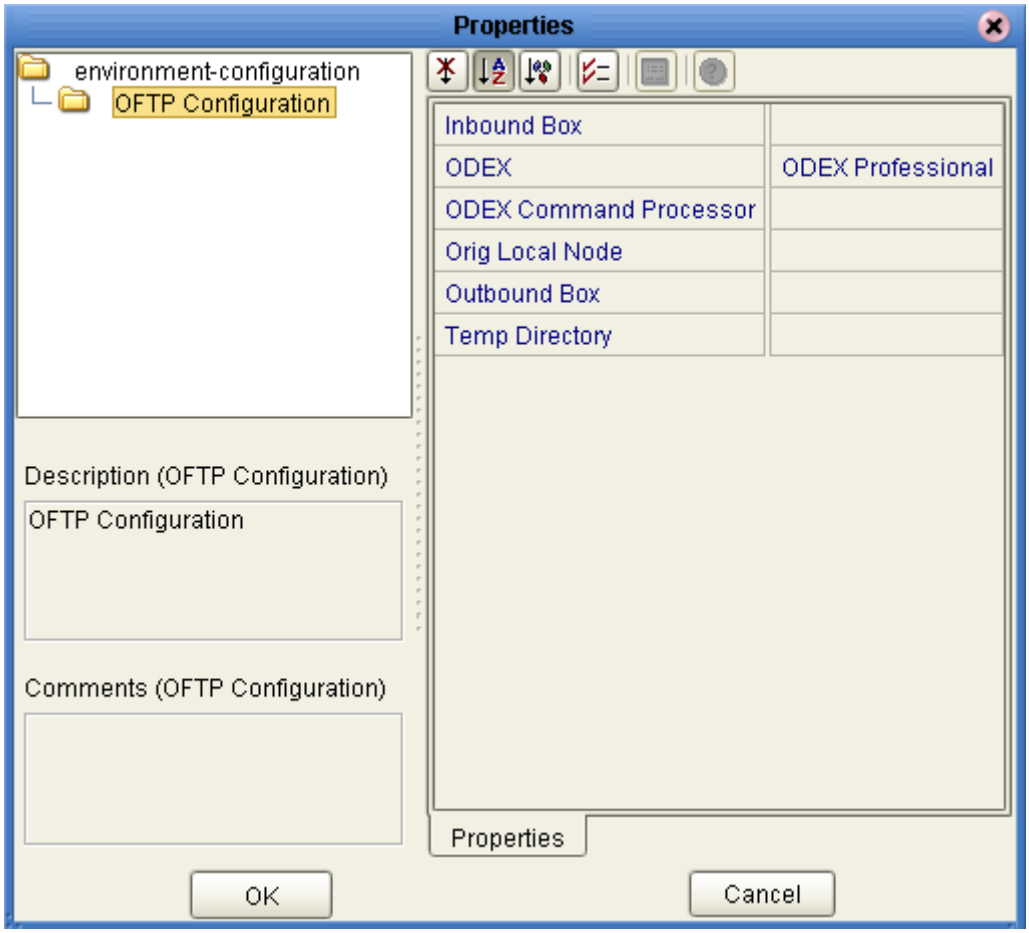
Required Values

A valid integer. For **ODEXBAT** configuration, this parameter must be set to 1.

3.1.3 Setting the Environment Properties of the ODETTE-FTP eWay

The parameters in the ODETTE-FTP Environment section allow you to configure characteristics of the ODEX OTD file (based on **oftp.xsc**), that is, the default values needed when scheduling an outgoing file to ODEX or extracting an incoming file from ODEX.

Figure 4 The eWay Properties - Environment Configuration



Inbound Box

Description

Specifies the full directory path (on a Logical Host) pointing to the local folder to be used as storage when extracting received files from ODEX. This path, concatenated with Inbound File, forms the target file into which ODEX extracts the received file. In ODEX, this target file is usually referred to as **PCFile**. See ODEX documentation for further details.

Required Values

A valid directory on an eGate Logical Host.

ODEX

Description

This eWay is based on the ODEX batch interface. There are slight batch syntax differences between the two ODEX products, ODEX Professional (Windows) and ODEX Plus (UNIX). You must indicate which product you are using.

Required Values

ODEX Professional (default) or ODEX Plus.

ODEX Command Processor

Description

Specifies the full path (on a Logical Host) pointing to the **odexbat.exe** executable. Because the format is platform dependent, it is important to pay attention to drive letters and path component separators etc.

Required Values

Valid path location on a Logical Host.

Orig Local Node

Description

Specifies the Local Code ID for the File Node level originator in the ODEX OTD. When a file is sent, the originator can be identified using the OTD's Internal Node information (see **User Directory Configuration** on page 47). Your internal node is your own electronic data interchange (EDI) code.

For large organizations, departmental or sectional information may be needed to further identify the originator of a file. See the appropriate ODEX user's guides for more information.

Required Values

A valid ODEX Local Code ID.

Outbound Box

Description

Specifies the full path of the local folder where the Outbound File for ODEX is located. This path, concatenated with the Inbound File, forms the source file that is scheduled to ODEX. In ODEX, this source file is usually referred to as **PCFile** in the **SNDFILE** batch command.

Note: *This is a file name only and does not include the path name of the parent directory. The parent directory must be given in the Outbound Box parameter.*

Required Values

A valid file name.

Temporary Directory

Description

Specifies the full path of the working eWay directory to buffer data files and generated ODEX batch command files. This directory is used for temporarily depositing intermediate files, such as the working file for holding the contents of the payload nodes in the OTD.

Required Values

A valid directory on a Logical Host. You must create this directory before configuring the eWay.

Chapter 4

Implementing the ODETTE-FTP eWay

This chapter explains the eWay's implementation and provides sample Projects to help you understand how to implement the eWay Intelligent Adapter for ODETTE-FTP in a production environment. Implementation examples and additional information for the ODETTE-FTP eWay samples are included on your installation CD-ROM.

Each of the eGate Integrator sample Projects allow you to observe and/or create end-to-end data-exchange scenarios involving eGate, the eWay, and the ODEX application.

Additional sections in this chapter explain the ODETTE-FTP (OFTP) Object Type Definition (OTD) structure, ODEX configuration, and other information you need to know in implementing this eWay.

What's in This Chapter

4.1 Sample Implementations

All the sample Projects in this section use one ODETTE-FTP eWay Connection configuration that is consistent with the corresponding ODEX configuration. This section provides an overview of some sample implementations.

Note: *See* **ODEX Configuration** *on page 47 for information on configuring ODEX with the eWay. For complete configuration information, see the appropriate ODEX user's guides.*

4.1.1 ODEX Configuration for Examples

There are two ODEX servers configured on a LAN for demonstration purposes.

Note: *See* **ODEX Configuration** *on page 47 for information on where you enter these settings and what they mean.*

All the examples exchange data between a current site (10.1.191.48) with a Windows operating system (OS) and a remote site (10.1.201.27) with a UNIX OS. The complete ODEX configuration is not explained here. However, you do need the following ODEX configuration information to understand the OTD configuration:

Current Site

- SELF (INTERNAL NODE):
- DISPLAY NAME: SeeBeyond
- SSID: O093120780410000000SBYN01
- LOCALCODE: DEFAULT INTERNAL NODE

Trading Partner (external network node):

- DISPLAY NAME: ABC
- SSID: O093120780420000000SBYN02
- LOCALCODE: TRAD0001
- CALLED ADDRESS: 10.1.201.27

Remote Site

- SELF (INTERNAL NODE):
- DISPLAY NAME: ABC
- SSID: O093120780420000000SBYN02
- LOCALCODE: DEFAULT INTERNAL NODE

Trading Partner (external network node):

- DISPLAY NAME: SeeBeyond
- SSID: O093120780410000000SBYN01
- LOCALCODE: TRAD0002 (has nothing to do with LOCALCODE in the other site)
- CALLED ADDRESS: 10.1.191.48

4.1.2 OTD Configuration

The examples mimic a situation where the ODETTE-FTP eWays run on the current site and exchange data with a remote peer/trading partner.

Note: *Note: See* **Chapter 3*"Configuring the ODETTE-FTP eWay"*** *for details on how to configure the eWay parameters.*

Choose applicable configuration parameters for ODETTE-FTP OTD Configuration from the following list:

- ODEX: ODEX Professional
- Odex Batch Processor: C:\EDI32\ODEXBAT.EXE
- Batch File: C:\WORK_AREA\ODEX_TEST\SCHFILE.CMD
- Outbound Box: C:\WORK_AREA\ODEX_TEST\OUTGOING_BOX
- To Partner: TRAD0001
- Outbound File: INVOICE101.DAT
- Outbound Virtual File: INVOICE101
- Inbound Box: C:\WORK_AREA\ODEX_TEST\INCOMING_BOX
- From Partner: TRAD0001
- Inbound File: ORDER101.DAT
- Inbound Virtual File: ORDER101

4.2 Locating and Importing the Sample Projects

The eWay sample Projects are included in the **OdetteFTPeWay.sar** file. This file is uploaded separately from the sar file during installation.

Once the **OdetteFTPeWay.sar** file is uploaded to the Repository, you can begin downloading the sample Projects using the **DOCUMENTATION** tab in the Enterprise Manager to a folder of your choosing.

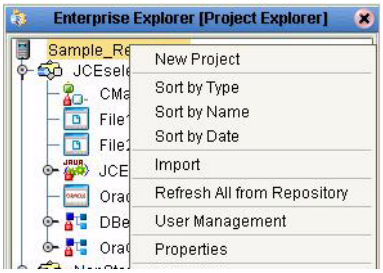
Before using the sample Project, you must first import it into the SeeBeyond Enterprise designer using the Enterprise Designer Project Import utility.

Note: *eInsight is a Business Process modeling tool. If you have not purchased eInsight, contact your sales representative for information on how to do so.*

To Import a Sample Project:

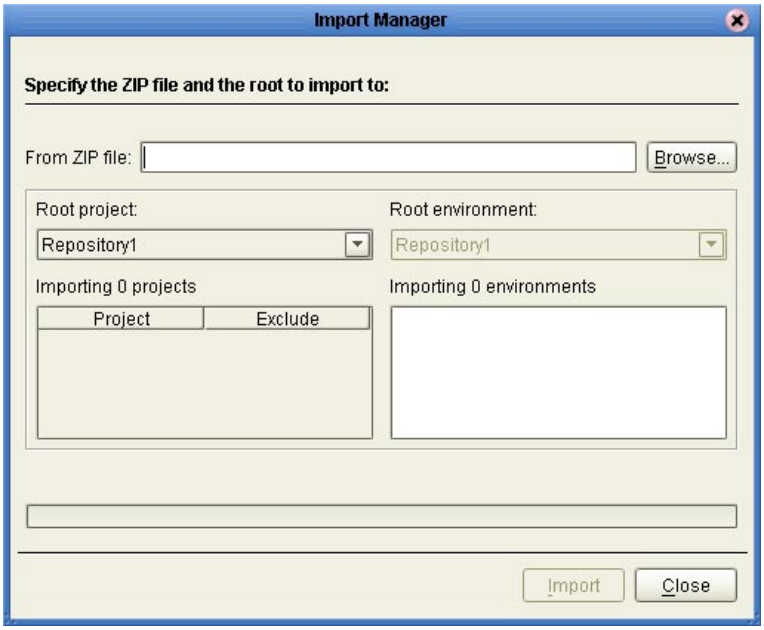
1 From the Enterprise Designer's Project Explorer pane, right-click the Repository and select **Import** (as displayed in Figure 5).

Figure 5 Importing the sample Project



2 In the **Import Manager** window, browse to the directory that contains the sample Project zip file.

Figure 6 Select the Project file



3 Select the sample file and then click **Open**.

4 Click the **Import** button. If the import is successful, then click the **OK** button on the **Import Status** window.

5 Close the Import Manger window.

4.3 Working with the ODETTE-FTP Sample Projects

The ODETTE-FTP includes two samples. Both samples demonstrate reading records from input files, inserting these records into the appropriate ODETTE-FTP tables, and writing the results to the output file(s).

The first sample queries data with the ODETTE-FTP system. The results are written to an output file using the File eWay. For complete instructions for installing and running the OFTP JCE sample, see **Using the Sample Project in eGate** on page 30.

The second sample uses an eInsight BPEL business process query data with ODETTE-FTP. The results are written to an output file using the File eWay. For complete instructions for installing and running the OFTP BPEL sample, see **Using the Sample Project in eInsight** on page 32.

The steps required to run the sample Projects include:

- Setting the Properties
- Creating the Environment Profile
- Deploying the Project
- Running the Sample

Components

The components of the sample Projects include:

- Inbound Fileeway
- Outbound Fileeway
- ODETTE-FTP eway
- One DTD for marshalling the methods in the querysent operation
- ODEX professional (On Windows) should be started and running
- ODEX plus(On UNIX) should be started and running

4.3.1 Setting the Properties

Each sample Project contains properties accessible either through the File or ODETTE-FTP eWay, located on the Project Explorer Connectivity Map, or from the ODETTE-FTP eWay External System, located in the Environment.

To Configure File eWays:

1. On the Connectivity Map, double-click the **Inbound File eWay**
2. Select **Inbound File eWay** in the Templates dialog box and click **OK**.
3. The **Properties** window for the Inbound File eWay opens. Modify the parameter settings for your system. Change the Directory and Input file name to match the location and name of the sample data file.
4. Click **OK** to close the **Properties** window.
5. On the Connectivity Map, double-click the Outbound File eWay, select **Outbound File eWay** in the templates dialog box and click **OK**. The **Properties** window for the Inbound File eWay opens.
6. Modify the required parameter settings for your system, including the target Directory and Output file name.
7. Click **OK** to close the **Properties** window.

To Configure the ODETTE-FTP eWay:

1. On the Connectivity Map, double-click the ODETTE-FTP eWay. The **Properties** window for the ODETTE-FTP eWay opens.
2. Modify the parameter settings for your system. Click **OK** to close the **Properties** window.

4.3.2 Creating the Environment Profile

An eGate Environment represents the physical system required to implement a Project. A typical Environment contains several components, including Logical Hosts, Integration Servers, Message Servers, and External Systems. Environments are created using the Enterprise Designer's Environment Explorer

To Create a New Environment:

1. On the Environment Explorer, highlight and right-click the eWay profile.
2. Select Properties, and enter the configuration information required for the eWay. for more information, see **Setting the Environment Properties of the ODETTE-FTP eWay** on page 21.

4.3.3 Deploying the Project

For instruction on the steps required to deploy a Project, see the *eGate Integrator User's Guide*.

4.3.4 Running the Sample

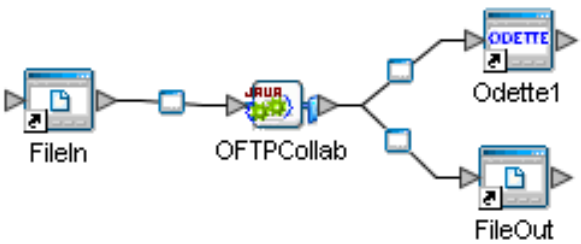
For instruction on the steps required to run the Sample Project, see the *eGate Integrator Tutorial*.

4.4 Using the Sample Project in eGate

In this sample, the File eWay triggers the OFTP Collaboration to send a query for status information (of received files) to the ODETTE-FTP eway. If the batch status is true then it returns the status information of the received files from ODEX plus. If the batch status is false it returns to the output file. The summary of the query result is then published and sent to the File eWay to write to the output file.

Follow the instructions given in **Locating and Importing the Sample Projects** on page 27 for importing the **OFTP_QueryRcv_JCE.zip** file contained in the eWay sample folder on the installation CD-ROM.

Figure 7 OFTP JCE sample



To work with the sample Project, follow the instructions given in the *eGate Integrator Tutorial*.

Note: *The Properties for this sample are the same as mentioned in the BPEL (eInsight) sample. The difference is that the BPEL sample queries the status information of files that have already been sent (and files which are scheduled to be sent) from ODEX Professional while the JCE (eGate) sample queries the status information of files recieved from ODEX plus.*

JCE Sample Code

```
public void receive( com.stc.connector.appconn.file.FileTextMessage input,
com.stc.connector.application.odetteext.ODEXOTD Odette_1,
com.stc.connector.appconn.file.FileApplication FileClient_1 )

    throws Throwable

  {

    String temp = "";

    Odette_1.queryRecv();

    logger.debug( "Recv File Count =  " + Odette_1.getRecvFileCount() + "Odex BatchStatus:
" + Odette_1.getBatchStatus().getSuccess() );

    if (Odette_1.getBatchStatus().getSuccess()) {

      temp += "BatchStatus is success";

      for (int i = 0; i < Odette_1.getRecvFileCount(); i++) {

        logger.debug( "\n FILE RECV FROM:" + Odette_1.getRecvFiles( i ).getTRDName()
+ "VPN: " + Odette_1.getRecvFiles( i ).getVirtFile() + "EERP: " + Odette_1.getRecvFiles( i
).getEERPSent() + "index = " + i );

        temp += "\n FILE RECV FROM:" + "\n Attempted: " + Odette_1.getRecvFiles( i
).getAttempted() + "\n EERPSent:" + Odette_1.getRecvFiles( i ).getEERPSent() + "\n
EERPSentTime:" + Odette_1.getRecvFiles( i ).getEERPSentTime() + "\nFailed:" +
Odette_1.getRecvFiles( i ).getFailed() + "\nFileRecv:" + Odette_1.getRecvFiles( i
).getFileRecv() + "\nFileRecvTime:" + Odette_1.getRecvFiles( i ).getFileRecvTime() +
"\nForward:" + Odette_1.getRecvFiles( i ).getForward() + "\nLastReasonCode:" +
Odette_1.getRecvFiles( i ).getLastReasonCode() + "\nLastReasonTxt:" +
Odette_1.getRecvFiles( i ).getLastReasonTxt() + "\nLocalCode:" + Odette_1.getRecvFiles( i
).getLocalCode() + "\nLocalFile:" + Odette_1.getRecvFiles( i ).getLocalFile() +
"\nLocalFileTime:" + Odette_1.getRecvFiles( i ).getLocalFileTime() + "\nMsgType:" +
Odette_1.getRecvFiles( i ).getMsgType() + "\nProcessed:" + Odette_1.getRecvFiles( i
).getProcessed() + "\nProcessing:" + Odette_1.getRecvFiles( i ).getProcessing() + "\nRetriable:"
+ Odette_1.getRecvFiles( i ).getRetriable() + "\nStatus:" + Odette_1.getRecvFiles( i
).getStatus() + "\nTRDName:" + Odette_1.getRecvFiles( i ).getTRDName() + "\nTryCount:" +
Odette_1.getRecvFiles( i ).getTryCount() + "\nUserData:" + Odette_1.getRecvFiles( i
).getUserData() + "\nVirtFile:" + Odette_1.getRecvFiles( i ).getVirtFile() + "\nWorkFile:" +
Odette_1.getRecvFiles( i ).getWorkFile();

      }
```

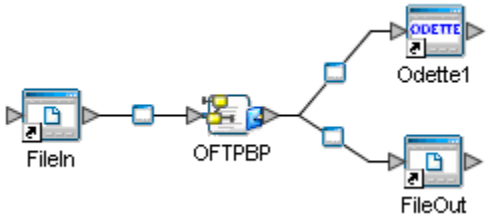
```
    } else {
      logger.debug( "CHECK NEXT RECV FILE..." );
      temp += "CHECK NEXT RECV FILE...";
    }
    FileClient_1.setText( temp );
    FileClient_1.write();
  }
```

4.5 Using the Sample Project in eInsight

In this eInsight Business Process sample, the eWay queries ODEX Professional for status information on outgoing files (for example, scheduled time, sent time, end-to-end response received time, failure, and retriability). For scheduled and sent files, the summary of the query result is published and a File eWay picks up the result then writes it to a local file.

Follow the instructions given in **Locating and Importing the Sample Projects** on page 27 for importing the **OFTPQuerySent_BPEL.zip** file contained in the eWay sample folder on the installation CD-ROM.

Figure 8 OFTP_BP sample



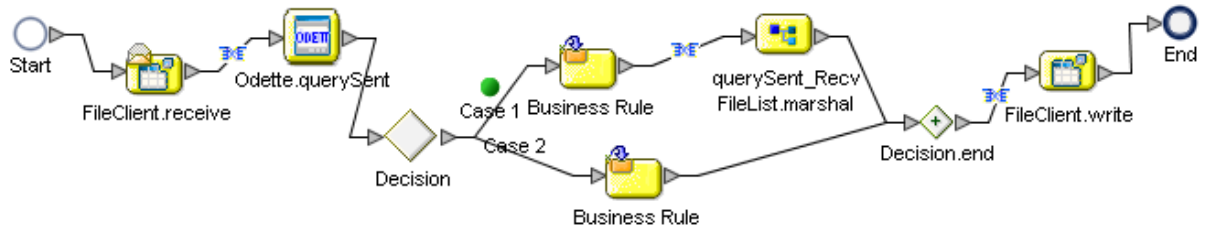
Note: *eInsight is a Business Process modeling tool. If you have not purchased eInsight, contact your sales representative for information on how to do so.*

Before recreating the sample Business Process, review the *eInsight Business Process Manager User's Guide* and the *eGate Integrator Tutorial*.

OFTP_BP

This business process describes the ODETTE-FTP process displayed in Figure 9

Figure 9 OFTP Business Process



Business Process

1. The File eWay sends a request for querying the status information of sent files to ODETTE-FTP eway.
2. If the batch status is true then it returns the status information. If the batch status is false it returns false to the output file.
3. The result is published and sent to the File eway to write to the output file.

Appendix A

The ODEX Application

To communicate with OFTP, the ODETTE-FTP eWay uses ODEX, a third-party standalone application. The eWay only launches and monitors the application, while ODEX does the actual OFTP file transfer. This appendix explains the ODEX application and what is needed for it to operate in conjunction with the ODETTE-FTP eWay

What's in This Appendix

A.1 ODEX Overview

Because the ODETTE-FTP eWay operates in conjunction with ODEX, its implementation is product specific. Correct implementation of this eWay assumes that ODEX is properly installed and configured on at least one eGate Logical Host. The eWay cannot configure ODEX; it can only keep necessary information available to communicate with and control the third-party application.

Note: *For complete details on how to set up and configure ODEX, see the appropriate documentation for the product.*

ODEX Communication with the eWay, between the ODETTE-FTP eWay's processes (eGate Collaboration thread) and ODEX processes, is needed so the eWay can operate ODEX.

A.2 ODEX Components

Figure 1 on page 8 shows an architectural overview of the ODETTE-FTP eWay as it interacts with the following basic ODEX components:

- **Virtual File (VF)**: The data transfer unit in OFTP. Before the eWay can extract a specific VF, it must know the virtual file name (VFN) to look for. In turn, the VFN is mapped to a local file (EDI or non-EDI) and pointed to by a full path in a Logical Host's system.
- **Data**: The data sent or received via ODEX. This can be an EDI file (for example, EDIFACT) or non-EDI file (for example, image or binary).

Note: *EDI files contain specific routing information in a "Service" section, but non-EDI files do not contain such information.*

- **ODEX server**: The ODEX engine that runs on the Logical Host and handles the actual data interchange jobs over OFTP.
- **dexbat.exe file**: The ODEX batch processor that communicates with the ODEX server to carry out a specific data transfer, that is, queuing a file to be sent to a destination or extracting a file from ODEX's received file Repository.
- **Batch file**: A text file containing ODEX batch commands. The text syntax for this file is based on the ODEX batch language. See the appropriate ODEX user's guides for details.
- **ODEX configuration database**: The database used by ODEX to store metadata. It is not open to another application. On Windows, Microsoft Access 2000 is used as the Repository, while on UNIX, an ISAM file is used.
- **ODEX response file**: A log file containing information on the ODEX batch file just executed. This file is overwritten each time a batch file operation (batch job) is performed. The response file is parsed by the OTD for detailed invoke information.
- **ODEX batch launch**: The startup of a batch job. The OTD implementation has methods to launch the odexbat.exe file. The Collaboration calls the methods on the OTD instance to launch an EDI job.
- **ODEX batch return**: Checking the status of a launched job. The OTD implementation has methods that check the return code and expose it to the Collaboration. The Collaboration then checks the status of the EDI job that has just been launched.

You can use the previous list as a convenient glossary of ODEX terminology. For more information, see the appropriate ODEX user's guides.

Batch Operation

Every data transfer job is initiated by a batch command file. Even though a batch command is called, for example, **SNDFILE** (see OTD Configuration and ODEX Operation on page 36), the actual OFTP data exchange is not done by the batch execution directly. Instead, odexbat.exe executes the batch file, which only delivers a

job to the ODEX server. The ODEX server then carries out the data exchange asynchronously.

A.3 OTD Configuration and ODEX Operation

The ODEX OTD contains methods that allow you to control the batch operations of the ODEX application. By using the eGate Enterprise Manager's OTD Editor GUI to drag and drop the desired methods into the OTD node structure, you can call specific ODEX batch functions, as you wish.

The ODEX OTD exposes methods that call the following ODEX batch commands:

- **SNDFILE**: Hands over a local file to the ODEX server, to be sent to a destination, for example, to a trading partner. You must configure the desired destination at the ODEX server by using the ODEX configuration GUIs. When the **SNDFILE** command finishes, ODEX has taken the request, put this job in its queue, and copied the data into its Repository. The actual data exchange happens asynchronously at the time the job's batch file executes.
- **SNDODETT**: The same as **SNDFILE**, but the file sent is an EDI-formatted file instead of a non-EDI file.
- **RCVFILE**: Extracts from the ODEX server's received Repository, a file that meets certain specified criteria.
- **RCVODETT**: The same as **RCVFILE**, but the file is an EDI-formatted file instead of a non-EDI file.

For an explanation of the ODEX OTD, see **ODEX OTD Structure** on page 37.

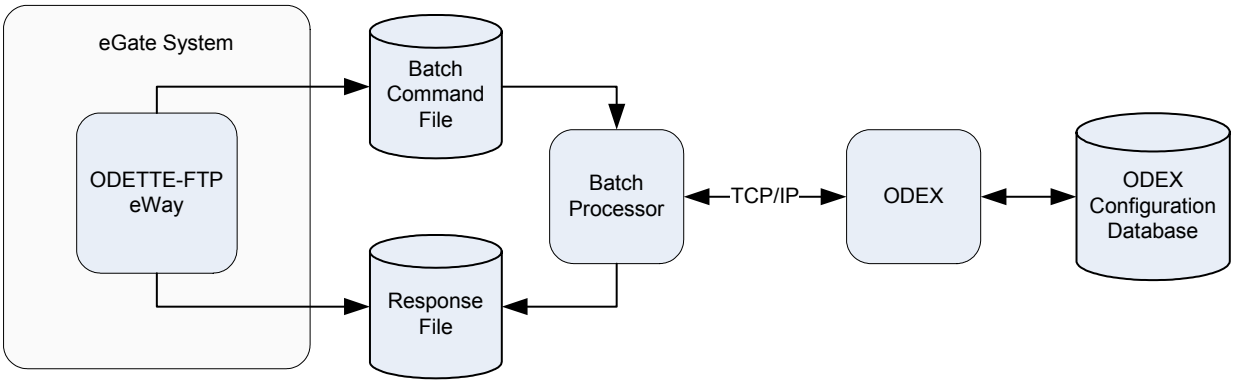
A.4 ODEX Products

The ODEX application has two available products, ODEX Plus (UNIX) and ODEX Professional (Windows). These products differ somewhat in terms of their available features and software operation. For a complete explanation of the differences, see the appropriate ODEX user's guides.

A.4.1 Hardware Configuration

Figure 1 shows a diagram of how ODEX Professional and its odexbat.exe (ODEX batch processor) can run on different machines.

Figure 1 ODEX Professional and Batch Processor

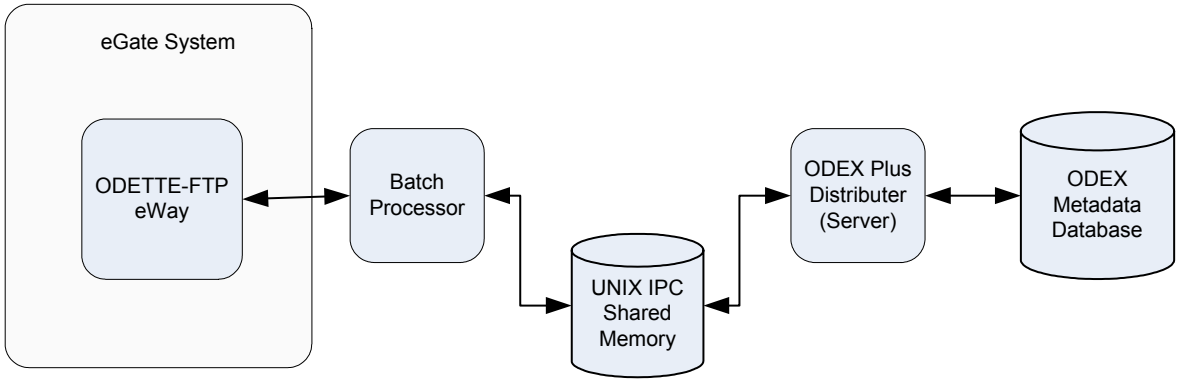


In **ODEX Professional**: Its batch processor interacts with ODEX over TCP/IP. The ODETTE-FTP eWay communicates with the **odexbat.exe** via a batch command file and a response file.

Note that there is a difference between the batch interface of ODEX Professional and that of ODEX Plus. The batch processor in ODEX Professional runs as a TCP/IP client and talks to ODEX as a TCP/IP server. This setup allows the **odexbat.exe** file to run on other machines besides the machine where ODEX is running.

Figure 2 shows a diagram of ODEX Plus and odexbat.exe running on the same machine.

Figure 2 ODEX Plus and Batch Processor

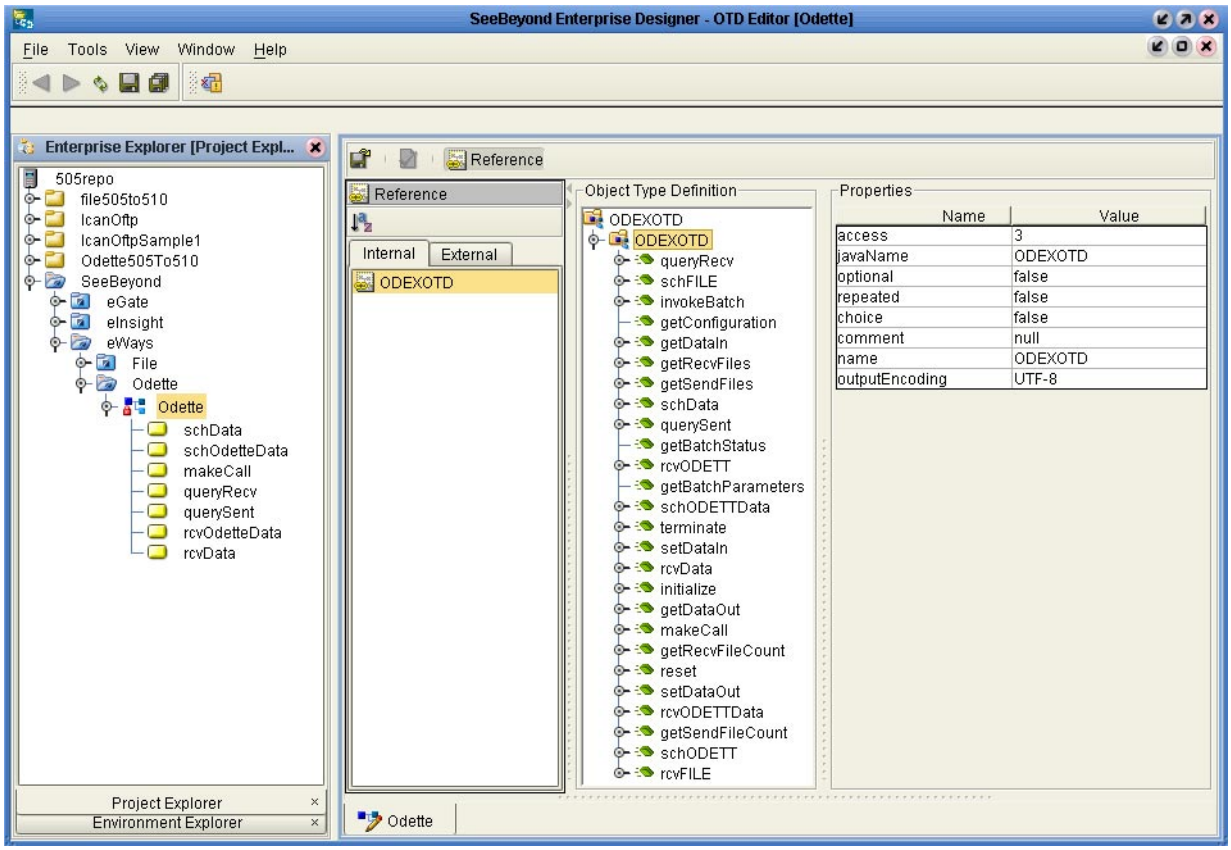


In **ODEX Plus**: ODEX and its batch processor (**odexbat.exe**) must run on the same machine.

A.5 ODEX OTD Structure

This section explains the structure and layout of the ODEX OTD (oftp.xsc). Figure 19 shows an example of the ODEX OTD in the Collaboration Editor Main window.

Figure 3 ODEX OTD Editor



A.5.1 OTD Exposed Methods

As shown in Figure 3, the ODEX OTD exposes important methods available to the eWay. As a result, ODEX attribute-related nodes exposed through the ODEX OTD are from OTD node structure.

The ODEX OTD exposes the following methods:

- **queryRecv** queries ODEX for all receiving and received file information, for example who sent the file, when it was received, and the EERP send time.
- **schFILE** generates a **SNDFILE** command and runs it. This method relays a file to ODEX so ODEX can send it. This operation is done asynchronously, that is, when the method returns a "success" code, it means the job and the data has been handed over to ODEX for ODEX to schedule. ODEX also copies the data into its own Repository, ensuring its safety. ODEX schedules the date and time it actually sends the data, as determined by how you have configured ODEX and/or the commands it receives. The data could be sent immediately or as specified by a command, for example, one hour later.
- **invokeBatch** starts the ODEX batch processor (odexbat.exe) as a native process, and waits until it sends the return code (blocking mode). The return code is from the batch file that is being executed. This code can be accessed via the **ReturnCode** node in **InvokeStatus**. This method is introduced so the user can write a

customized batch command file and run it by calling **invokeBatch**. The actual batch file executed is in the **BatchFile** node, and it defaults to the configuration parameter set under Batch File.

Note: *See* **Chapter 3** *for details on the eWay's configuration parameters.*

Before an eGate Collaboration calls **invokeBatch**, it can optionally change the actual batch file executed by setting the **BatchFile** node to a full path pointing to a batch file on the Logical Host. This feature provides more flexibility for the Collaboration.

- **getConfiguration** retrieves information on configuration parameters in the ODEX OTD sub-nodes of the top-level nodes **BatchStatus** and **BatchParameters**.
- **getDataIn** receives its contents from a file extracted from ODEX by calling **schFile** or **schODETT**. These contents can then be copied to another OTD for further routing (for example, being sent to an eGate IQ).
- **getRecvFiles** retrieves information contained in the **RecvFiles** OTD node.
- **getSendFiles** retrieves information contained in the **SendFiles** OTD node.
- **schData** schedules the content of the outbound payload, the **DataOut** node, to ODEX to be sent. The content must be in a non-EDI format.
- **querySent** returns the time stamp for when the EERP is sent for a received VF from a trading partner. This method returns a null if the file has not been sent yet.
- **getBatchStatus** retrieves the return conditions for the batch command execution.
- **rcvODETT** generates a **RCVODETT** command and runs it. This command is the same as **rcvFILE** except the data must be in a specific EDI format.
- **getBatchParameters** retrieves the parameters needed for invoking batch commands. These commands are made by OTD methods such as **makeCall**, **schFile**, **schODETT**, **rcvFile**, and **rcvODETT**.
- **schODETTData** schedules the content of the outbound payload, the **DataOut** node, to ODEX to be sent. The content must be in a specific EDI format.
- **terminate** terminates the ODEX OTD, stops the server, and cleans up all the connections.
- **setDataIn** sets the contents from a file extracted from ODEX by calling **schFile** or **schODETT**. These contents can then be copied to another OTD for further routing (for example, being sent to an eGate IQ)
- **rcvData** is the same as **rcvFILE** except the data is loaded in the inbound payload, the DataIn node, which is convenient for further routing.
- **initialize** initializes (starts) the **ODEX OTD**.
- **getDataOut** retrives contents from files extracted from ODEX by calling **schFile** or **schODETT**.
- **makeCall** generates a **DIRUPDAT** command and runs it. This command can make ODEX generate an outgoing call even if there is nothing to send out. This method is only needed when the trading partner does not initiate a session.

- **getRecvFileCount** retrieves the number of received files contained in the getRecvFileCount OTD node.
- **reset** resets (clears) the data content of the **ODEX OTD**.
- **setDataOut** sets information for the **DataOut OTD** node.
- **rcvODETTData** is the same as **rcvODETTData** except the content must be in a specific EDI format.
- **getSendFileCount** retrieves the numeric count contained in the **SendFiles OTD** node.
- **schODETT** generates a **SNDODETT** command and runs it. This command is the same as **schFILE** except the data must be in a specific EDI format.
- **rcvFILE** generates a **RCVFILE** command and run it. This method extracts the data from ODEX's Repository of received files. The actual data could have been received previously, so the extracting and receiving of the file are both asynchronous.

A.5.2 OTD Attributes

The ODEX OTD has the following attribute nodes:

- **DataOut**: The outgoing payload; this is a buffer. This node can receive its contents from another OTD, for example, an Event in an IQ. This OTD can then be sent to your trading partner (by calling **schFILE** or **schODETT**).
- **DataIn**: The incoming payload; this is a buffer. This node can receive its contents from a file extracted from ODEX (by calling **schFILE** or **schODETT**). These contents can then be copied to another OTD for further routing, for example, being sent to an eGate IQ.

Note: *Transferring send or receive payloads into and out of ODEX is implemented via a local file on the Logical Host. This file resides in a folder indicated by the configuration parameter Temp Directory. The ODEX batch command interface is used to facilitate these operations.*

- **SendFiles**: This is a repeating node, and its entries are loaded by invoking querySent. These entries contain status information for the outbound/sent files in the ODEX Repository. The status information sub-nodes contained in this node are:
  - **TRDName**: The trading partner's name, extracted from the ODEX User Directory using LocalCode.
  - **LocalCode**: The local code ID for the trading partner to whom the file is sent.
  - **WorkFile**: The FTP file name in the ODEX Repository, for the scheduled file.
  - **VirtFile**: The VFN for the scheduled file.
  - **FileSentTime**: The time when the file was sent; -1 means it has not been sent yet.
  - **FileSent**: This Boolean return indicates whether the file has been sent.

- **EERPRecvTime**: The time when the file's EERP has been returned by the destination; -1 means it has not been returned yet.
- **EERPRecv**: This Boolean return indicates the file's EERP has been returned.
- **MsgType**: The type of the first message in a file. The type is UNKNOWN if no valid message found.
- **UserData**: The OFTP userdata.
- **TryCount**: The number of attempts made by the current operation.
- **LastReasonTxt**: The text showing the last reason for the current operation on the file (see **Table 1 on page 45**).
- **LastReasonCode**: This numeric code indicates the last reason for the current operation on the file (see **Table 1 on page 45**).
- **SchTime**: The time the file is scheduled to be sent.
- **Failed**: This Boolean return indicates whether the current operation on the file has failed (and either is not allowed to retry or has exceeded a predefined retry limit).
- **Attemped**: This Boolean return indicates whether the current operation on the file has been attempted and failed (the operation can still be retried).
- **Processing**: This Boolean return indicates the file is being processed.
- **Processed**: This Boolean return indicates the file processing is complete.
- **Forward**: This Boolean return indicates the file needs to be forwarded.
- **Retriable**: This Boolean return indicates the current operation is retriable.
- **LocalFile**: The local file name that has been loaded into ODEX to be scheduled for sending.
- **LocalFileTime**: The time the local file became scheduled.
- **Status**: The ODEX file status code (see **Table 2 on page 46**).

- **SendFileCount**: This node represents the count of entries in SendFiles.
- **RecvFiles**: This is a repeating node, and its entries are loaded by invoking **queryRecv**. These entries contain status information for the inbound/received files in the ODEX Repository. The status information sub-nodes contained in this node are:
  - **TRDName**: The trading partner's name, extracted from the ODEX User Directory using LocalCode.
  - **LocalCode**: The local code ID for the trading partner to whom the file is sent.
  - **WorkFile**: The FTP file name in the ODEX Repository, for the scheduled file.
  - **VirtFile**: The VFN for the scheduled file.
  - **FileRecvTime**: The time when the file was received; -1 means it has not been received yet.
  - **FileRecv**: This Boolean return indicates whether the file has been received.

- **EERPSentTime**: The time when the file's EERP has been sent by the destination; -1 means it has not been sent yet.
- **EERPSent**: This Boolean return indicates the file's EERP has been sent.
- **MsgType**: The type of the first message in a file. The type is UNKNOWN if no valid message found.
- **UserData**: The OFTP userdata.
- **TryCount**: The number of attempts made by the current operation.
- **LastReasonTxt**: The text showing the last reason for the current operation on the file (see Table 3 on page 61).
- **LastReasonCode**: This numeric code indicates the last reason for the current operation on the file (see Table 3 on page 61).
- **Failed**: This Boolean return indicates whether the current operation on the file has failed (and either is not allowed to retry or has exceeded a predefined retry limit).
- **Attempted**: This Boolean return indicates whether the current operation on the file has been attempted and failed (the operation can still be retried).
- **Processing**: This Boolean return indicates the file is being processed.
- **Processed**: This Boolean return indicates the file processing is complete.
- **Forward**: This Boolean return indicates the file needs to be forwarded.
- **Retriable**: This Boolean return indicates the current operation is retriable.
- **LocalFile**: The local file name that has been created for a file extracted from ODEX.
- **LocalFileTime**: The time the local file was extracted.
- **Status**: The ODEX file status code (see **Table 2 on page 46**).

- **RecvFileCount**: This node represents the count of entries in **RecvFiles**.
- **BatchStatus**: This node contains all the return conditions for batch command execution, in the following sub-nodes:
  - **ReturnCode** (integer) is the return code from odexbat.exe.
  - **Success** (Boolean) is true if the return code is 0.
  - **Error** (Boolean) is true if the return code is 4.
  - **FatalError** (Boolean) is true if the return code is 8.
  - **NotAttempted** (Boolean) is true if the return code is 12.
  - **Terminated** (Boolean) is true if the return code is 1.

Note: *Note: See Table 2 on page 34 for a list and explanation of the ODEX batch processor return codes.*

- **BatchParameters**: This node contains all the parameters needed for invoking batch commands. These commands are made by OTD methods such as **makeCall**,

schFILE, **schODETT**, **rcvFILE**, **rcvODETT**. For any given method, only some of the parameters apply, so check each method for detail. The sub-nodes contained in this node are:

- **ToWhom**: The unique name for the trading partner to whom data is sent or a call is going to be made. This ID must be a valid LocalCode in your ODEX User Directory (see **"User Directory Configuration" on page 47**). The initial (default) value is taken from the configuration parameter **To Partner**.

Note: *When **restoreDefault** is called, the current value of this node is set to the value of the configuration parameter To Partner. See* **Chapter 3** *for details on the eWay's configuration parameters.*

- **OutboundFile**: The local file name to be sent to your trading partner. This file must be located in a local directory indicated by OutboundBox. The initial (default) value is taken from the configuration parameter Outbound File.
- **OutboundVFN**: The VF name (VFN) used when ODEX sends the local file. The file name's syntax must comply with the restrictions for a VFN, as defined by RFC2044 guidelines. The initial (default) value is taken from the configuration parameter **Outbound Virtual File**.

Note: *See the appropriate ODEX user's guide for a detailed explanation of RFC2044 guidelines.*

- **OutboundBox**: The full path pointing to a local system directory where an outbound file (indicated by OutboundFile) is located. The implementation of the OTD methods **schFILE** and **schODETT** locate the physical file to be sent there. The initial (default) value is taken from the configuration parameter Outbound Box.
- **OrigLocalNode**: A second-level LocalCode identifying the departmental originator, in addition to Network Node level information. The initial (default) value is taken from the configuration parameter **Orig Local Node**.

Note: *See the appropriate ODEX user's guide for a detailed explanation of the Network Node, and LocalCode.*

- **EarlyDate**: A second-level **LocalCode** identifying the departmental originator, in addition to Network Node level information. The initial (default) value is taken from the configuration parameter **Orig Local Node**.
- **EarlyTime**: The earliest date that ODEX can send a file. Use this command for scheduling a file to ODEX to be sent. The format is **YYMMDD**. The default value is the current date.
- **Format**: The format of the transfer. The valid values are **U**, **T**, **F**, and **V** (see **"Format" on page 16**). The default value is **U**, unformatted.
- **RecSize**: When the format of the transfer is **F**, RecSize is the size of the record.
- **Priority**: The priority for ODEX to schedule a file to be sent (see **"Priority" on page 19**).

- **UserData**: The OFTP UserData is an eight-character string whose usage is agreed between two trading partners exchanging data over OFTP (see **"User Data" on page 19**). The initial (default) value is taken from the configuration parameter UserData.
- **FromWhom**: The unique name for the trading partner from whom data is received and extracted. This ID must be a valid **LocalCode** in your ODEX User Directory. The initial (default) value is taken from the configuration parameter From Partner.
- **InboundFile**: The local file name to be used to hold incoming files extracted from ODEX. This file must be located in a local directory indicated by **InboundBox**. The initial (default) value is taken from the configuration parameter Inbound File.
- **InboundVFN**: The VF name (VFN) the eWay looks for to extract received files from ODEX. For example, a typical VFN could be INVOIVE101. The initial (default) value is taken from the configuration parameter Inbound Virtual File.

Note: *VFN syntax must comply with the restrictions for VFNs, as defined by the RFC2044 guidelines. It is a good idea to agree upon VFNs, with your trading partners and/ members, so the file names can have easily recognizable meanings. See the appropriate ODEX user's guide for details.*

- **InboundBox**: A full path pointing to a local system directory where an inbound file (indicated by InboundFile) is to be created. The implementation of the OTD methods **rcvFILE** and **rcvODETT** create the physical file in that directory, to hold data extracted from ODEX. The initial (default) value is taken from the configuration parameter Inbound Box.
- **VirtDate**: The date stamp of the VF to be extracted. This date is given in the format **YYMMDD**.
- **VirtTime**: The time stamp of the VF to be extracted. This time is given in the format **HHMMSS**.

Note: *Every VF carries its date-transferred and time-of-transfer information.*

- **MsgType**: If a file contains EDI data, then a specified message of **MsgType** can be extracted from it. The default is UNKNOWN indicating a non-EDI file.
- **New**: The flag indicating the latest non-EDI file extracted from ODEX. This value only applies only to **rcvFILE**.
- **Old**: The flag indicating the oldest non-EDI file extracted from ODEX. This value only applies only to **rcvFILE**.
- **BatchFile**: The full path pointing to a pre written ODEX batch command file on the Logical Host. This command/file is executed when an eGate Collaboration calls the OTD method invokeBatch. The initial (default) value is taken from the configuration parameter Batch File.

A.5.3 ODEX Last Reason Codes

Table 3 shows a list of the ODEX last reason codes.

Table 1 ODEX Last Reason Codes

Code	Name	Description
00	File Successfully processed	No problem has occurred
01	Invalid filename	The virtual file name (VFN) does not contain valid ODETTE-FTP characters. Ensure that only upper-case alphabetic characters, numbers 0 to 9 and the characters & ()-./ are used.
02	Invalid destination	The destination EDI code (file or message node) is not profiled in the ODEX User Directory.
03	Invalid origin	The origin EDI code (file or message node) is not profiled on the ODEX User Directory.
04	Storage record format not supported	The storage format for the VF cannot be supported by the destination. This problem would occur if, for example, a variable-length record was sent to a machine that could only handle fixed-length records.
05	Maximum record length not supported	The maximum record length that the destination machine can handle has been exceeded.
06	File size is too large	The file is too large for either the OFTP or the remote site to handle.
10	Invalid record count	The number of records field, stored on the EFID, does not match the number counted during the transmission.
11	Invalid byte count	The number of bytes field, stored on the EFID, does not match the number counted during the transmission.
12	Access method failure	The remote site has had a problem storing the virtual file. This code is often caused by the receiver when it is running out of disk space.
13	Duplicate file identity	The VFN, date, and time must be unique. This message states that a second VF has been transmitted with exactly the same name, date, and time as an existing file.
99	Unspecified reason code	There has been an error that does not fit into any of the above categories.

A.5.4 ODEX File Status Flags

Table 4 shows a list of the ODEX file status flags.

Table 2 ODEX File Status Flags

Flag	Description
0x80	Receiving/received file
0x40	Sending/sent file
0x20	Forwarding file
0x10	EERP sent/received status
0x08	Processing complete
0x04	Being processed
0x02	Failed but retriable
0x01	Failed; no retry allowed

A.5.5 ODEX OTD and ODEX Batch Jobs

If it is operating correctly, the ODEX server is always up and running. However, **odexbat.exe**, the ODEX batch processor, does not necessarily run continually. An eGate Collaboration starts and monitors **odexbat.exe** via the ODEX OTD.

The ODEX OTD exposes attributes and methods that allow a Collaboration to do the following ODEX batch (**odexbat.exe**) operations:

1. Set necessary parameters for starting a batch job. The Collaboration schedules a file (EDI or non-EDI) to ODEX for sending out, making an outgoing call to see whether the trading partner has something to send in, extracting received files from the ODEX Repository. The Collaboration can specify files according to predefined criteria, for example, files from certain trading partners or files having certain VFNs.
2. Start **odexbat.exe**.
3. Receive the **odexbat.exe** return code. This code is another important way for the OTD (and the Collaboration) to check on a job's status. Additional detailed return information can be extracted by parsing the response file (see step 4).

Note: *See Table 2 on page 34 for a list and explanation of the **odexbat.exe** return codes.*

4. Log the batch job execution in the **odexbat.exe** response file. This information includes the error code and any applicable error messages. The Collaboration parses this file for detailed error information based on the error-level return code.
5. Support the payload. For the convenience of the Collaboration developer, the payload nodes are exposed in the OTD so the Collaboration can route data from eGate to external systems (then to trading partners) and from external systems into eGate.

A.6 ODEX Configuration

Before you can use the eWay, ODEX must be correctly installed and configured on the appropriate Logical Host. The eWay does not configure ODEX, and it can only keep necessary information that allows it to generate and start the operation of ODEX batch files. Then, ODEX does the file transfer over OFTP to and from trading partner s. This section explains essential steps you must do and considerations to keep in mind when configuring ODEX to operate with eGate and the ODETTE-FTP eWay.

Note: *For details on how to configure the eWay Connection for the ODETTE-FTP eWay, see* **Chapter 3**.

A.6.1 User Directory Configuration

ODEX has its own User Directory feature, where you can enter your trading partner information. The ODEX User Directory dialog box is the GUI where you set up your User Directory.

When First Running ODEX

When you first fun ODEX, it prompts you for your own user information. It is especially important that you enter your own SSID, communication method, and other EDI-related information. In this way, you allow ODEX to have information necessary for your trading partners to communicate with you.

EDI Requirements

Before any EDI job can be defined or carried out, your trading partners members must be entered and organized in the ODEX User Directory. This feature becomes your EDI "phone book." You must configure your trading partners correctly in the User Directory.

This configuration includes entering their correct SSID, communication method, and all other information required for EDI transmission. ODEX calls this configuring trading partner profiling.

ODEX Plus

Figure 4 shows an example of the USer Directory dialog box in ODEX Plus.

Figure 4 ODEX Plus User Directory Dialog Box

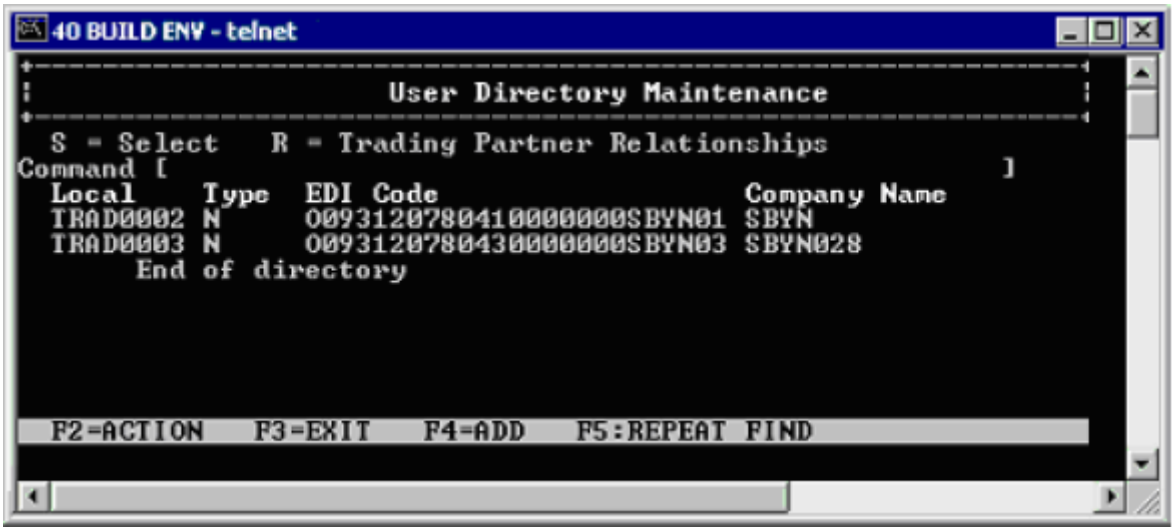


Figure 5 shows a list of trading partners s already profiled. You can use this dialog box add, delete, and update their information.

Figure 5 Updated ODEX Plus User Directory Dialog Box

A typical profile of a trading partner () includes:

- **Display Name**: SBYN
- **SSID**: O0931207804100000000SBYN01
- **Local Code**: TRAD0002

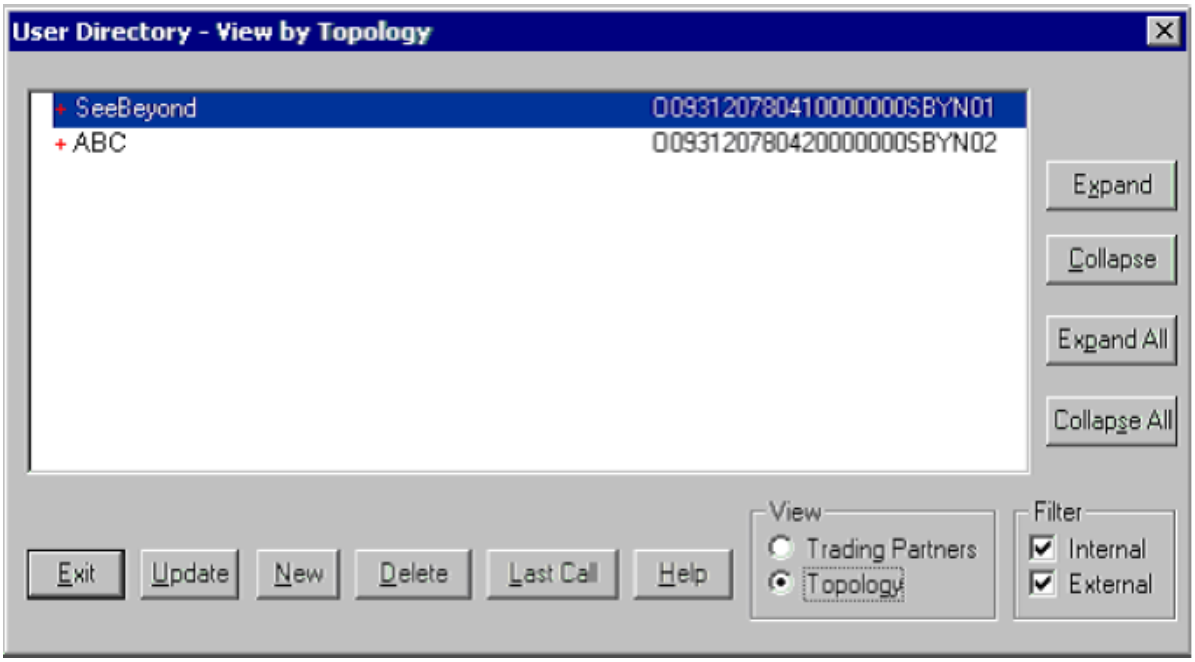
- **Network/Subnet/Address**: [T] indicating TCP/IP and the IP of the trading partner's host (here you define the communication details)

The Local Code ID is important in ODEX, because, after a trading partner is profiled, ODEX refers to it with this unique name. For example, in using the batch command, you must specify the trading partner you want to send file to by passing its Local Code ID to the batch command.

ODEX Professional

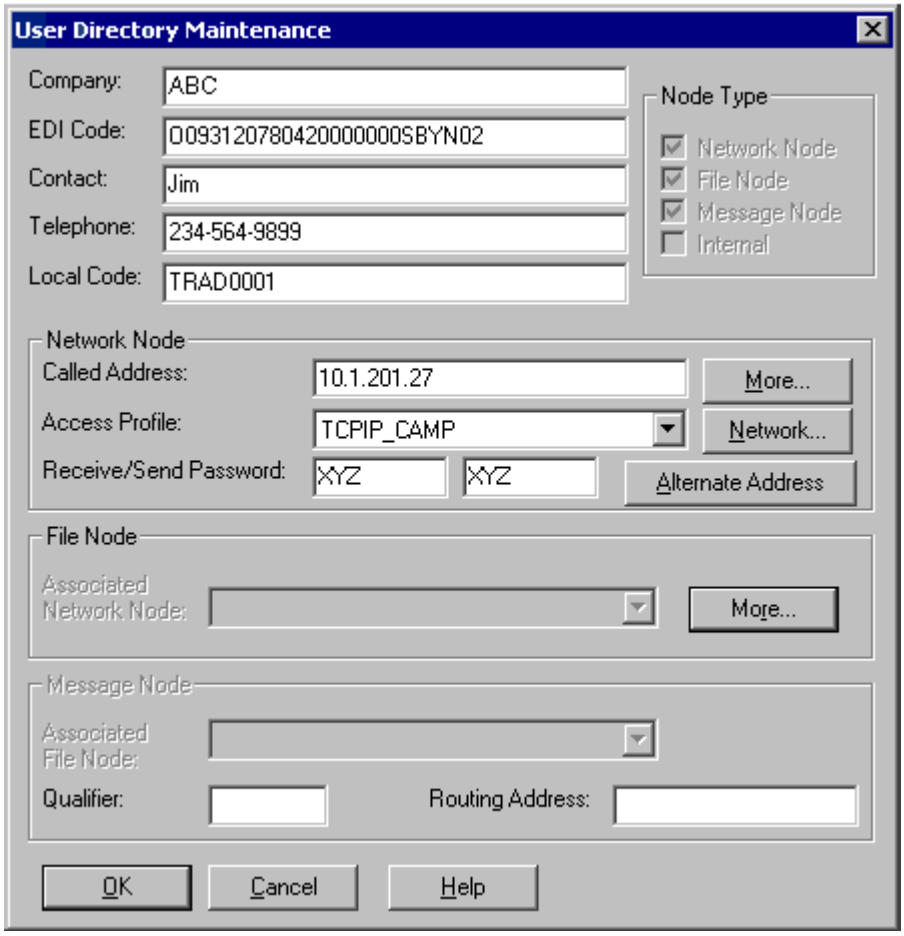
Figure 6 shows the User Directory dialog box for ODEX Professional.

Figure 6 ODEX Professional User Directory - View by Topology Dialog Box



Double-click on the name of the desired trading partner in this dialog box to display the complete User Directory entries for a given partner in the User Directory Maintenance dialog box.

Figure 7 ODEX Professional User Directory Maintenance Dialog Box



Compare the User Directory features in ODEX Plus and ODEX Professional. In Figure 6 there are data communication peers on the network, and they profile each other as trading partners.

Also note, in Figure 6, that the called IP address is 10.1.201.27. This address must be where your trading partner's EDI host is running.

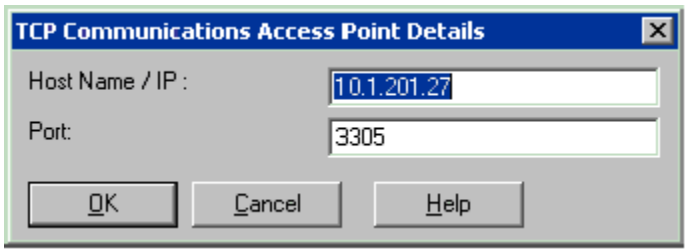
A.6.2 Basic ODEX Configurations

This section explains additional basic parameters of ODEX, which you must configure.

Communication Configuration

Figure 8 shows a TCP/IP communication configuration. For X.25 and ISDN, you need to enter additional information. See the appropriate ODEX user's guides for details.

Figure 8 ODEX TCP Communications Address Point Details Dialog Box



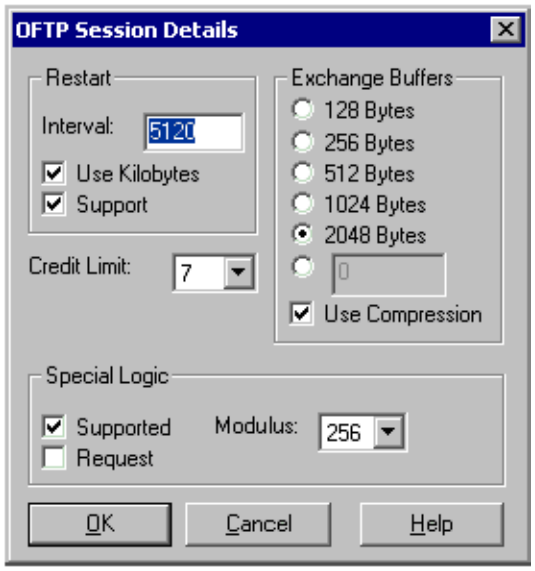
You must specify in ODEX which communication protocols you are using (see Figure 8). You also need to specify input parameters, for example, for X.25, ISDN, or TCP/IP, as agreed upon between you and your trading partners.

OFTP Configuration

OFTP parameters such as **SEND PASSWORD**, **RECEIVE PASSWORD**, **SSID**, and **SFID** are needed by the OFTP stack to start file transfer sessions and other operations. Configure this information as part of trading partner profiling.

Also use the dialog box shown in Figure 26. You can configure this information once, and you do not have to change it again unless your trading partners change or alter their system parameters.

Figure 9 ODEX Session Details Dialog Box



Additional Configuration

Keep in mind that ODEX configurations such as EDI and Communication configurations can be extremely complex. ODEX handles these operations well and provides user-friendly GUIs for their configuration. ODEX also provides excellent online Help and documentation.

Basically, the eWay only keeps a minimum set of configuration parameters in its own configuration file, so it can launch an EDI or non-EDI job exchanging data with one or more trading partners. Note that most of the elements in an EDI job are preconfigured, for example, trading partners a user can exchange data with and underlying communication protocols to be used. However, once you configure this information, data transmission and scheduling by ODEX become automatic.

A.6.3 EERP Handling

You must pay special attention to EERP support in ODEX. EERP is the OFTP method of acknowledgement to the originator (sender) of a file, that the file is received by its destination (receiver).

EERP is activated in the ODETTE-FTP eWay provided that you have configured ODEX correctly. For a given trading partner, you can configure its EERP to happen:

- When the file is received by ODEX
- When the file is copied out (extracted)
- On request (manually)

When you configure ODEX to work with the ODETTE-FTP eWay, your trading partner's EERP must be configured either on file receipt or when it is copied out. If your trading partner has one of these configurations, EERP is activated.

The OTD does not expose methods to do EERP operation. Instead, you must configure ODEX as explained in this section, so EERP can be sent at the proper time. The Collaboration can then use the queryRecv and **querySent** methods to query status information about files received or sent. This information includes the EERP status for a given file.

Note: *Note: See the appropriate ODEX user's guides for complete information on how ODEX handles EERP.*

A.7 Recovery, Data Integrity, and Security

OFTP and ODEX adequately address the need for data recovery and integrity, as well as security.

ODEX supports the full the OFTP protocol, which includes OFTP restart functionality. In the ODEX configuration, if the restart option is enabled, an interrupted transfer is picked up during the next data exchange session, at the exact point where it was left off.

In terms of security, ODEX is a public X.25 system. These systems are protected by a very rigorous security regime. It is virtually impossible to "hack" into an X.25 server and change essential attributes and information.

Note: *ODEX describes its support of these features in its own documentation. For details, see the appropriate ODEX user's guides.*

Index

W