IBM WebSphere Partner Gateway Enterprise and
Advanced Editions

Mapping Guide

Version 6.1

IBM

IBM WebSphere Partner Gateway Enterprise and
Advanced Editions

Mapping Guide

Version 6.1

Contents

About this book

Before you can use the IBM ^(R) WebSphere ^(R) Partner Gateway product to translate data, or to send or receive transactions, messages, or files, you must define certain information. This information describes how your system sends and receives data, how data is formatted in your application files and mapped to a standard, to whom you send data and from whom you receive data, and other pertinent information.

This document explains how you can use Data Interchange Services client to develop data transformation, validation, and functional acknowledgment maps that the WebSphere Partner Gateway product utilizes when processing EDI documents, XML documents, and Record Oriented Data (ROD) documents.

Audience

This document is intended for the person responsible for creating data transformation maps, validation maps, and functional acknowledgment maps using the Data Interchange Services client.

Typographic conventions

This document uses the following conventions.

Table 1. Typographic conventions

Convention	Description
`Monospace font`	Text in this font indicates text that you type, values for arguments or command options, examples and code examples, or information that the system prints on the screen (message text or prompts).
bold	Boldface text indicates graphical user interface controls (for example, online button names, menu names, or menu options) and column headings in tables and text.
italics	Text in italics indicates emphasis, book titles, new terms and terms that are defined in the text, variable names, or letters of the alphabet used as letters.
`Italic monospace font`	Text in italic monospace font indicates variable names within monospace-font text.
ProductDir	*ProductDir* represents the directory where the product is installed. All IBM WebSphere Partner Gateway product pathnames are relative to the directory where the IBM WebSphere Partner Gateway product is installed on your system.
`%text%` and `$text`	Text within percent signs (%) indicates the value of the Windows^(R) text system variable or user variable. The equivalent notation in a UNIX^(R) environment is $*text*, indicating the value of the *text* UNIX environment variable.
Underlined colored text	Underlined colored text indicates a cross-reference. Click the text to go to the object of the reference.

Table 1. Typographic conventions (continued)

Convention	Description
Text in a blue outline	(In PDF files only) An outline around text indicates a cross-reference. Click the outlined text to go to the object of the reference. This convention is the equivalent for PDF files of the "Underlined colored text" convention included in this table.
" "(quotation marks)	(In PDF files only) Quotation marks surround cross-references to other sections of the document.
{ }	In a syntax line, curly braces surround a set of options from which you must choose one and only one.
[ ]	In a syntax line, square brackets surround optional parameters.
< >	Angle brackets surround variable elements of a name to distinguish them from one another. For example, *<server_name><connector_name>*tmp.log.
/, \	Backslashes (\) are used as separators in directory paths in Windows installations. For UNIX installations, substitute slashes (/) for backslashes.

Terminology changes

Beginning with WebSphere Partner Gateway, Version 6.1, certain general terms used in previous releases were changed to improve clarity.

Table 2. General terminology changes for Version 6.1

Old term	New term	Definition
community operator	hub administrator	The *hub administrator* is a company responsible for managing the day-to-day operation of the hub community. The hub administrator maintains the hardware and software infrastructure of the hub community. The Hub Administrator profile has the widest authorities in the system. Usually, hub administrators are the system administrators who are responsible for a smooth operation of the server.
community manager	internal partner	The *internal partner* is the primary company and driving force within the hub community. This company is responsible for the purchase and construction of the hub community, including definition of the electronic business processes transacted between them and their external partners. The Internal Partner profile is usually reserved for the company that owns the server. Members or users belonging to this profile have a wide range of authorities but are usually not authorized to make system-wide changes. There can only be one Internal Partner profile for a given WebSphere Partner Gateway installation. The person functioning as the internal partner (Internal Partner profile) can also be assigned to be the hub administrator.

Table 2. General terminology changes for Version 6.1 (continued)

Old term	New term	Definition
community participant	external partner	*External partners* are the companies that do business with the internal partner via the hub community. The External Partner profile has limited options on the server. For example, an external partner (External Partner profile) can only use or view objects and documents that relate to it.
target	receiver	A *receiver* is a listening or polling component for receiving documents from internal partners or external partners. For example, you can poll a folder on the file system, a JMS queue, or an HTTP URL for documents coming from business partners or from internal systems. A receiver is used for both inbound and outbound flows.
gateway	destination	A *destination* is used by WebSphere Partner Gateway to deliver documents to an existing resource. For example, when sending a document to a partner, your own WebSphere Partner Gateway server needs a destination that points to the URL that is configured on your partner's server.

These changes in general terms necessitate the following changes to the user interface of the Community Console of WebSphere Partner Gateway.

Table 3. User interface terminology changes for Version 6.1

Old term	New term
Participant Connections	Partner Connections
Participant Profiles	Partner Profiles
Community Participant	Partner
Participant Type	Partner Type
Target	Receiver
Gateway	Destination
Source Gateways	Return Destinations
Target Gateways	Destinations
Gateway Type	Operation Mode
Gateway Transport	Destination Transport
Document Flow Definition	Document Definition
Document Flow	Document Type
Interchange Qualifier	Business Identifier (EDI Interchange Qualifier)
Interchange Identifier	Business Identifier (EDI Interchange Identifier)
Group Application	Group Application (GS)

Related documents

The complete set of documentation available with this product includes comprehensive information about installing, configuring, administering, and using WebSphere Partner Gateway Enterprise and Advanced Editions.

You can download this documentation or read it directly online at the following site: http://www.ibm.com/software/integration/wspartnergateway/library/

Note: Important information about this product may be available in Technical Support Technotes and Flashes issued after this document was published. These can be found on the WebSphere Business Integration Support Web site, http://www.ibm.com/software/integration/wspartnergateway/support. Select the component area of interest and browse the Technotes and Flashes sections.

New in this release

New in release 6.1

WebSphere Partner Gateway 6.1 supports the following new features:
- New business protocols: AS3, SOAP with attachments, CIDX, and ebXML Message Service (ebMS) 2.0 support
- Improved support for Custom XML documents includes better organization, full XPath expression support, search fields, user defined attributes, and synchronous support
- New IPv6 support and enhanced FTP Scripting for supporting AS3
- Reorganization of Document Definition attributes
- New Document Definition attributes for use with User Exits.
- Non-repudiation configurable by document type and trading partner level
- Document viewer has additional user-defined search fields.
- Improved AS Viewer support based on MDN return status
- EDI Configuration Wizard and EIF Import Wizard (previously delivered in the GA02 Support pack)
- New Alert notification mode to send notifications to all related parties (source and target participants) or all subscribed contacts, which reduces alert configuration
- Resend and Gateway permissions now available to users other than the hubadmin administrator
- New user group for allowing multiple users to have the ability to be hub administrators
- LDAP support for log-on authentication
- Use of WebSphere Application Server logging and tracing for WebSphere Partner Gateway components
- Property file configuration data now centrally located and managed by the WebSphere Partner Gateway Console
- WebSphere MQ is no longer a prerequisite product; the WebSphere Platform Messaging support is now used for internal communications
- Selective archive based on partner and/or document type
- Migration of WebSphere Partner Gateway configuration by exporting and importing document definitions and configuration information from one WebSphere Partner Gateway instance to another instance

- A simplified single machine (simple mode) installation option
- WebSphere Application Server Network Deployment now used for multiple machine deployments enabling clustering and central infrastructure management
- Support for using WebSphere Process Server, Version 6.1 as a back-end integration system

Notes:
1. The XML-based administrative API is deprecated in Version 6.1.
2. WebSphere Partner Gateway, Version 6.1 does not support the RC5 algorithm.

New in release 6.0

WebSphere Partner Gateway (which was known as WebSphere Business Integration Connect in previous releases) has the following new features:
- The ability to de-envelope EDI transactions and to validate and transform EDI transactions within those envelopes
- The ability to envelope individual EDI transactions before they are delivered
- The ability to receive multiple record-oriented-data (ROD) and XML documents or EDI interchanges in a single file and to split them into individual documents or interchanges
- The ability to translate among any combination of ROD, XML, and EDI documents
- The introduction of a new transport--FTP Scripting, which can be used for both targets and gateways to communicate with value added networks (VANs) as well as other FTP servers
- The ability to support more than one certificate for certain functions, so that if the primary certificate expires, the secondary certificate can be used
- The ability to send documents from an HTTP or HTTPS gateway through a proxy server to partners

Note that WebSphere Partner Gateway Version 6.0 does not support the RC5 algorithm.

Chapter 1. Getting Started

This section describes how to install and launch Data Interchange Services client.

About Data Interchange Services client

Data Interchange Services client is an interface that allows you to use a PC to create and maintain XML Schema document definitions, XML DTD document definitions, EDI Standards, ROD document definitions, and maps.

Installing the product

Data Interchange Services client has an installation wizard that guides you through installation. As with any installation, you should begin by closing any applications that you currently have running.

Before installing Data Interchange Services client, make sure your PC meets the following hardware and software requirements:
- Intel $^{(TM)}$ Pentium$^{(R)}$ PC
- CD-ROM drive for installing the distributed material
- 256 MB RAM
- Make sure you have enough space on the hard drive on which you are installing Data Interchange Services client for the application, EDI standards, and data files. Minimum recommended space is 40 MB.
- Microsoft Windows 2000
- Microsoft Windows XP
- Data Interchange Services client is designed to support any database with Open Data Base Connectivity (ODBC) Version 2 drivers. The following are examples of database software products and database connectivity products that support ODBC.
  - Microsoft Access 2000
  - DB2 Server Version 8.1
  - Oracle 9
  - Oracle 10

To install Data Interchange Services client, perform the following:
1. Insert the Data Interchange Services client installation CD ROM. The installation process starts automatically.
2. In the Welcome screen, click **Next**.
3. In the License Agreement screen, read the license agreement, select ″**I accept the terms of the license agreement**″ then click **Next**.
4. In the Directory Name screen, either accept the default directory or click **Browse** to navigate to a different directory, then click **Next**. The default directory is C:\Program Files\IBM\DIS Client\V6.0.
5. At the Confirmation screen, confirm the installation location, then click **Next**. The product begins to install.
6. After the installation is complete, click **Finish**.

Launching the product

You can launch Data Interchange Services client by double-clicking the desktop icon created during installation, or by using the Windows Start menu.

The following step describes how to launch Data Interchange Services client using the Windows Start menu.

Note: The following instructions assume you accepted the default installation location, which is: C:\Program Files\IBM\DIS Client\V6.0.

Click **Start** → **Programs** → **IBM Data Interchange Services 6.0** → **DIS Client V6.0**. The Data Interchange Services client main application window opens.

Chapter 2. Learning about mapping

A Data Interchange Services client data transformation map ("DT map") relates a source document to a target document. In Data Interchange Services client, there are three types of documents: EDI, XML, and ROD (flat file). An EDI document can be a transaction such as a purchase order or an invoice in a format defined by a standards body such as X12 or EDIFACT. An XML document can be any type of information encoded using Extensible Markup Language tags according to an XML schema or DTD. A Record Oriented Data (ROD) flat file document can be one of many types of formats used by data processing applications.

Mapping process overview
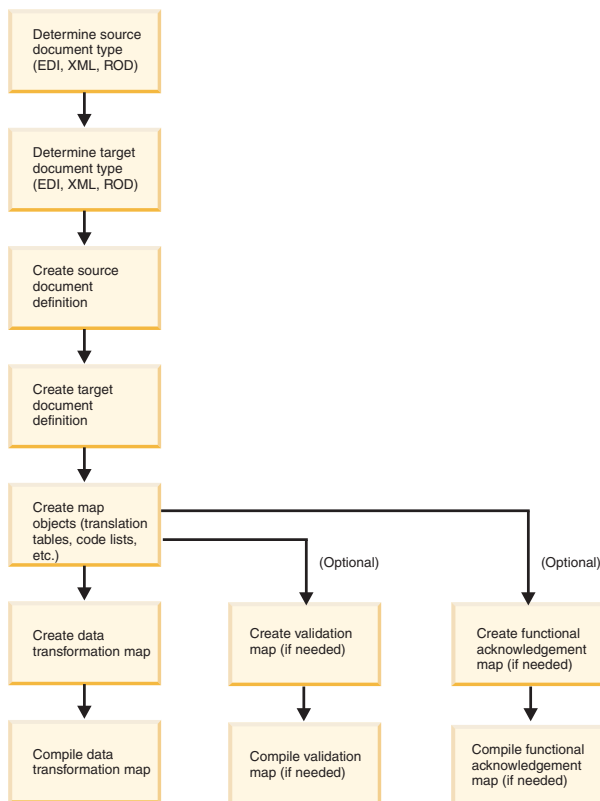
How mapping works

Data Interchange Services client allows you to create or import document definitions for the source and target documents, and then create a map which relates the elements in the source document to elements in the target document.

Data Interchange Services client allows any combination of source and target document types in a map. For example, you can create a map which transforms an EDI document into an XML document, or an ROD document into an EDI document. Similarly, EDI-to-EDI and XML-to-XML maps are possible, along with the other combinations.

A map contains the commands necessary to transform a source document into a target document. Data Interchange Services client provides a graphic user interface (GUI) which simplifies creation of the mapping commands. In the simplest cases, Data Interchange Services client allows "drag and drop" mapping between source and target elements. This type of mapping handles cases where there is a one-to-one correspondence between source and target elements. You can also drag and drop elements when creating more complex mapping commands. After completing the map, you compile it into a "control string" using Data Interchange Services client, and export the control string to WebSphere Partner Gateway.

In addition to the data transformation maps described above, there are two types of specialized maps - validation maps and functional acknowledgment maps.

- **Validation maps** - A validation map provides the instructions needed to perform additional validation beyond what is specified in the EDI Standard. In a validation map, there is no target document. A special command (FAError()) is provided to allow any errors to be reported in an EDI functional acknowledgment. You may create your own validation maps. Special "service segment" validation maps are provided with the product. The service segment validation maps validate the "envelope" segments such as the X12 ISA or the EDIFACT UNB. (These are called "control" segments in X12 and "service" segments in EDIFACT.) WebSphere Partner Gateway allows both service segment validation and a user-specified validation map to be applied to a given EDI document.

- **Functional acknowledgment maps** - A functional acknowledgment map is a special data transformation map used in creating EDI acknowledgments. EDI acknowledgments include the X12 TA1 interchange acknowledgment and the X12 997 and EDIFACT CONTRL functional acknowledgments. WebSphere Partner Gateway creates all EDI acknowledgments using a single ROD document definition. A functional acknowledgment map is used to convert this ROD document into an EDI document such as an X12 997 transaction set or an EDIFACT CONTRL message. Several standard functional acknowledgment maps are provided with the product.

Map types

Data Interchange Services client supports three types of maps that you can use to transform data that you exchange with other business and trading partners: data transformation maps, validation maps, and functional acknowledgment maps.

The following table describes each type of map supported by Data Interchange Services client.

Map types supported by Data Interchange Services client

Map types	Description
Data transformation map	Data transformation maps are the primary type of map used by the translator component of WebSphere Partner Gateway to convert a document from one format to another. They provide the information needed to gather data from a source document and create a target document using the source data. These maps provide a powerful method of mapping data from one document format to another.

Map types	Description
Validation map	Validation maps provide the instructions needed to perform additional validation beyond what is specified in the EDI Standard. The name of the validation map is specified in WebSphere Partner Gateway when it is used to perform additional validation on the source or target document in a translation. Validation maps contain mapping commands that are instructions used to provide additional validation of an EDI document.
Functional acknowledgment map	Functional acknowledgment maps provide the instructions to the translator component of WebSphere Partner Gateway on how to produce a functional acknowledgment. A functional acknowledgment map is a special data transformation map used in creating EDI acknowledgments.

All three map types provide the ability to perform the following tasks:

- Directly relate two elements (data transformation maps only)
- Generate custom functional acknowledgments (functional acknowledgment maps only)
- Generate extended validation (validation maps only)
- Manipulate data
- Set fixed values into elements
- Validate data
- Change values of data based on equivalent value specified in a translation table
- Specify conditional translation processing
- Save data or set flags for use in other translations
- Handle repeating compound or simple elements in similar or different ways
- Handle hierarchical loops (HLs)
- Provide comments anywhere in the map
- Group commands and comments together
- Copy or move mapping commands and comments
- Switch to a different map or execute another map after the current one is finished

Data transformation maps

Data transformation maps are the primary type of map used by the translator component of WebSphere Partner Gateway to convert a document from one format to another.

Data transformation maps consist of commands specifying how to transform a source document into a target document. In each data transformation map, a source document definition and a target document definition is specified. The source document definition describes the layout of the source document (the document to be translated). The target document definition describes the layout of the target document (the output document). Source and target document definitions are specified when the map is first created. You can change the source

and target document definitions after a map is created, but you can not change the source syntax type or target syntax type. For instance, if you are using an EDI document definition as your source document definition, you can change which version of the EDI Standard you are using, but you cannot indicate that you want the source document syntax type to be ROD instead of EDI.

You indicate whether a data transformation map is source or target-based when you create it. This cannot be changed after the map is initially created.

- **Source-based**: A data transformation map can be source-based. In a source-based data transformation map, the map is constructed based on the order elements are defined in the source document definition. This provides an efficient and predictable method of processing the data and the mapping commands. In a source-based map, the source document definition is the base document definition. If the source document contains hierarchical loops and you need to use the special hierarchical loop mapping command (HLLevel()) in a data transformation map to construct the target document, the map must be source-based.

- **Target-based**: There are cases where it is easier to produce the target document if the mapping commands are executed based on the order elements are defined in the target document definition. Data transformation maps support target-based processing. If the target document contains hierarchical loops and you need to use the special hierarchical loop mapping command (HLLevel()) in a data transformation map to construct the target document, the map must be target-based. Generally, source-based maps should be used whenever convenient because translation based on the source document is usually more efficient than translation based on the target document. In a target-based map, the target document definition is called the base document definition.

Validation maps

Validation maps provide the instructions needed by the translator component of WebSphere Partner Gateway to perform additional validation beyond what is specified in the EDI Standard.

A validation map is similar to a source-based data transformation map. Unlike a data transformation map, however, a validation map has no target. Certain specific commands such as MapTo() are not available in a validation map. The command FAError() is available in a validation map, allowing errors to be reported in an EDI acknowledgment.

Each validation map has an associated source document definition. The source document definition describes the layout of the document to be validated. It is specified when the map is first created. The source document definition is always an EDI document definition. You can change the source document definition after a map is created, but you can not change the source syntax type. For instance, you can change which version of the EDI Standard you are using, but you can not indicate that you want the source document type to be a Record Oriented Data instead of an EDI Standard.

The following validation maps are provided by Data Interchange Services, but they are selected by an attribute in the WebSphere Partner Gateway configuration. They are used to perform service segment validation during data transformation processing.

Validation maps

Map name	Description
&WDI_E99AENV_VAL	UN/EDIFACT based on E99A service segments and code lists
&WDI_UCSENV_VAL	UCS based on UCS 4050 service segments and code lists
&WDI_X44ENV_VAL	X12 based on X12 4040 service segments and code lists

WebSphere Partner Gateway may be configured to validate received EDI documents. This is called "source validation." When source validation is performed, WebSphere Partner Gateway may be configured to generate EDI acknowledgments, which may include functional or interchange acknowledgments. Basic EDI validation includes checking things like the minimum and maximum lengths of each element, checking for missing mandatory segments and elements, and so on. Validation maps can provide additional types of tests if needed. Service segment validation can be selected, and a user-specified validation map can also be selected.

WebSphere Partner Gateway may also be configured to validate EDI documents that were generated by WebSphere Partner Gateway, to ensure the documents are correct before they are sent to trading partners. This is called "target validation." The goal of target validation is simply to pass or fail the document in question. No acknowledgments are generated for target validation. A user-specified validation map can be specified for target validation, but service segment validation does not apply. (The service segments have not yet been generated when target validation is applied.)

The Error() mapping command is used in validation maps to indicate that an error exists in the document. The Error() command can be used by a validation map when a source EDI document is going to be translated or when validating an EDI document that is produced by a translation. The Error() command will not produce information that can be used in a functional acknowledgment. If an error should be included in a functional acknowledgment, you must use the FAError() mapping command instead. It works like the Error() command except that functional acknowledgment information is specified in the command. The functional acknowledgment information is stored internally and becomes the source document when creating the EDI functional acknowledgment target document. Both Error() and FaError() commands log message UT0033 and write the message to an internal file.

Functional acknowledgment maps

Functional acknowledgment maps provide the instructions to the translator component of WebSphere Partner Gateway on how to produce a functional acknowledgment.

The translator component of WebSphere Partner Gateway can automatically generate EDI Standard functional acknowledgments for EDI documents received from a trading partner. To produce a functional acknowledgment, a functional acknowledgment map must be associated with the source document that is being translated in WebSphere Partner Gateway. The source document must be an EDI document. When a document is going to be translated, it is first validated as specified in WebSphere Partner Gateway. The translator component of WebSphere

Partner Gateway provides a standard level of validation on the EDI document. If a functional acknowledgment is going to be generated, results from validation of an EDI document are written to an internal document. Validation maps are created to provide additional validation on an EDI document. Results of the validation map are also written to the internal file. The generation of a functional acknowledgment uses the functional acknowledgment map specified in WebSphere Partner Gateway and this internal file as its source document. The functional acknowledgment map contains mapping commands that indicate how to use the validation results contained in this internal file to create a specific functional acknowledgment. If a document is accepted for translation by the validation process, then the appropriate data transformation map is used to translate the source document. All of this is controlled by the association of the source document and the data transformation map in WebSphere Partner Gateway.

The source document definition for all functional acknowledgment maps is a Record Oriented Data (ROD) document definition with the name &FUNC_ACK_META contained in the dictionary &FUNC_ACK_METADATA_DICTIONARY. It describes the layout of the internal file generated by the validation process. The name of the source document definition cannot be changed. You cannot create a functional acknowledgment map without this ROD document definition in your database.

The target document definition in a functional acknowledgment map describes the layout of the functional acknowledgment. It must be an EDI document definition with a name of 997, 999 or CONTRL.

Normally, you should not need to create or modify a functional acknowledgment map. Data Interchange Services client provides several maps to produce the most common functional acknowledgments. Create a functional acknowledgment map only when a custom functional acknowledgment is required. The functional acknowledgment maps provided by Data Interchange Services are listed in the following table.

Functional acknowledgment maps

Map name	Description
&DT_FA997V2R4	Functional Acknowledgment 997 – X12 Version 2 Release 4
&DT_FA997V3R5	Functional Acknowledgment 997 – X12 Version 3 Release 5
&DT_FA997V3R7	Functional Acknowledgment 997 – X12 Version 3 Release 7
&DT_FA999V3R3	Functional Acknowledgment 999 – UCS Version 3 Release 3
&DT_FACONTRL	Functional Acknowledgment CONTRL – UN/EDIFACT prior to D94B
&DT_FACONTRL94B	Functional Acknowledgment CONTRL – UN/EDIFACT Version 2 Release 1 (D94B)
&WDI_TA1_ACK	This map is used to generate a TA1

The &DT_FA997V3R7 functional acknowledgment map is used to generate functional acknowledgments for X12 version 3, release 7 and higher. The &DT_FACONTRL94B functional acknowledgment map is used to generate

functional acknowledgments for UN/EDIFACT version 2, release 1 (D94B) and higher. All of the functional acknowledgment maps listed above should not be modified or deleted. If you need to make a customization, it is recommended that the appropriate map is copied and the needed change is made in the copied map.

Important: The &WDI_TA1_ACK map name can not change. It is recommended that a MapSwitch() mapping command be added to execute a custom TA1 map.

Object types

Data Interchange Services client provides the following types of objects that you can use when creating or editing maps: User Exit profiles, code lists, transformation tables, and global variables.

The following table describes each available object type.

Object types

Object type	Description
User Exit	A User Exit profile is used to identify a user exit that can be called during a translation.
Code list	A code list defines a list of acceptable values that are used in maps to validate any source simple element or variable.
Translation table	A translation table is used to change one value into another corresponding value during translation of a document.
Global Variable	Global variables are used with data transformation maps, validation maps, and functional acknowledgment maps. During translation, you can put data into a global variable. The data contained in a global variable is available to other translations.

User Exit profiles

A User Exit profile defines a field exit routine to Data Interchange Services client. It provides Data Interchange Services client with the information it needs to call a field exit routine. A field exit routine can be called from data transformation maps, validation maps, and functional acknowledgment maps. Field exit routines are invoked during the translation of a document when the Exit() function is encountered in a mapping command.

You need a User Exit profile defined for each user provided field exit routine you use. The name of a User Exit profile is used as a logical name for the field exit routine throughout Data Interchange Services client. A User Exit profile identifies the name of the class and method used to invoke the exit routine.

You call a field exit routine to perform special processing of almost any nature. Field exits are expected to return a character string if they complete successfully, though it is not required. The returned character string can be used in a mapping expression or can be used to update a variable or a simple element in the target document. The field exit can be used for almost anything. Field exits can accept

several parameters. Parameters can be variables, constants, the result of an expression, or a path that identifies a simple element in the source document definition.

Field exit routines

Field exit routines are used within data transformation maps, validation maps, and functional acknowledgment maps in the Exit() function. They are defined by creating a User Exit profile. The class and method within the class to be executed are specified in the User Exit profile. A field exit routine is written in Java.

Code lists

A code list is a list of acceptable values that is used in maps to validate any source simple element or variable. Code lists are provided by EDI Standards. They indicate which values are valid for a specific EDI data element. A code list can also be created to specify any list of valid values. When the translator component of WebSphere Partner Gateway validates or translates a document, and the use of a code list is specified to validate a piece of data, the specified code list is searched to see if it contains the value.

Translation tables

A translation table is used to translate a value into a corresponding value. Translation tables are used in data transformation maps, validation maps, and functional acknowledgment maps. Translation tables contain a list of unique values called source values. Each unique source value has a corresponding target value assigned to it. The target values do not have to be unique. You should always define all possible values in the translation table.

Use a translation table in a map to indicate that an alternate value is to be used instead of the value contained in a simple element or variable. An example of where a translation table might be useful is when a simple element contains an internal part number that must be converted to a trading partner's corresponding part number. When the instruction to use a translation table is encountered during translation, the value in the source simple element is looked up in the specified translation table. If it is found in the table, the corresponding value is then substituted for the original source value. If it is not found, there is an option that allows you to specify a default value.

In data transformation maps, validation maps, and functional acknowledgment maps, the lookup of a value is normally done against the source values in a translation table. When a matching source value is found, the corresponding target value is substituted for the original value. There is an option that allows you to perform a lookup using the target value. This allows you to use the same table to translate a value to a trading partner's equivalent value and also to translate a trading partner's value to your equivalent value. Using the target value for a lookup is useful only when target values are unique. When the option to perform a lookup using the target value is used, the lookup attempts to locate a target value in the translation table that matches the value in the simple element or variable. When found, the corresponding source value is substituted for the original value.

Global variables

A global variable defines a variable that can be used across mapping translations. These variables are used much like variables in most programming languages. They can hold and manipulate data in data transformation maps, validation maps, and functional acknowledgment maps. While a document is being translated, data can be put into a global variable. After the translation of the document ends, the

data remains in the global variable. The data in the variable may be available in the next translation, depending on the scope of the variable, regardless of the map that is used to perform the translation. During subsequent translations within the scope of the variable, the data in the global variable can be obtained, manipulated and changed.

One attribute of a global variable is scope. The scope of a global variable can be *group* or *interchange*. Global variables that have a scope of interchange are created and initialized when an interchange is encountered. They are reset at the start of the next interchange. Variables with a scope of group are created and initialized when a group is encountered. They are reset at the start of the next group. Not all syntax types support the concept of groups or interchanges. When the source document type does not support groups, the scope of group is treated the same as interchange. If the document type does not support interchanges, the variable is reset at the start of each input document. For example, if the input file contains multiple documents and is split into separate documents by the RODSplitter or XMLSplitter, the global variables are reset for each input ROD or XML document in the file.

Other attributes of global variables include data type, maximum length, and initial value. Maximum length dictates how long the data within a global variable can be. The maximum length is only available for certain data types. Global variables can also have an initial value. When a global variable is created or reset automatically (goes out of scope), the initial value will automatically be put into the global variable. The default initial value for a numeric data type is zero. An empty string is the default initial value for character string and binary data types.

Data transformation maps, validation maps, and functional acknowledgment maps use global variables in map commands. Every global variable is available to every map. A global variable must be defined before it can be used in a map command. They can be defined from the Global Variables list window of the Mapping Functional Area, the Data Transformation Map Editor, the Validation Map Editor, or the Functional Acknowledgment Map Editor.

A global variable must reside in the database accessed by the translator component of WebSphere Partner Gateway before a map using the global variable can be used successfully to translate a document. Ensure the global variable exists in the database by defining the global variable in the database using Data Interchange Services client, or use export and import functions to copy the global variable to the database. Global variables can be exported by themselves or with maps or map control strings.

Certain changes to a global variable affect all maps that utilize the global variable. All maps that use a global variable that have had certain changes must be recompiled. The resulting map control string must then be exported to the database accessed by the translator component of WebSphere Partner Gateway. A map must be recompiled when the name, scope, or data type of a global variable has been changed.

Control strings

To make translation of a document more efficient, Data Interchange Services client uses control strings to drive translation. There are several types of control strings used within Data Interchange Services client. The visible control strings are the map control strings. Instead of using a map directly, the translator component of WebSphere Partner Gateway uses a control string representing the map. Maps are

never directly used to translate a document. Map control strings are created by
compiling maps. Map control strings are required for all map types: data
transformation maps, validation maps, and functional acknowledgment maps.

When you compile a map, control strings are generated for the map and often for
the source and target document definitions referenced by the map. The map
control strings associated with compiled maps can be listed on the Control Strings
list window in the Mapping Functional Area. Control strings associated with
document definitions can not be viewed. These are managed with the map control
strings. Document definition control strings are exported with the map control
strings that need them.

Map control strings should be recompiled any time the corresponding map is
changed or any time the related source or target document definition is changed.
Changes will not be available to the translator component of WebSphere Partner
Gateway until the recompile has been successfully completed and the new control
string has been exported to the database accessed by the translator component.

Queries

A query is a request for specific information from the database. It allows you to
determine what information you want to see and the order in which you want to
see it. Queries are used to display most objects on a list window. They allow you
to decide what columns of data will be displayed on the list window, and how it
will be sorted. Queries also have the ability to provide selection criteria. Selection
criteria can be used to limit the records that are displayed in a list window.
Queries can also be used to improve performance by limiting the number of
documents and columns that appear on a list window.

The query function in Data Interchange Services client gives you extensive control
over the information that appears in any list window. Queries control both the
number of documents that are listed and the number of fields that displayed. They
are particularly useful after you have been using Data Interchange Services client
for a while and the number of items on your list window is large.

A query is executed each time any list window is opened in Data Interchange
Services client. You see the results of that query displayed on the list window of
each tab. For example, when you open the Data Transformation Maps list window
in the Mapping Functional Area, the default "All" query is often run. This query
locates all data transformation maps and displays the results on the list window.
The list window opened with this query lists all data transformation maps and
sorts them by name.

You can change which query is executed to display the result on the list window. It
can be changed to any available query that exists on the database. The available
queries can include default queries provided by Data Interchange Services client
and other queries that you or other persons have created. Use the Properties
button on the list window toolbar to change which query is executed on an open
list window.

A query can be created that is "public," "protected," or "private." This function is
only useful in installations where the Data Interchange Services client
Configuration database is shared by several users. A public query is available to all
users of the shared Configuration database. Any user can access the query and can
alter or even delete the query. Protected queries are available to all users of the

shared Configuration database; however, only the creator of the query can alter or delete it. Only the user that created the query can access private queries.

Databases

Data Interchange Services client uses three different databases, two of which can have multiple occurrences. The three databases are: Data Interchange Services client database, Document Manager database, and Configuration database.

A **Data Interchange Services client database** contains build time data and runtime data. This database is used only by Data Interchange Services client. It is used to develop and maintain maps, document definitions, and runtime objects such as control strings, code lists, translation tables, User Exit profiles, and global variables. Runtime data is moved from this database to a corresponding Document Manager database using export and import functions.

Data Interchange Services uses WebSphere Partner Gateway's **Document Manager database** when it is translating documents. The Document Manager database contains only runtime data. Maps and document definitions are not kept in this database. Runtime data is moved to this database from a Data Interchange Services client database using export and import functions. Data Interchange Services client can view and manipulate the data contained in this database. The translator component of WebSphere Partner Gateway typically accesses only the Document Manager database.

The **Configuration database** is used only by Data Interchange Services client. It contains information used to access the other databases. The information includes queries, the message log, preference information, and information about the other databases defined to Data Interchange Services client. It can reside on a local system or a remote system. The Configuration database can be a single-user database or it can be a shared database.

Build time data consists of maps, ROD document definitions, EDI document definitions, XML schema document definitions, and XML DTD document definitions. Build time data is data that is not directly used in the translation of a document.

Runtime data consists of control strings, code lists, User Exit profiles, translation tables, and global variables. Runtime data is data that is used during translation of a document. The translator component of WebSphere Partner Gateway uses only runtime data.

If you have multiple Data Interchange Services client users and require concurrent access, then define the database in a multi-user database management system. IBM's DB2 can be used for this. Any multi-user database supporting ODBC Version 2 or later can be used by Data Interchange Services client. The Data Definition Language (DDL) used to create databases, tables, and indexes is not the same for every DBMS. Sample DDL is distributed in the WebSphere Partner Gateway installation.

Data Interchange Services client can access many Data Interchange Services client databases and Document Manager databases simultaneously. This allows you to work with multiple databases at one time. Many users set up more than one database. Typically a Data Interchange Services client database is paired with a Document Manager database. You may even establish several pairs of these databases. For instance, you may have a Data Interchange Services client database

and Document Manager database pair established for a development environment and additional pairs for test and production environments. All of these can be accessed from the same Data Interchange Services client application.

Each database to be accessed by Data Interchange Services client must be identified to Data Interchange Services client. A database is defined to Data Interchange Services client using a database definition. The databases defined to Data Interchange Services client can be listed by selecting Databases from the View menu. This displays the Databases list window. The list window is used to list and perform maintenance functions on database definitions. Use the Database Editor to add, view, and change information about a database definition. The editor is accessed through the Databases list window.

Data Interchange Services client is installed with a default single user Data Interchange Services client database. The database is located in the install directory. The database is defined to Data Interchange Services client using a database definition called "Development." This database can be used, deleted, or ignored. The ODBC Data Source Name for this default database is "WDIClientWBIC60win." This database can be copied to create additional local databases if desired. It has the name "wdiclientwbic60dev.mdb" in the install directory. Use "Microsoft Access Driver" when creating an ODBC definition for copies of this database using the Windows Data Source Administrator tool.

A database physically located on a user's machine is commonly referred to as a local database. The "Development" database installed with Data Interchange Services client is a local database. Other databases installed by the user on their local machine are also considered to be local databases. There can be more than one local database.

Import and export

Data Interchange Services uses an export process to move information out of Data Interchange Services client. During the export process, the data is extracted from a database, then formatted and put into a file. Data Interchange Services client also has the ability to export information from one Data Interchange Services client database to another database without the intermediate import step. Data Interchange Services client uses an import process to move previously exported data from a file into the Data Interchange Services client database. The import process can also import XML schemas and XML DTDs.

You use the export and import process when you need to:
- Install EDI Standards in a Data Interchange Services client database
- Import XML schemas and XML DTDs to a Data Interchange Services client database
- Move profiles, XML document definitions, EDI Standards, ROD document definitions, and maps from one Data Interchange Services client database to another
- Share profiles, maps, and EDI Standards with other Data Interchange Services client users
- Copy runtime data to a Document Manager database

The file created by the export process and input to the import process is interchangeably called an export file or an import file. Though the two terms are commonly used, they refer to the same file.

Data Interchange Services client provides an additional function that combines the export and import processes. Use the "Export to Other Database" function to export data out of one database and import it directly into another database. Export to another database when you want to duplicate items in various databases. For instance, if you create a translation table in a test database and want to duplicate it in a production database, use the "Export to Other Database" command.

COBOL copybooks can also be imported into Data Interchange Services client. Importing a COBOL copybook can be used to produce a ROD document definition and ROD records used in a ROD document definition.

Chapter 3. Learning about the Graphical User Interface

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

Through the Data Interchange Services client interface you can perform the following tasks:

- Create, update, and manage the following Data Interchange Services client components:
  - Maps
  - XML document definitions
  - EDI Standards
  - Profiles, tables, and other supporting objects
  - Proprietary document definitions
- Import Extensible Markup Language (XML) schemas and Document Type Definitions (DTDs) into Data Interchange Services client
- Set up and maintain database definitions
- Set preferences
- Work with the message log

Main application window

The Data Interchange Services client main application window is used to manage all functions that can be performed in Data Interchange Services client. It contains a navigation bar and a menu.

The navigator bar consists of a series of buttons that each open a functional area. A button is provided to open the help for Data Interchange Services client. A database selector is also provided. You select the database you want to work with before opening any functional area. The functional areas include:

Functional area icon	Functional area name
	Mapping
	XML
	EDI
	Record Oriented Data

Use the menus to access all functions with Data Interchange Services client. The menus that appear, and the contents of each menu, vary depending on whether a list window or editor is active. The following table describes the menu options.

Menu options

Menu option	Description
File	Used to open editors and functional areas, work with queries, import, close windows, and close the application
Actions	Used to perform actions on an object, such as export, copy and delete
Edit	Used only in editors to perform the common cut, copy, and paste functions
Navigate	Used only in editors when an editor is opened with more than a single item
View	Used to indicate whether toolbars and other elements of a window should be displayed, refresh list windows, or access other functions such as database definitions, preferences, or the message log
Window	Used to navigate or arrange the open windows
Help	Used to access help and obtain information about the application

Most work done within the application is done in list windows and editors. A list window displays members of a single component. List windows can be displayed by themselves or in a functional area window. A functional area window contains several list windows, each listing members from related components. The members selected for display in a list window are governed by the current query for that component. Most components in a functional area use a default query that results in all members of the component being selected and displayed in the list window. The default query displays only the most commonly referenced columns in the list window to improve performance.

List windows can also be opened individually without the involvement of a functional area. This occurs by selecting the Open Query List function on the File menu or by pressing buttons within various dialog or editors within the application. The Open Query List function displays the Query list window. Within this list window, you can select the component you want to work with along with the query to use to select members of the component to be displayed.

You may select alternate queries once the list window is displayed. One or two other queries are provided for each component type by default. One of these allows you to specify selection criteria while the other, if provided, lists most columns for each item within a component. You may create and use your own queries. Use of selection criteria is a useful way to create queries that will list only a subset of the members of a component. It is recommended that you include only the columns that are really needed in a query. This will improve performance, especially when listing a large number of members, or the database is not local, or if network performance is not optimal when accessing the database.

Use list windows by themselves or within a functional area to maintain the individual items of an object type. Items can be added, edited, renamed, deleted, and so on, using the list window.

You can have several functional area windows opened simultaneously. Also, the same functional area list window can be opened more than one time. This allows you to open several list windows on the same component.

Data Interchange Services client can view multiple databases at the same time. This allows you to work with development, test, and production database simultaneously. When opening a functional area, it will open using the database selected on the navigator bar.

Editors are used to view and update items in components. In most cases, they are also used to create the items in a component. Another way of creating items in a component is to import them into a database. There are a few item types that use a wizard dialog to collect the needed information and then create the item.

Mapping functional area

The Mapping functional area is used to view and maintain maps and related items. You access the Mapping functional area by clicking the Mapping button on the Data Interchange Services client navigator bar. This displays a functional area window that contains several list windows.

The following table describes the mapping components represented in each list window of the Mapping functional area.

Mapping component	Description
Data transformation maps	Data transformation maps are used to transform a document in any format into a document in any format.
Validation maps	Validation maps provide additional validation for EDI documents.
Functional acknowledgment maps	Functional acknowledgment maps are used to create EDI functional acknowledgments. Data Interchange Services client provides a basic set of functional acknowledgment maps to produce common functional acknowledgments. You need to create your own or modify these only if you have special requirements.
Global variables	Global variables are used with data transformation maps, validation maps, and functional acknowledgment maps. During translation, you can put data into a global variable. The data contained in a global variable is available to other translations.
Control strings	The translator component of WebSphere Partner Gateway does not use maps directly to translate or validate documents. Instead, map controls strings are used. Maps are compiled into map control strings for use in translation and validation.
Translation tables	Translation tables are used to change one value into another corresponding value during translation of a document.

Mapping component	Description
Code lists	Code lists define a list of acceptable values that are used in maps to validate any source simple element or variable.
User Exit profiles	A User Exit profile is used to identify a user exit that can be called during a translation.

XML functional area

The XML functional area provides access to XML dictionaries, schema document definitions, DTD document definitions, and Namespace objects. You can access the XML functional area by clicking the XML button on the Data Interchange Services client navigator bar. This displays a functional area window that contains one list window for each XML related object type.

The XML functional area contains one list window for each of the following components:

- XML dictionaries
- Schema document definitions
- DTD document definitions
- Namespace objects

XML provides a readable, easy to use data format that trading partners use to exchange data between their computer applications. In essence, XML provides the building blocks for electronic versions of common business documents.

The translator component of WebSphere Partner Gateway can translate data from one document format into an XML document for transmission to a trading partner. Conversely, the translator component can translate data in an XML document into another format.

In order to begin exchanging XML documents with a trading partner, you must select an XML schema or DTD that defines the layout of the document you want to send or receive. The schema or DTD must be imported into Data Interchange Services client.

EDI functional area

The EDI functional area is used to view and maintain EDI Standards. You can access the EDI functional area by clicking the EDI button on the Data Interchange Services client navigator bar. This displays a functional area window that contains several list windows. Each of these list windows displays a component that is part of the EDI functional area.

The EDI functional area contains one list window for each of the following components:

- EDI dictionary
- EDI document definitions
- EDI segments
- EDI data elements
- Code lists

EDI Standards provide a common data format that trading partners use to exchange data between their computer applications. In essence, EDI Standards provide the building blocks for electronic versions of common business documents.

The translator component of WebSphere Partner Gateway can translate data from one document format into an EDI document. Conversely, the translator component can translate data in an EDI document into another format. The latest set of EDI Standards currently approved by the primary EDI Standards organizations can be downloaded from the following product support page: http://www.ibm.com/ software/integration/wspartnergateway/support/.

In order to begin exchanging EDI documents with a trading partner, you must select an EDI document definition that corresponds to the information you want to send or receive. Ideally, you and your trading partner can agree on an EDI document definition that requires no customization to meet your needs, but this is not always possible. If you and your trading partner need to exchange specific information that is not included in existing EDI Standards, Data Interchange Services client allows you to customize currently approved EDI Standards to fit your needs.

CAUTION: When altering EDI Standards, you should work in close partnership with your trading partners. If you customize EDI Standards without informing your trading partners of the changes, they may not be able to process the EDI documents you send. If you need to alter a component of an EDI Standard, it is recommended that you copy the EDI Standard and then alter the copy

Record Oriented Data (ROD) functional area

The Record Oriented Data functional area is used to view and maintain ROD document definitions, which describe the layout of proprietary documents. There are two types of ROD formats: raw data records and Control and Data (C and D) records.

Raw data

Each record in raw data format identifies itself by containing a unique record identifier (a record ID). Raw data can be either delimited or fixed-position. For fixed-position data, the identifier starts in the same position and extends for the same length in each record. In delimited data, each value is separated by a delimiter such as a comma for comma-separated data. The Record ID is actually a field in the record. As long as records contain identifiable record IDs, you can use application data without modification. In the illustration below, which is an example of fixed-position data, several records of application data contain the record ID in the first three positions. In this example, HDR is the record ID of the header record, NAM is the name, and so on.

Application data with record identifier

HDR0123456 092091C321

NAMSmithson, Patricia Jeanne

SSN5555555555

PRVCity Regional Clinic

PID05050505-X505

ADR555 Cedar Road

ADRAny City IL

ADR61001-1101

TOT1555.00

When your data has no record identifier for Data Interchange Service client to associate with each record, you have two choices:

- Modify your application data to contain a record identifier.
- Modify your application to use control (C) and data (D) records.

Control and Data (C and D) Format

If no record ID clearly specifies the type of information contained in a record, that record structure can be indicated by using Control and Data (C and D) records. Also use C and D records when you need to use multiple data formats in a single file or you need to use overrides offered in the C record that are not offered in raw data. You also can use this type of format to override fields within service segments (such as ISA, GS, UNB, and UNH). There are a number of ways to structure C and D records. This example, shows an example of a C and D record in which:

- A C or D is in the first column (byte)
- Record name is in the next 16 columns
- Application data starts in column 18

This example also shows the use of a Z record to indicate the end of the document.

Sample Application Records with C and D Records

1 2 3 4 5 6

1 2345678901234567890123456789012345678901234567890

CSPSTT16 AFTSU09 IL

DPOHDR P0123456 092091C321

DP0NOTE INCOMPLETE INVOICE INFORMATION SLOWS REIMBURSEMENT

DP0NOTE PATIENTS WITH MULTIPLE CLAIMS NEED COMPLETE HISTORY

DINVITEM 005500550055

DITEMDESC SURGICAL PROCEDURE

DITEMDIAGN CARPAL TUNNEL PAIN

DINVITEM 005500550055

DITEMDESC SURGICAL PROCEDURE

DITEMDIAGN LIGAMENT INFLAMMATION

DNAME SMITHSON, PATRICIA JEANNIE

CSPSTT16 AFTSU09 IL

Use the General page online help for "Record ID Information Object" for complete information on RAW data and C&D Formats.

You can access the Record Oriented Data functional area by clicking the Record Oriented Data button on the Data Interchange Services client navigator bar. This displays a functional area window that contains several list windows. Each of these list windows displays a component that is a part of the Record Oriented Data functional area.

Database field

The Database field in the navigation bar of Data Interchange Services client allows you to choose which database to work with. By default, the Database field is set to Development, which is the name of the database installed with Data Interchange Services client. If you want to work with a different database, you must select the database you want to work with before opening any functional area.

Help

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

The Help system contains information about each Data Interchange Services client window and field and explains how to use the operations presented to you. You also have access to the Data Interchange Services client glossary of terms and Data Interchange Services client reference material. It also contains a complete list of mapping commands and functions.

Viewing the message log

Data Interchange Services client contains a message log. Significant error messages and other information is written to the message log as needed.

The following steps describe how to view the message log.
1. Open the Message Log list window by selecting Message Log from the View menu.
2. To view the complete message, select the message, then select **Open** or **View** from the File menu.

By default, the message log behaves in the following way:
- Messages are deleted from the message log when they expire. Whenever you start Data Interchange Services client, it automatically removes expired messages from the message log.
- When the message log is disabled, no messages will be written to it.

The following table describes the contents of each field in a message log.

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

Using lock and unlock

The Lock function is provided as a mechanism to protect an item from being worked on until it is manually unlocked using the Unlock function. If proper processes are implemented within your installation, the Lock and Unlock functions can act as a control mechanism to protect your data.

The Lock function is used to lock one or more items selected in a list window. Items become locked automatically when they are being worked on in Data Interchange Services client. This function provides a mechanism to ensure that two people are not trying to alter or work with the same item simultaneously. Items become locked when they are opened, renamed, copied, or compiled. If Data Interchange Services client terminates abnormally, an item may be left in a "locked" state.

Viewing the item in a list window can show you who currently has a lock on an item. You can check with them to determine why an item is locked. It may be inadvertent and might need to be manually unlocked using the Unlock function.

The following steps describe how to lock one or more items.

1. Open a list window that displays the item or items you want to lock.
2. Then select each item you want to lock in the list window, then select **Lock** from the Actions menu.

   The lock function begins and the Execution Status window appears. A message is added to the window for each item that is locked. When Data Interchange Services client is finished locking the selected items, a message is displayed on the Execution Status window indicating the lock function has been completed. Click **Cancel** on the Execution Status window to terminate the lock function in process.

   Note: Not all components can be locked or unlocked. For these items, the Lock and Unlock functions are disabled

Using multiple databases

Data Interchange Services client allows you to do work in as many databases as you like at the same time. This can be very convenient for comparing information in the development, test, and production databases. Each database to be accessed by Data Interchange Services client must be identified to Data Interchange Services client using a database definition.

The following steps describe how to work in multiple databases.

1. Open a functional area window by clicking on its respective button on the navigator bar or by selecting Open Functional Area from the File menu. This opens the functional area list window for the database currently selected on the navigator bar. The name of the current database appears in the Database field on the navigator bar.

2. You can select a different database by selecting any defined database from the Database drop-down list.

A background frame color is associated with each database. The background frame color is used whenever a list window or editor is opened for a database. This makes it a little easier to know which database you are dealing with when looking at a list window or editor by using a different color for each defined database. For instance, if you have a test database defined with the color blue, all list windows and editors for that database will have the background frame color set to blue. If you have a production database defined with the color red, all list windows and editors for that database will have the background frame color set to red. If you have a map editor opened for both the test and production databases, it is easy to tell which database each editor is associated with because the background frame color is blue for the test database or red for the production database.

Using queries

Queries are maintained using the Query list window. The Query list window is used to list available queries for most object types. From the window, queries can be created, changed, viewed, copied, and deleted. They can also be executed there. Executing a query, sometimes called "running" a Query, causes a list window to be opened. The list window executes the selected query and displays the results. The Query list window is displayed by selecting the Open Query List function on the File menu.

Creating queries

The Query list window is used to list, maintain, and create queries.

The following steps describe how to create a query.

1. Open the Query list window by selecting **Open Query List** from the File menu.
2. Click **New**. This opens the Query editor.
3. In the Query editor, do the following:
   a. Type a name for the query in the Query Name field.
   b. Type a description in the Description field.
   c. Select **Public**, **Protected**, or **Private** to indicate how you want to share the query.
   d. Click **Next**. This opens the Selected Column Information window.
4. In the Selected Column Information window, select the columns that you want in the query, then click **Next**. The Sort Information window appears.

5. In the Sort Information window, indicate how you want the information sorted, then click **Next**. The Selection Criteria window appears.

6. In the Selection Criteria window, specify the filter conditions you want to use to limit the results of the query, then click **Finish**.

Now, you can run the query.

Running queries

Queries can be executed from the Query list window.

The following steps describe how to run a query.

1. Open the Query list window by selecting **Open Query List** from the File menu.

2. In the Query list window, make the following selections:

   a. From the Functional Area drop-down list, select the functional area for the query you want to run. The object types associated with the selected functional area appear in the Object Type list. The functional area choices are: Mapping, XML, EDI, Record Oriented Data, or Message Log.

   b. Select the object type from the Object Type list. The queries defined for the selected object type appear in the Queries field.

   c. In the Queries field, select the query you want to run.

3. Click **Run**.

Chapter 4. Creating maps

You can create three types of maps using Data Interchange Services client: data transformation maps, validation maps, and functional acknowledgment maps.

Mapping task list

The following task list outlines the steps involved in creating a map, including the steps you must take before and after creating a map. You can use this task list as a checklist each time you create a new map.

1. From your trading partner agreement, identify the business documents that best fit your business needs using the industry-supplied, national, or international formats. You and your trading partners must agree on what documents to use, and the content and format of the documents to use.

2. Obtain the business document implementation guide. The implementation guide contains specific information about the content and format of a business document that will be exchanged between you and your trading partner.

3. Ensure that you understand your source and target data so that you can take the source data and put it into a target business document format.

   - **EDI**: For EDI data, refer to the documentation for the standard you are using. You may obtain your standard documentation from various sources, such as industry groups, standards bodies, vendors, or trading partners. Most transaction sets or messages have three general areas: The **heading** area contains information about the entire document, such as company name and address. The **detail** area contains the message details such as quantities and prices. The **summary** area contains more information about the entire message, such as total amount due.

   - **XML**: Extensible Markup Language (XML) is a language that is becoming a popular way to represent structured documents and data. XML defines the syntax used to represent the data, including information such as how to tell an element name from an element value. However, it does not define the semantics or structure of the data. For example, it does not specify where a purchase order number should appear, or even whether it is a part of any particular document. The structure of an XML document is defined by a Document Type Definition (DTD) or schema. The syntax of the DTD is defined as part of the XML language or schema. The DTD or schema provides a list of all components included in the XML document and their relationship to each other. For example, a DTD or schema may state that the header element of the document contains a PONum element. The meaning of the PONum element may be described either in the comments within the DTD or schema, a separate document, or both. Unlike EDI standards where there are a small number of dominant EDI standard formats that define the document structure, there are numerous different XML DTDs and schemas. Also, since XML is extensible, users are free to create their own DTDs and schemas if they choose, instead of restricting users to a fixed subset of DTDs and schemas. You may obtain your DTDs and schemas from various sources, such as industry groups, standards bodies, vendors, or trading partners, or you may create them yourself.

   - **Record Oriented Data (ROD)**: A record layout clearly identifying each field, its content, structure, and position, can help you determine the

different record types and the relationships for records and data fields. You must also determine the relationship of your application data to the target business document.

4. Create a mapping implementation guide. Determine the relationship of your data with the standard format. You should examine each source document element in your data and make a decision on a corresponding target document element. This enables you to identify mapping commands and additional mapping objects needed during the mapping process. A mapping implementation guide can be very useful during this step.

5. Create source and target document definitions.

6. Create additional mapping objects to use during the mapping process, such as translation tables or code lists. In addition, identify if a validation map and functional acknowledgment map are needed.

7. Create a data transformation map. At this point, use the various available mapping techniques to create a map. This might include a "drag and drop" from a source field to target field, or the specification of an "assignment command" to set a target field to a specific value. Other mapping commands and mapping functions can be use to format data, convert data from one value to another, or qualify the movement of data based on the content of the source message. "Conditional statements" can be used to provide alternate courses of action for the translation engine.

8. Compile the data transformation map control string.

9. Export the data transformation map control string.

10. Create a validation map (if needed).

11. Compile the validation map control string.

12. Export the validation map control string.

13. Create a functional acknowledgment map (if needed).

14. Compile the functional acknowledgment map control string.

15. Export the functional acknowledgment map control string.

After the map control strings have been exported, the following tasks can be performed from WebSphere Partner Gateway.

1. Import all mapping and mapping objects.

2. Set up partner rules for mapping execution.

The final step is to test the mapping (in WebSphere Partner Gateway) and make any necessary adjustments (in Data Interchange Services client).

Creating database definitions

You create a new database definition whenever you have a database that Data Interchange Services client needs to access. Data Interchange Services client distributes Microsoft Access databases as the local database.

The following steps describe how to create a new database definition.

1. Open the Databases list window by selecting Databases from the View menu.

2. Click **New** on the toolbar.

3. Type the name of the database definition. The name uniquely identifies the name of the database definition. This is a required field.

4. Type a description of the database.

5. Select the ODBC Data Source Name (DSN) from the list of defined data source names. This is a required field. The data source name must be defined to Windows before you can select it in a database definition.

6. Specify the database qualifier if it is required by the database system to identify the database. The database qualifier is used when connecting to multi-user, or server, databases. For instance, on DB2 for Windows this corresponds to the database schema.

7. Select the Database Type. The database type indicates whether the database is a Data Interchange Services Database or is a Document Manager database.

8. Click **Change** if you wish to change the color of the window background for this database. Clicking the Change button opens the Windows Color dialog. Select a color to use as the window background for this database and click **OK**. Click **Cancel** to close the dialog without changing the color for the database.

9. Click **OK**. Data Interchange Services client saves the new database definition and closes the Database editor.

Note: Data Interchange Services client uses ODBC to access databases. This means that an ODBC Data Source Name (DSN) is required to access each database. The Data Source Name can be any name of your choice. Use the Data Source Administrator tool found within the Control Panel of most Windows 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.

Creating dictionaries

Dictionaries must be created before document definitions can be created. Data Interchange Services client allows you to create three types of dictionaries: XML dictionaries for DTD or schema document definitions, EDI dictionaries for EDI document definitions, and ROD dictionaries for ROD document definitions.

Creating XML dictionaries

DTD or schema document definitions are assigned to an XML dictionary. The XML dictionary must be created before importing the DTD or schema into Data Interchange Services client.

The following steps describe how to create an XML dictionary.

1. Open the XML Dictionary Definition editor. Do this by clicking the XML button on the main application window, then move to the XML Dictionaries tab and click the New button.

2. In the General tab of the XML Dictionary Definition editor, type a name for the XML dictionary definition in the Dictionary Definition field.

3. Continue entering information in the remaining fields, then click the Save button on the menu bar. The new XML dictionary appears at the bottom of the XML Dictionary list window until you re-open it, at which point it appears in alphabetical order.

Creating EDI dictionaries

EDI document definitions are assigned to an EDI dictionary. The EDI dictionary must be created before creating EDI document definitions using Data Interchange Services client.

The following steps describe how to create an EDI dictionary.

1. Open the EDI Dictionary Definition editor. Do this by clicking **EDI** on the main application window, then move to the EDI Dictionaries tab and click **New**.
2. In the General tab of the EDI Dictionary Definition editor, type a name for the EDI dictionary definition in the Dictionary Definition field.
3. Continue entering information in the remaining fields, then click **Save** on the menu bar. The new EDI dictionary appears at the bottom of the EDI Dictionary list window until you re-open it, at which point it appears in alphabetical order.

Creating ROD dictionaries

ROD document definitions are assigned to an ROD dictionary. The ROD dictionary must be created before creating ROD document definitions using Data Interchange Services client.

The following steps describe how to create an ROD dictionary.

Note: In addition to requiring a ROD dictionary, a ROD document definition also requires a Record ID information object to indicate whether the ROD document definition also uses the Record ID information object to ensure that all loops and records within the ROD document definition use the same Record ID information object.

1. Open the ROD Dictionary Definition editor. Do this by clicking the ROD button on the main application window, then move to the ROD Dictionaries tab and click the New button.
2. In the General tab of the ROD Dictionary Definition editor, type a name for the ROD dictionary definition in the Dictionary Definition field.
3. Continue entering information in the remaining fields, then click the Save button on the menu bar. The new ROD dictionary appears at the bottom of the ROD Dictionary list window until you re-open it, at which point it appears in alphabetical order.

Creating ROD Record ID information objects

You create a new record ID information object when you set up your first ROD document definition or when you set up an ROD document definition whose record IDs are structured differently than your existing data formats. You can use the same record ID information object for any ROD document definition whose record IDs have the same structure or that are set up as C and D records.

The following steps describe how to create a data format record ID information profile.

1. Open the Record ID Information Object editor. Do this by clicking the ROD button on the main application window, then move to the Record ID Information Objects tab and click the New icon.
2. In the General tab of the Record ID Information Object editor, enter the following information:
   - Name: Type a name in the Name field. This is a required field, as indicated by the blue asterisk. The first and last characters can be A-Z and 0-9. All middle characters can be A-Z, 0-9, _ and --. You cannot type spaces within the name. The name can be 30 characters long
   - Description: You can optionally type a description of the data format record ID information profile.
   - Raw Data: If your application uses raw data, select the Raw Data radio button, then enter the following information in the Record ID group box:

-**Position**: Type the position of the first character of the Record ID in the Position field. -**Length**: Type the length of the Record ID in the Length field. -**Data Type**: Select the data type from the Data Type drop-down list.

- C and D Records: If your application uses C and D format, the Position, Length, and Data Type fields are filled in automatically.

3. Click the Save button on the toolbar. The new Record ID information object appears at the bottom of the Record ID Information Object list window until you re-open it, at which point it appears in alphabetical order.

Creating document definitions

Data Interchange Services client allows you to create three types of document definitions: XML, EDI, and ROD.

Creating DTD document definitions

You create a new DTD or schema document definition when you plan to process an XML document defined by an XML DTD or schema which has not previously been imported into Data Interchange Services client.

The following steps describe how to create a new DTD or schema document definition.

Note: DTD and schema document definitions are assigned to an XML dictionary. The XML dictionary must be created before importing the DTD into Data Interchange Services client.

1. Select **Open Import File** from the File menu. This opens the Select Import File page.
2. In the Files of Type drop-down list, select XML DTD File. The window changes so only files with a type of "DTD" are displayed.
3. Enter the name of the file to select a directory and file name of the DTD to be imported. Click **Open** when you have selected your DTD file. The Select a Database page appears if you have more than one database defined to Data Interchange Services client. If you have only one database defined, the Import XML DTD page appears
4. If the Select a Database page appears, select the Data Interchange Services client database you want to import the DTD into, then click **OK**. The Import XML DTD page appears.
5. On the Import XML DTD page, enter the appropriate information and click **Import**.

At this point the DTD is imported into a DTD document definition, which is created as part of the import.

Important: The name of the DTD document definition must be unique amongst DTD document definitions and schema document definitions within the same dictionary. An imported DTD replaces any existing DTD document definition or schema document definition that has the same name.

You can now use the DTD Document Definition Editor to view or update the DTD document definition.

Creating EDI document definitions

The EDI Document Definition editor allows you to define and structure the items that make up an EDI document definition. From the EDI Document Definition editor, you can create and edit EDI segments that are a part of this EDI document definition.

The following steps describe how to create a new EDI document definition.

Important: Typically, EDI document definitions can be downloaded from the Web and imported, and therefore do not need to be created.

Note: EDI document definitions are assigned to an EDI dictionary. The EDI dictionary must be created before creating EDI document definitions using Data Interchange Services client.

1. Open the EDI Document Definition editor. Do this by clicking **EDI** on the main application window, then clicking the **New** icon.
2. In the EDI Document Definition editor, General tab, enter a name for EDI document definition.
3. You may optionally enter a more complete description of the EDI document definition in the Description field, and a brief summary of the EDI document definition's purpose in the Purpose field.
4. Select the EDI dictionary in which you want the EDI document definition to appear through the Dictionary Name drop-down list. This is a required field.
5. You can optionally enter a functional group, such as IN for invoice.
6. Click the Details tab. The Details tab contains a grid that is used to identify the list of EDI segments that make up the EDI document definition. Each row in the grid represents an EDI segment. Use the information in the table below to create the EDI segments needed to create the EDI document definition.
7. Click the Comments tab. You can optionally type a comment.
8. When you have completed entering information, click the Save button on the toolbar to save the EDI document definition. Your new EDI document definition appears at the bottom of the EDI Document Definitions list window until you reopen it, at which point it appears in alphabetical order.

EDI segment values contained in Details tab

Value	Description
Table	The value in this field indicates the table in which the EDI Segment belongs. A table is a section of the EDI document definition. Many EDI document definition have only a single table (numbered "1". Some EDI document definitions have a grouping that uses the value "1" to represent header information in the document, the value "2" to represent the details of the document, and the value "3" to represent the trailer information of the document. The valid values for the field are 1 to 32,767.

Value	Description
Position	The value in this field indicates the relative position of the EDI segment within the specific table of the EDI document definition. Each value must be unique within the table. The values do not have to be sequential. When the EDI document definition is saved, the list of EDI segments will be sorted based on the table and position number. The valid values for this field are 1 to 32,000.
Segment	This column is used to identify the name of the EDI segment. Select a name from the list of available EDI segments.
Requirement Designator	Use this column to indicate whether the EDI segment is optional, mandatory, conditional, or floating. The conditional value indicates that the EDI segment may or may not be present depending on semantic conditions. Use the EDI Notes action to see a list of Notes or to maintain Notes for the EDI document definition.
Maximum Repeat	This value indicates the maximum number of times the EDI segment can occur at this position. The value can be from 1 to 9,998. This column will be disabled if the Unlimited Repeat column is set. The first EDI segment within a Loop should have the Maximum Repeat column set to 1.
Unlimited Repeat	This check box is used to indicate that the EDI segment can repeat infinitely. A set check box indicates the EDI segment can repeat infinitely. A reset check box indicates the EDI segment can not repeat infinitely. When the check box is set, the Maximum Repeat column will be disabled. The first EDI segment within a Loop should not repeat infinitely.
Loop ID	The ID of the group of related EDI segments if the EDI segment is part of a Loop. By convention, a Loop is the same as a segment group in UN/EDIFACT, ODETTE, or TRADACOMS EDI Standards. The Loop Level and Loop ID columns are used together to determine the EDI Segments that are part of a specific Loop.
Maximum Loop Repeat	This value indicates the maximum number of times the Loop can repeat. The value can be from 0 to 999,998. This column will be disabled if the Unlimited Loop Repeat column is set. A value should be entered only for the first EDI Segment within a Loop. The first EDI Segment within a Loop should have the Maximum Repeat column set to 1. A value of zero in this column is treated as one.

Value	Description
Unlimited Loop Repeat	This check box is used to indicate that the Loop can repeat infinitely. A set check box indicates the Loop can repeat infinitely. A reset check box indicates the Loop can not repeat infinitely. When the check box is set, the Maximum Loop Repeat column will be disabled.
Loop Level	The Loop Level indicates the current nesting level of a Loop. EDI Segments that are not within a Loop should have a value of zero in this column. When a Loop starts, the Loop Level for each EDI Segment at that level of the Loop should have a value of "1". If a Loop is nested within that first Loop, then that nested Loop Level should be "2" to indicate the second level of looping. All EDI segments that are part of nested Loop should have the Loop Level of "2". Nested Loops can be up to 16 levels deep. The Loop Level and Loop ID columns are used together to determine the EDI segments that are part of a specific Loop.
Description	This column describes the EDI segment. This value in this column can not be changed.

Creating ROD document definitions

You create a new ROD document definition when you need to define the layout of a proprietary document that will be processed by Data Interchange Services client.

The following steps describe how to create a new ROD document definition.

Note: ROD document definitions are assigned to an ROD dictionary. The ROD dictionary must be created before creating ROD document definitions using Data Interchange Services client. A ROD document definition also requires a Record ID information object to indicate whether the ROD document definition also uses the Record ID information object to ensure that all loops and records within the ROD document definition use the same Record ID information object.

1. Open the ROD Document Definition editor. The following steps describe how to open the ROD Document Definition editor.

    a. Click **Record Oriented Data** on the main application window. The Record Oriented Data list window opens.

    b. Click **New** on the Record Oriented Data list window's toolbar. The Record Oriented Data editor opens.

2. On the General tab page, type in the name of the ROD document definition. The name can be up to sixteen characters long and can contain alphanumeric characters and any of the special characters "!@#$%&*()_-+=';:<,>.?". It is suggested that the name not start with an ampersand ("&") or dollar sign ("$"). ROD document definitions provided by Data Interchange Services client may begin with either of these two characters. Also, the name of a ROD document definition must be unique against the names of all ROD document definitions regardless of the ROD dictionary.

3. From the Dictionary Name drop-down list, select the name of the ROD dictionary that will contain the ROD document definition. The ROD document

definition is able to utilize all Loops, Records, Structures, and Fields defined in the dictionary. Using a dictionary is a convenient way to reuse Loop definitions, Record definitions, Structure definitions, and Field definitions.

4. You can optionally type a description of the ROD document definition in the Description field.

5. From the Record ID Information drop-down list, select the name of the Record ID Information object to associate with the ROD document definition. The Record ID Information object indicates whether the ROD document definition will be C and D records format or Raw Data format, and it indicates the position and length of the field that contains the record identifier.

6. In the Records Delimiter drop-down field, select the type of delimiter that will terminate Records. If no delimiter is required, select No Delimiter.

7. If each Record is delimited, then field delimited records are supported. Checking the Delimited Fields checkbox indicates that fields in a Record are not fixed in position, but instead are separated by a delimiter. Enter the delimiter into the Field Delimiter field.

8. From the Character Set drop-down list, select the character set used in the document defined by this ROD document definition, if it uses a code page other than the system default code page.

   Note: In WebSphere Partner Gateway, the selected character set is used as a default if no character encoding is specified for the document definition.

9. In the Byte Order group box, select the byte order as needed. The Byte Order options are: Native Order, Big Endian, and Little Endian.

10. Click on the Details tab to display the Details tab page. Use this tab page to identify the first level of Loops and Records that are contained in the ROD document definition. Refer to the help topic on that tab page for additional information.**Important**: Complete the General tab page before working on the Details tab page when creating a new ROD document definition. This is necessary because some information needed to fill the Details tab page is obtained from the General tab page. Changing the name of the ROD document definition, which dictionary the ROD document definition belongs to, or the Record ID Information object used by the ROD document definition on the General tab page causes all information on the Details tab page to be reset. This is not an issue with existing ROD document definitions since those field can not be changed on existing ROD document definition. There are two ways to approach providing information for the Details tab. One method is to create the Loops and Records that are part of the first level of the ROD document definition by typing the new loops and records into the Details tab page. After the ROD document definition has been saved, you will need to go open the ROD Loop editor and ROD Record editor for each loop and record created. In the editor you will have to complete the definition for each loop and record. This in turn will require you to create structures and fields that are a part of your ROD document definition. Another method for creating a ROD document definition is to first create all of the fields that are a part of the ROD document definition. Then create all of the structures that are a part of the ROD document definition. Follow this up by creating all of the records that are a part of the ROD document definition. Create the loops that are a part of the ROD document definition. Finally, create the ROD document definition. In this scenario, you use the Details tab page of the ROD Document Definition editor to select the existing loops and records that are a part of the first level of the ROD document definition.

11. If you are using Raw Data format (as opposed to C and D records format) you must click the Raw Data tab to display the Raw Data page. On this page,

identify either the first or last record within the ROD document definition. Records you have defined in a ROD document definition do not have be in the order they are defined to the ROD document definition, so one of these fields must be filled in when using raw data format so Data Interchange Services client can determine where each document begins and ends. The following list describes all of the available fields.**Note for Beginning Records and Ending Records**: The drop-down list will be empty if you have not created any records for the ROD dictionary. You must create the record using the ROD Record Editor first, and then return to this field to select the name of the record.**Caution for Beginning Records and Ending Records**: The records in the list are all of the records that qualify for use with the ROD dictionary. You will get a warning message when you save the ROD document definition if you select a record that is not currently used in the ROD document definition. You must make sure that the record is used as the first record (for Beginning Record field) or as the last record (for the Ending Record field) within the ROD document definition.**Note for ROD fields**: The drop-down list will be empty if you have not created any ROD fields for this ROD dictionary. You must create the field using the ROD Field editor first, and then return to this field to select the name of the ROD field.**Caution for ROD fields**: The fields in the list are all of the fields defined within the ROD Dictionary. You will get a warning message upon saving if you select a field that is not currently used in the ROD document definition. You must make sure that the field is used within the ROD document definition. Although reuse is allowed with other ROD fields, this field must occur in your ROD document definition only once.

- **Beginning Record field**: Select the name of the record that is the first record within the ROD document definition. You must enter a value in this field or the Ending Record field if you are using Raw Data format.
- **Ending Record field**: Select the name of the record that is the last record within the ROD document definition. You must enter a value in this field or the Beginning Record field if Raw Data format will be used.
- **Field Containing the Interchange Sender Qualifier field**: This field contains the name of the ROD field that will contain the interchange sender qualifier. It is necessary to select a value for this field only if the interchange sender qualifier will appear in the document defined by the ROD document definition. The interchange sender qualifier found in the document will be used as the interchange sender ID in the EDI envelope header segment when the document defined by this ROD document definition is used as a source document in a translation. It will override the value provided by WebSphere Partner Gateway. A Record Oriented Data document being created during translation will have the interchange sender qualifier value from the EDI envelope header segment placed into the ROD field identified in this field.
- **Field Containing the Interchange Sender ID field**: This field contains the name of the ROD field that will contain the interchange sender ID. It is necessary to select a value for this field only if the interchange sender ID will appear in the document defined by the ROD document definition. The interchange sender ID found in the document will be used as the interchange sender ID in the EDI envelope header segment when the document defined by this ROD document definition is used as a source document in a translation. It will override the value provided by WebSphere Partner Gateway. A Record Oriented Data document being created during translation will have the interchange sender ID value from the EDI envelope header segment placed into the ROD field identified in this field.

- **Field Containing the Interchange Receiver Qualifier field**: This field contains the name of the ROD field that will contain the interchange receiver qualifier. It is necessary to select a value for this field only if the interchange receiver qualifier will appear in the document defined by the ROD document definition. The interchange receiver qualifier found in the document will be used as the interchange receiver qualifier in the EDI envelope header segment when the document defined by this ROD document definition is used as a source document in a translation. It will override the value provided by WebSphere Partner Gateway. A Record Oriented Data document being created during translation will have the interchange receiver qualifier value from the EDI envelope header segment placed into the ROD field identified in this field.

- **Field Containing the Interchange Receiver ID field**: This field contains the name of the ROD field that will contain the interchange receiver ID. It is necessary to select a value for this field only if the interchange receiver ID will appear in the document defined by the ROD document definition. The interchange receiver ID found in the document will be used as the interchange receiver ID in the EDI envelope header segment when the document defined by this ROD document definition is used as a source document in a translation. It will override the value provided by WebSphere Partner Gateway. A Record Oriented Data document being created during translation will have the interchange receiver ID value from the EDI envelope header segment placed into the ROD field identified in this field.

- **Field Containing the Sending Trading Partner Business ID field**: A field in a document can contain the business ID of the sending trading partner. If the document contains such a field, identify the name of corresponding ROD field in this field.

- **Field Containing the Receiving Trading Partner Business ID field**: A field in a document can contain the business ID of the receiving trading partner. If the document contains such a field, identify the name of corresponding ROD field in this field.

12. Click the Comments tab and add any comments you have about the ROD document definition.

13. Click **Save** on the toolbar. Data Interchange Services client saves the new ROD document definition to the database.

14. Click the Overview tab to display the Overview tab page. This tab page displays a graphical representation of your ROD document definition. The information on this tab page is refreshed each time the ROD document definition is saved.

15. Close the editor when you are done.

Importing COBOL copybooks:

Importing allows you to take a COBOL source file and use it to create WebSphere Partner Gateway fields, structures, and records, and then save them in a Record Oriented Data dictionary.

The following steps describe how to import a COBOL copybook.

1. From the main application window, click **Record Oriented Data**.

2. From the ROD Dictionaries tab, double-click the dictionary in which you want to store the new records and fields from the COBOL copybook.

3. Click I**mport COBOL Copybook**. The Select COBOL Copybook File dialog box appears.

4. Enter the correct path and file name, and click **Open**.

5. Click **Close** when the status dialog box indicates the import is complete. The path and file name of the copybook file display in the Copybook File field, and the Record ID info drop-down list and **New** button are enabled.

Creating data transformation maps

Data transformation maps are used to transform a document from its current format into another format. Often, an existing data transformation map can be used with conditional mapping commands to produce similar documents with minor variations, but if there is no similar data transformation map, a new map must be created.

The following steps describe how to create a new data transformation map.

1. Open the Data Transformation Maps list window by clicking the Mapping button in Data Interchange Services client, then clicking the Data Transformation Maps tab.

2. Click **New** on the Data Transformation Maps list window. The Create a Data Transformation Map - Map Name wizard appears.

3. Enter the name of the map and a description, then click **Next**.

4. Indicate whether the map will be source- or target- based, then click **Next**. In general, it is better to make the map source-based. The map must be target-based if the target document definition contains hierarchical loops and you need to use the special hierarchical loop mapping command (HLLevel()) to create the hierarchical loops. You may also want to make the map target-based if you are very familiar with the target document definition and not very familiar with the source document definition. Source-based maps usually result in more efficient translation of a document.

5. Select the syntax type of source document definition, then click **Next**.

6. Select the name of the dictionary that contains the source document definition, then click **Next**.

7. Select the name of the document definition, then click **Next**.

8. Specify the syntax type of target document definition, then click **Next**.

9. Select the name of the dictionary that contains the target document definition, then click Next.

10. Select the name of the target document definition, then click **Next**.

11. On the Confirmation page, ensure that the information displayed is correct, then click **Finish**. The map information you entered is saved to the database, the wizard closes, and the Data Transformation Map editor opens.

12. Click the Comments tab, and type any information you have about the map into the Comments field.

13. Click the Details tab. In the Details page, perform the mapping tasks necessary to create the mapping commands that instruct the translator component of WebSphere Partner Gateway how to translate the document. At this point, use the various available mapping techniques to create a map. This might include a "drag and drop" from a source field to target field, or the specification of an "assignment command" to set a target field to a specific value. Other mapping commands and mapping functions can be use to format data, convert data from one value to another, or qualify the movement of data based on the content of the source message. "Conditional statements" can be used to provide alternate courses of action for the translator component.

14. Click **Save** when you are finished. Data Interchange Services client saves the map to the database. It is recommended that you periodically click the Save button to save your changes to the database while you are mapping. The first time a map is saved after mapping commands and comments have been entered, the save process may take a while, especially when the map is based on a large EDI document definition. This is because every node displayed in the Mapping Command page is saved to the database this one time. Subsequent saves will only save changes to the map.

15. Click **File** → **Close** to close the editor when you are finished creating the map.

Creating validation maps

Validation maps are used when you receive or create EDI documents that requires additional validation beyond what is automatically provided by Data Interchange Services client. Sometimes existing maps can be used with conditional mapping commands to produce similar validations with minor variations, depending on the document received from the trading partner. If there is no similar existing map, then a new map must be created.

The following steps describe how to create a new validation map.

1. Open the Validation Maps list window by clicking the Mapping button in Data Interchange Services client, then clicking the Validation Maps tab.

2. Click **New** on the Validation Maps list window. The Create a Validation Map - Map Name wizard appears.

3. Enter the name of the map and a description, then click **Next**.

4. Select the name of the dictionary that contains the source document definition, then click **Next**.

5. Select the name of the document definition, then click **Next**.

6. On the Confirmation page, ensure that the information displayed is correct, then click **Finish**. The map information you entered is saved to the database, the wizard closes, and the Validation Map editor opens.

7. Click the Comments tab, and type any information you have about the map into the Comments field.

8. Click the Details tab. In the Details page, perform the mapping tasks necessary to create the mapping commands that instruct the translator component of WebSphere Partner Gateway how to perform additional validation of the document. Many of the techniques used in data transformation maps can be used here. The FAERROR mapping command is specific to validation maps.

9. Click the Save button when you are finished. Data Interchange Services client saves the map to the database. It is recommended that you periodically click the Save button to save your changes to the database while you are mapping. The first time a map is saved after mapping commands and comments have been entered, the save process may take a while, especially when the map is based on a large EDI document definition. This is because every node displayed in the Mapping Command page is saved to the database this one time. Subsequent saves will only save changes to the map.

10. Click **File** → **Close** to close the editor when you are finished creating the map.

Creating functional acknowledgment maps

Functional acknowledgment maps are used to create custom functional acknowledgments. Often, an existing functional acknowledgment map can be used by copying the map and then making changes in the copied map. If there is no similar existing functional acknowledgment map, then a new map must be created.

The following steps describe how to create a new functional acknowledgment map.

1. Open the Functional acknowledgment Maps list window by clicking the Mapping button in Data Interchange Services client, then clicking the Functional Acknowledgment Maps tab.

2. Click **New** on the Functional Acknowledgment Maps list window. The Create a Functional Acknowledgment Map - Map Name wizard appears.

3. Enter the name of the map and a description, then click **Next**.

4. Indicate whether the map will be source- or target- based, then click **Next**. In general, it is better to make the map source-based. You may want to make the map target-based if you are very familiar with the target document definition and not very familiar with the source document definition. Source-based maps usually result in more efficient translation of a document.

5. Select the name of the document definition that describes the layout of the functional acknowledgment, then click **Next**. The list you can select from displays all 997, 999, and CONTRL EDI document definitions that are defined in the current database. The target function acknowledgment must be one of the document definitions in this list.

6. On the Confirmation page, ensure that the information displayed is correct, then click **Finish**. The map information you entered is saved to the database, the wizard closes, and the Functional Acknowledgment Map editor opens.

7. Click the Comments tab, and type any information you have about the map into the Comments field.

8. Click the Details tab. In the Details page, perform the mapping tasks necessary to create the mapping commands that instruct the translator component of WebSphere Partner Gateway how to create a functional acknowledgment. Use the Details page on-line help for complete information on performing mapping. At this point, use the various available mapping techniques to create a map. This might include a "drag and drop" from a source field to target field, or the specification of an "assignment command" to set a target field to a specific value. Other mapping commands and mapping functions can be use to format data, convert data from one value to another, or qualify the movement of data based on the content of the source message. "Conditional statements" can be used to provide alternate courses of action for the translator component.

9. Click **Save** when you are finished. Data Interchange Services client saves the map to the database. It is recommended that you periodically click **Save** to save your changes to the database while you are mapping. The first time a map is saved after mapping commands and comments have been entered, the save process may take a while, especially when the map is based on a large EDI document definition. This is because every node displayed in the Mapping Command page is saved to the database this one time. Subsequent saves only save changes to the map.

10. Click **File** → **Close** to close the editor when you are finished creating the map.

Mapping techniques

Using the following mapping techniques will make your mapping project easier and more efficient.

Common mapping techniques
- Commands and command groups
- Comments and comment groups
- Variables
- Drag and drop
- Simple elements
- Compound elements

Commands and command groups

Mapping commands are instructions that tell Data Interchange Services client what operation needs to be performed to accomplish a very specific task. Put a coordinated set of mapping commands together and a document can be transformed from one format to another. Mapping commands are represented as nodes on the tree in the Mapping Command window. Most mapping commands can be placed almost anywhere on the tree contained in the Mapping Command window. Some placement limitations exist, depending on the command and its parent node. Mapping commands can exist on their own or be contained within a command group.

A command group generally contains a set of related mapping commands and comments. A mapping command is displayed as a single node on the tree in the Mapping Command window. Grouping a set of related commands and comments together allows the set of related commands and comments to collapse to a single node on the tree. You can copy the command group instead of having to move or copy each mapping command or comment individually. Command groups can contain a description. Use the description to describe the mapping commands and comments contained within the command group.

Important: The following mapping commands cannot be placed into a command group: Qualify(), Default, ForEach(), HLLevel(), HLDefault, and HLAutoMapped.

The following icons are used to display the various mapping command related nodes on the tree in the Mapping Command window:

Map icons

Icon	Description
	Represents most mapping commands.
	Represents a command group.
	Represents conditional commands such as If()/ElseIf()/Else/EndIf.
	Represents the Qualify() and Default commands.
	Represents the ForEach() command.

Icon	Description
	Represents the special hierarchical loop mapping command (HLLevel()).
	Represents the special default hierarchical loop mapping command (HLDefault).
	Positioned to the right of a mapping command icon to indicate there is an error in the mapping command. A description of the error is contained within parentheses following the mapping command.

Comments and comment groups

Comments are nodes in the Mapping Command window that contain a comment describing anything useful about the map. They can describe what the mapping commands that follow the comment accomplish or they can document changes to the map. Comments can also be useful to document change history or revisions to a particular map. For example, comments can specify the date, time, person who made the change, and why the change was made. A comment can be placed almost anywhere in the tree contained in the Mapping Command window. Each comment is displayed as a single node on the tree in the Mapping Command window. Comments can exist on their own or be contained within a comment group.

A comment group is used to group a set of related comments together. This allows the set of related comments to collapse to a single node on the tree. You can move or copy the comment group instead of having to move or copy each comment individually. Comment groups can contain a description. Use the description to describe the comments contained within the comment group. The following figure shows comments documenting changes to the map. The comments are contained within a comment group. This is one method of creating a revision list for your map.

Grouped comments
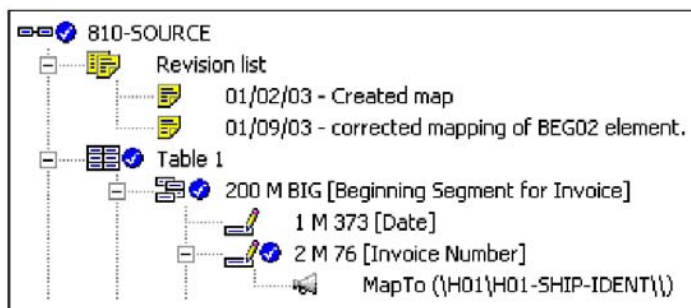
The following icons are used to display comments and comment groups on the tree in the Mapping Command window:

Comment icons

Icon	Description
	Represents a comment

Icon	Description
🖹	Indicates a comment group

Variables

Variables are used in programming languages to store and manipulate data. Maps also support variables. Three types of variables are supported: local variables, global variables, and special variables. Variables are used in mapping commands and functions, commonly within expressions.

Variables must exist before they can be used in mapping commands and functions. Use the Variables window to create the needed Global Variable or Local Variable. To use an existing variable, click on the variable, continue to hold the mouse button down, then drag the variable to a simple element in the Source or Target Document Definition window or to another variable. Once over the desired position, release the mouse button to "drop" the variable. Performing this action will create an assignment statement in the Mapping Command window.

In source-based maps, the assignment statement is placed within the corresponding simple element in the Mapping Command window when the variable is dropped on a simple element in the Source Document Definition window or in the Mapping Command window. If the drop target is a simple element in the Target Document Definition window or is another variable, the assignment statement is created within the selected node in the Mapping Command window. This is true unless the node does not support embedded commands, in which case the assignment statement is placed after the selected node. When the variable is dropped on a simple element in the Source Document Definition window or in the Mapping Command window, the assignment command results in the data contained in the simple element being copied to the variable. When the variable is dropped on a simple element in the target document definition, the assignment statement results in the data contained in the variable being copied to the simple element in the target document. When a variable is dropped on another variable, the assignment statement results in the data contained in the variable being dragged being copied to the variable that it was dropped on.

In target-based maps, the assignment statement is placed within the corresponding simple element in the Mapping Command window when the variable is dropped on a simple element in the Target Document Definition window or in the Mapping Command window. If the drop target is a simple element in the Source Document Definition window or is another variable, the assignment statement is created within the selected node in the Mapping Command window. This is true unless the node does not support embedded commands, in which case the assignment statement is placed after the selected node. When the variable is dropped on a simple element in the Target Document Definition window or in the Mapping Command window, the assignment command results in the data in the variable being copied to the simple element. When the variable is dropped on a simple element in the Source Document Definition window, the assignment statement results in the data contained in the simple element in the source document being copied to the variable. When a variable is dropped on another variable, the assignment statement results in the data in the variable being copied to the variable that it was dropped on.

Variables can also be dragged to the Mapping Command Editor. Drop the variable in the command wherever needed. Names of variable can be typed into the

Mapping Command Editor also. Use variables in mapping commands and functions as needed. The data within variables can be freely manipulated using the various commands and functions that are available with the exception of special variables. Data can be assigned to special variables, but data can not be obtained from special variables.

Local variables

Local variables are unique to the map they are defined in. A local variable must be defined to a map before it can be used in that map.

Local variables have a scope of document or loop. During translation, local variables defined with a scope of document will be created at the start of every document and deleted at the end of the document. Variables defined with a scope of loop will be created and initialized whenever a new loop iteration is started, and destroyed at the end of each loop iteration. A loop variable does not disturb the value of another variable with the same name at another level of looping. Local variables are maintained within the map they are defined. A user may add, delete, and alter the properties of any local variable.

Global variables

Global variables are similar to local variables, except they are not unique to any map; they are shared across data transformation maps, validation maps, and functional acknowledgment maps.

Global variables have a scope that is group- or interchange-oriented. Variables that have a scope of interchange are created when an interchange is encountered. They are reset at the start of the next interchange. Variables with a scope of group are created when a group is encountered. They are reset at the start of the next group. Not all syntax types support the concept of groups or interchanges. When the source document type does not support groups, the scope of group is treated the same as interchange. If the document type does not support interchanges, the variable is reset at the start of each input document. For example, if the input file contains multiple document and is split into separate documents by the RODSplitter or XMLSplitter, the global variables are reset for each input ROD or XML document in the file.

The difference between a global variable and a local variable is the scope of the variable. The scope of a variable indicates when a variable is created during translation and how long it will exist. Local variables are created at the document or loop level. They are map-specific and are created and maintained within a map editor. Global variables are independent of the document and can span documents. Global variables can be created and updated within a map; however, general maintenance of global variables is performed from the Global Variables list window.

Special variables

Special variables are a group of predefined variables used by Data Interchange Services client. They function much like local variables or global variables, except that they each have a special purpose.

A user can view the properties of a special variable, but no changes can be made. Special variable names always start with "DI". It is recommended that you do not start local variables or global variable names with "DI".

Special Variables are write-only. That means that you can set the value of a special variable, but you cannot read it or use it in an expression. Special variables are used in data transformation maps and functional acknowledgment maps. They are currently not useful in validation maps.

The following special variable is supported:
- **Name** : DICUserData
- **Scope** : Document
- **Type**: Character
- **Len**: 256
- **Description**: Used to set the data value that will be inserted into any output C record before the C record is written. Any combination of data up to 256 bytes can be placed into this variable.

Dragging elements

Data Interchange Services client allows you to perform many mapping tasks by dragging elements from one location to another.

You can drag elements to perform the following mapping tasks:
- Create a mapping command to associate an element in the source document definition with an element in the target document definition
- Create a mapping command to copy data contained in a simple element in the source document to a variable
- Create a mapping command to copy data contained in a variable to a simple element in the target document
- Put the path identifying an element into a mapping command in the Mapping Command Editor
- Put the name of an element or a position number into a parameter of a mapping command within the Mapping Command Editor
- Put the name of a variable into the Mapping Command Editor
- Move a mapping command, command group, comment, or comment group. Mapping commands and command groups that have been added at a particular location may also be moved or copied to another location by dragging the command. To move a command, select the command and drag it to the new location.
- Copy a mapping command, command group, comment, or comment group. To copy a command or command group to a new location, select the command and press Shift, then drag it to the new location. Be sure to check the commands to ensure they have the appropriate source and target path specifications at the new location.

Important: For moving or copying commands, command groups, comments, or comments groups, if the new location is not visible in the mapping window, j hold the drag and move up or down in the mapping window and Data Interchange Service client automatically scrolls up and down. If the new location has not been expanded (indicated by a '+' sign), hold the drag over the node and Data Interchange Service client will automatically expand that node.

Using the Mapping Command window

The Mapping Command window allows you to apply the advanced mapping capabilities of Data Interchange Services client.

The following steps describe how to use the Mapping Command window.

1. Open the Mapping Command window. The following steps describe how to do this.

   a. From the main application window, click the Mapping button. This opens the Mapping list window.

   b. From the Mapping list window, select the tab that represents the type of map you want to work with. The choices are: Data Transformation Map, Validation Map, and Functional Acknowledgment Map. The map list window for your selected map type appears.

   c. From the map list window, double-click the map you want to work with. The map loads, then the Map Command window opens. The following figure displays the Mapping Command window.
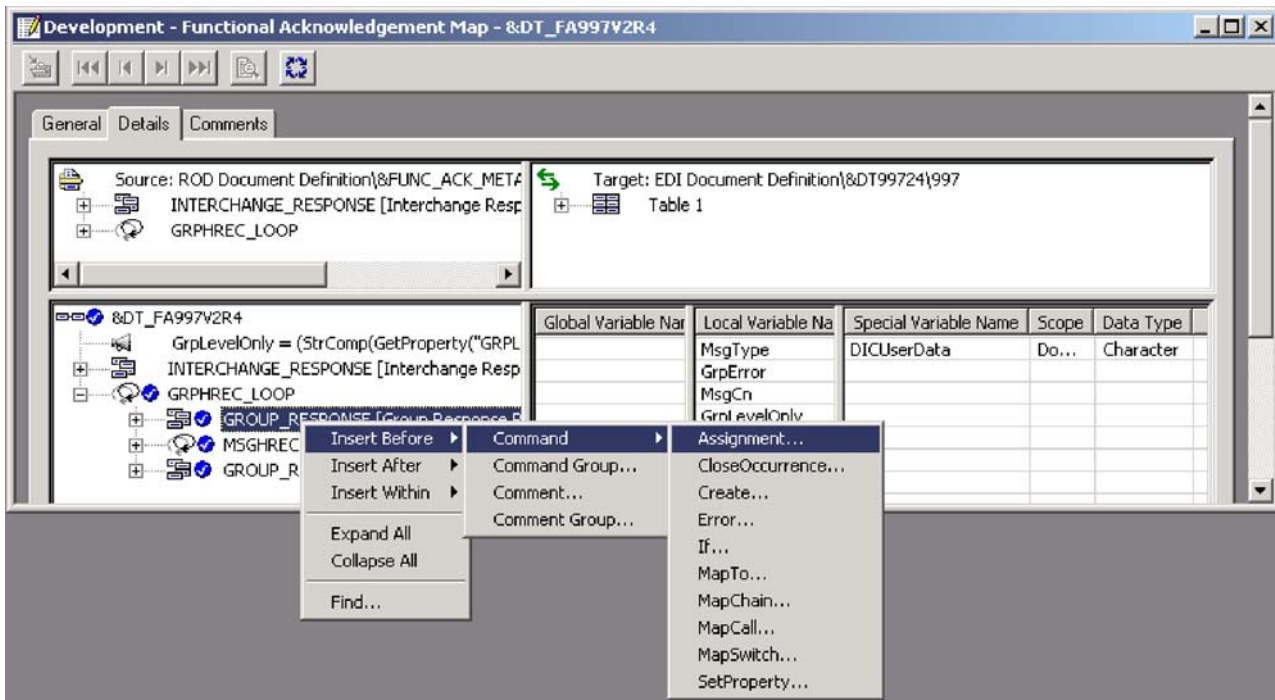
2. Right-click an element in the Mapping Command window and follow the cascading menus. You can choose to insert a command, command group, comment, or comment group. You can choose to insert your selection before, after, or within the element you have selected.

3. If you want to insert a command, select where the command should be inserted, select Command, and then select the command to be inserted. The Mapping Command Editor appears with a prototype of the command.

4. Edit the prototype to create the command you need. A right-click on the prototype parameters will display a list (if any) of the possible commands and functions available for that parameter. A click or selection on the command or function will replace the prototype parameter with the selection. For functions, the parameter will be replaced with the prototype for that function.

5. Click **Insert** if you want to add the command, and create another one similar to it.

6. Click **OK** when you have your last command ready.

Note: The command list contains a "Find" command. This can be useful in finding a particular mapping command, source path, target path, or variable.

Simple elements

You can use a drag and drop operation to create a mapping command that will copy data from a simple element in the source document to a simple element in the target document.

In a source-based map, the drag operation creates a MapTo() command. The command is inserted into the Mapping Command window within the element corresponding to the source element in the drag operation. Execution of the command during translation of the source document results in data contained in the element in the source document being copied to the element in the target document. If the simple element in the source document definition is repeating, the mapping command is executed each time the source element is encountered in the source document. When this occurs and the simple element in the target document definition is also repeating, a new occurrence of the element in the target document is created for each occurrence of the simple element in the source document. If the simple element in the target document definition is not repeating, then each occurrence of the simple element in the source document overwrites the data contained in the one simple element in the target document. If the simple element does not exist in the source document, the target element will not be created.

In source-based maps, the Mapping Command window contains a representation of the source document layout, similar to the document definition displayed in the Source Document Definition window. Elements in the Mapping Command window correspond to elements in the Source Document Definition window. As a result, simple elements in the Mapping Command window can be dragged instead of simple elements in the Source Document Definition window. The one exception to this is involving simple elements within nested hierarchical loops. When mapping simple elements within nested hierarchical loops, perform drag operations using the Mapping Command window instead of the Source Document Definition window. This ensures that the paths associated with the nested elements include the nesting levels when placed in mapping commands.

For source-based maps, if there is a qualification on the simple element (Qualify() and Default commands), the MapTo() mapping command is inserted within the qualification that contains the current selected node. If the current selected node is not within one of the qualifications, the command is inserted into the qualification that is expanded if only one is expanded. If it cannot be determined where the command should be inserted, an error message is displayed asking you to select a qualification and then repeat the drag operation.

In target-based maps, a drag of a simple element from the Source Document Definition window to the Target Document Definition window creates a MapFrom() command. The command is inserted into the Mapping Command window within the element corresponding to the target element in the drag operation. Execution of the MapFrom() command during translation of the source document results in data contained in the corresponding source element being copied to the element in the target document. If the simple element in the Target Document Definition window occurs more than once (repeats), a ForEach() command is also inserted in the Mapping Command window. The ForEach() command will be inserted into the Mapping Command window within the element corresponding to the target element in the drag operation. The MapFrom()

command is placed within the ForEach() command. The ForEach() command indicates that for every occurrence of the simple element in the source document, an occurrence of the simple element in the target document is created. A single simple element can have multiple ForEach() commands associated with it.

In target-based maps, the Mapping Command window contains a representation of the target document layout, similar to the document definition displayed in the Target Document Definition window. Elements in the Mapping Command window correspond to elements in the Target Document Definition window. As a result, drag operations can end on simple elements in the Mapping Command window instead of ending on simple elements in the Target Document Definition window. The one exception to this is involving simple elements within nested hierarchical loops. When mapping simple elements within nested hierarchical loops, perform drag operations using the Mapping Command window instead of the Target Document Definition window. This ensures that the paths associated with the nested elements include the nesting levels when placed in mapping commands.

Compound elements

Generally, you can drag and drop operations to map compound elements.

In a source-based map, the drag operation will create a MapTo() command. The command is inserted into the Mapping Command window within the element corresponding to the source element in the drag operation. Execution of the command during translation of the source document results in the creation of an occurrence of the compound element in the target document each time the compound element in the source document occurs.

In source-based maps, the Mapping Command window contains a representation of the source document layout, similar to the document definition displayed in the Source Document Definition window. Elements in the Mapping Command window correspond to elements in the Source Document Definition window. As a result, compound elements in the Mapping Command window can be dragged instead of compound elements in the Source Document Definition window. The one exception to this is involving compound elements within nested hierarchical loops. When mapping compound elements within nested hierarchical loops, perform drag operations from the Mapping Command window instead of the Source Document Definition window. This ensures that the paths associated with the nested elements include the nesting levels when placed in mapping commands.

For source-based maps, if there is a qualification on the compound element (Qualify() and Default commands), the MapTo() mapping command is inserted within the qualification that contains the current selected node. If the current selected node is not within one of the qualifications, the command is inserted into the qualification that is expanded if only one is expanded. If it cannot be determined where the command should to be inserted, an error message is displayed asking you to select a qualification and then repeat the drag operation.

In target-based maps, a drag of a compound element from the Source Document Definition window to the Target Document Definition window will create a ForEach() command. The command is inserted into the Mapping Command window within the element corresponding to the target element in the drag operation. Execution of the ForEach() command during translation of the source document results in an occurrence of the compound element in the target document being created for every occurrence of the compound element in the source document, unless the compound element in the target document is

non-repeating. In this case, only one occurrence of the compound element in the target document is created. Subsequent occurrences of the compound element in the source document may overwrite the one occurrence of the compound element in the target document definition. If the compound element in the target document does not repeat, you may need to use qualification (Qualify() and Default commands) or conditional mapping commands (If()/ElseIf()/Else/EndIf) to ensure a second occurrence of the compound element in the source document definition does not overwrite the first occurrence of the compound element in the target document. A single compound element can have multiple ForEach() commands associated with it.

In target-based maps, the Mapping Command window contains a representation of the target document layout, similar to the document definition displayed in the Target Document Definition window. Elements in the Mapping Command window correspond to elements in the Target Document Definition window. As a result, drag operations can end on compound elements in the Mapping Command window instead of ending on compound elements in the Target Document Definition window. The one exception to this is involving compound elements within nested hierarchical loops. When mapping compound elements within nested hierarchical loops, perform drag operations in the Mapping Command window instead of the Target Document Definition window. This ensures that the paths associated with the nested elements include the nesting levels when placed in mapping commands.

Chapter 5. Creating map objects

Map objects allow you to customize maps.

The following steps describe how to access the map objects.
1. Click **Mapping** to open the Mapping functional area.
2. Select one of the following tabs: **User Exits**, **Code Lists**, **Translation Tables**, or **Global Variables**.

Creating User Exit profiles

You create a new User Exit profile whenever your have a user program or exit routine that will be called by the translator component of WebSphere Partner Gateway.

The following steps describe how to create a User Exit profile.
1. Open the User Exits editor. The following steps describe how to do this.
   a. Click **Mapping** on the main application window.
   b. In the User Exits tab, click **New**. The User Exit editor opens.
2. On the General tab of the User Exit editor, type in the name of the User Exit profile in the Exit Name field.
3. In the Description field, you can optionally type a description of the User Exit profile.
4. From the Program Language drop-down list, select the programming language in which the exit routine is written.
5. In the Class Name field, type the name of the class in which the exit routine is contained.
6. In the Method Name field, type the name of the exit routine.
7. Leave the Output Length field as is; it is reserved for future use.
8. Leave the String Type group box as is; it is reserved for future use.
9. Click the Comments tab, and add any comments you have about the User Exit profile.
10. Click **Save** on the toolbar. Data Interchange Services client saves the new User Exit profile to the database.
11. Close the editor when you are done.

Creating code lists

You create a new code list when you need to ensure a value is valid during translation. For instance, you may want to ensure that a simple element contains a valid part number.

The following steps describe how to create a new code list.
1. Open the Code Lists editor. The following steps describe how to do this.
   a. Click **Mapping** on the main application window.
   b. In the Code Lists tab, click **New** to open the Code List editor.

2. On the General tab of the Code List editor, type in the name of the code list into the Name field. This value must be unique among translation tables and code lists.

3. In the Description field, you can optionally type a description of the code list.

4. In the Data Type field, indicate whether the values contained in the code list will be character or numeric.

5. In the Maximum Length field, identify the maximum length of the values. The length can be up to 35 characters long.

6. Click **New** to open the Add New Code List Entry dialog to create the code list entries. Each entry identifies a valid value within the code list. In the dialog, enter a value that is to be considered valid and its corresponding description. Click **Insert** to insert the entry. The dialog remains open so another entry can be made. Click **OK** to add the last entry to the code list, or click **Cancel** if the last entry has already been made.

7. Click the Comments tab, and type any comments you have about the code list.

8. Click **Save** on the toolbar. Data Interchange Services client saves the new code list to the database.

9. Close the editor when you are done.

Creating translation tables

You create a new translation table when you need to translate a value in a document to a corresponding value. For instance, a simple element that contains an internal part number that must be converted to a trading partner's corresponding part number.

The following steps describe how to create a new translation table.

1. Open the Translation Table editor. The following steps describe how to do this.

   a. Click **Mapping**g on the main application window.

   b. In the Translation Tables tab, click **New**. The Translation Table editor opens.

2. On the General tab of the Translation Table editor, type in the name of the translation table in the Name field.

3. You can optionally type a description of the translation table in the Description field.

4. In the Source Value group box, indicate whether the source values will be character or numeric in the Data Type field.

5. In the Source Value group box, identify the maximum length of the source values in the Maximum Length field. The length can be up to 35 characters long.

6. In the Target Values group box, indicate whether the target values will be character or numeric in the Data Type field.

7. In the Target Value group box, identify the maximum length of the target values in the Maximum Length field. The length can be up to 63 characters long. The combined length of the source and target values can not exceed 68 characters.

8. Click **New** to open the Add New Translation Table Entry dialog to create the translation table entries. Each entry associates a source value with a target value. In the dialog, enter a source value and its corresponding target value. Click the **Insert** to insert the entry. The dialog remains open so another

association between a source value and a target value can be made. Click **OK** to add the last entry to the translation table, or click **Cancel** if the last association has already been made.

9. Click the Comments tab, and add any comments you have about the translation table.

10. Click **Save** on the toolbar. Data Interchange Services client saves the new translation table to the database.

11. Close the editor when you are done.

Creating global variables

Create a new global variable when you need a variable for a data transformation map, validation map, or functional acknowledgment map that needs to retain its information beyond the current document being translated.

The following steps describe how to create a new global variable.

1. Open the Global Variables editor. The following steps describe how to do this.
    a. Click **Mapping** on the main application window.
    b. In the Global Variables tab, click **New**. The Global Variable editor opens.

2. On the General tab page, type in the name of the global variable into the Name field.

3. You can optionally type a description of the variable in the Description field.

4. Indicate the scope of the global variable as either Interchange or Group.

5. From the Data Type drop-down list, select the data type of the global variable. The choices are: Binary, Boolean, Character, Integer, and Real.

6. In the Maximum Length field, you can optionally set the maximum length of the global variable.

7. In the Initial Value field, type an initial value for the global variable.

8. Click **Save** on the toolbar. Data Interchange Services client saves the new global variable to the database.

9. Close the editor when you are done.

Chapter 6. Compiling maps

A map must be compiled into a map control string before the translator component of WebSphere Partner Gateway can perform translations related to the map. A map control string is a compiled representation of a map. A map itself is never directly used to translate a document.

Data Interchange Services client uses the map as input to a program that compiles the map to produce a control string. The source and target document definitions referenced by the map are also usually compiled into control strings during this process. The translator component of WebSphere Partner Gateway uses the control string in its translation processing. Before Data Interchange Services client can recognize a change related to a map, the map must be recompiled. You recompile a map in the following situations:

- when the map is changed
- when source document definition is changed
- when target document definition is changed
- when a global variable used by the map is changed

Compiling data transformation maps

A data transformation map needs to be compiled into a map control string before the translator component of WebSphere Partner Gateway can perform translations related to the map.

The following steps describe how to compile a data transformation map.

1. Open the Data Transformation Maps list window by clicking the Mapping button on the Data Interchange Services client navigator bar, then selecting the Data Transformation Maps tab.
2. Select one or more maps in the Data Transformation Maps list window.
3. Click **Compile** on the toolbar, or select **Compile** from the Actions menu.

Note: Alternatively, you can compile data transformation maps from the Data Transformation Map editor. From the editor, press the Compile button on the toolbar, or select Compile from the Actions menu. If a change has been made in the editor, a dialog appears asking if the change can be saved. Click **OK** to save the change and continue the compile process. Click **Cancel** if you do not want to save the change or compile the map.

Data Interchange Services client compiles the selected data transformation maps and displays an Execution Status window. When Data Interchange Services client is finished compiling, a message appears on the Execution Status window indicating that the compile has been completed. You can click **Cancel** on the Execution Status window to stop the compile function.

In addition to compiling or recompiling the map, the map control string must be copied to the database accessed by the translator component of WebSphere Partner Gateway. This is commonly done using the export and import functions.

Note: Compiling a data transformation map involves a large amount of database access. If your database is on a network, it may be preferable to move your map

and document definitions to a local database and do the compile on the local database. After the compile has completed, the control string can be exported to the desired database.

Compiling validation maps

A validation map needs to be compiled into a map control string before the translator component of WebSphere Partner Gateway can perform additional validation on an EDI document using the map.

The following steps describe how to compile a validation map.

1. Open the Validation Maps list window by clicking the Mapping button on the Data Interchange Services client Navigator bar, then clicking the Validation Maps tab.
2. Select one or more maps in the Validation Maps list window.
3. Click **Compile** on the toolbar, or select **Compile** from the Actions menu.

Note: Alternatively, you can compile validation maps from the Validation Map editor. From the editor, click **Compile** on the toolbar or select **Compile** from the Actions menu. If a change has been made in the editor, a dialog appears asking if the change can be saved. Click **OK** to save the change and continue the compile process. Click **Cancel** if you do not want to save the change or compile the map.

Data Interchange Services client compiles the selected validation maps and displays an Execution Status window. When Data Interchange Services client is finished compiling, a message appears on the Execution Status window indicating that the compile has been completed. You can click **Cancel** on the Execution Status window to stop the compile function.

In addition to compiling or recompiling the map, the map control string must be copied to the database accessed by the translator component of WebSphere Partner Gateway. This is commonly done using the export and import functions.

Note: Compiling a validation map involves a large amount of database access. If your database is on a network, it may be preferable to move your map and document definition to a local database and do the compile on the local database. After the compile has completed, the control string can be exported to the desired database.

Compiling functional acknowledgment maps

A functional acknowledgment map needs to be compiled into a map control string before the translator component of WebSphere Partner Gateway can create a functional acknowledgment based on the map.

The following steps describe how to compile a functional acknowledgment map.

1. Open the Functional Acknowledgment Maps list window by clicking **Mapping** n on the Data Interchange Services client navigator bar and selecting the Functional Acknowledgment Maps tab.
2. Select one or more maps in the Functional Acknowledgment Maps list window.
3. Click **Compil**e on the toolbar, or select **Compile** from the Actions menu.

Note: Alternatively, you can compile functional acknowledgment maps from the Functional Acknowledgment Map editor. From the editor, click the Compile button

on the toolbar, or select Compile from the Actions menu. If a change has been made in the editor, a dialog appears asking if the change can be saved. Click **OK** to save the change and continue the compile process. Click **Cancel** if you do not want to save the change or compile the map.

Data Interchange Services client compiles the selected functional acknowledgment maps and displays an Execution Status window. When Data Interchange Services client is finished compiling, a message appears on the Execution Status window indicating that the compile has been completed. You can click **Cancel** on the Execution Status window to stop the compile function.

In addition to compiling or recompiling the map, the map control string must be copied to the database accessed by the translator component of WebSphere Partner Gateway. This is commonly done using the export and import functions.

Note: Compiling a functional acknowledgment map involves a large amount of database access. If your database is on a network, it may be preferable to move your map and document definitions to a local database and do the compile on the local database. After the compile has completed, the control string can be exported to the desired database.

Recompiling control strings

Map control strings should be recompiled any time the corresponding map is changed or any time the related source or target document definition is changed. Changes are not available to the translator component of WebSphere Partner Gateway until the recompile has been successfully completed and the new control string has been exported to the database accessed by the translator component.

The following steps describe how to recompile map control string.

1. Open the Control Strings list window by clicking **Mapping** on the Data Interchange Services client navigator bar and selecting the Control Strings tab.
2. Select one or more map control strings in the Control Strings list window.
3. Click **Compile** on the toolbar, or select **Compile** from the Actions menu.

Data Interchange Services client recompiles the selected control strings and displays an Execution Status window. When Data Interchange Services client is finished recompiling, a message appears on the Execution Status window indicating that the compile has been completed. You can click **Cancel** on the Execution Status window to stop the compile function.

In addition to recompiling the map control string, you must also copy it to the database accessed by the translator component of WebSphere Partner Gateway. This is commonly done using the export and import functions.

Chapter 7. Importing and exporting data

The import process is used to move previously exported data from a file into a database. The import process can also import XML schemas and XML DTDs. Data Interchange Services client uses an export process to move information out of Data Interchange Services client. During the export process, the data is extracted from a database and then formatted and written to a file.

Importing data

The import process is typically used to perform the following tasks:

- Install EDI Standards into a Data Interchange Services client database.
- Import information into another database
- Save data as a backup
- Share data with another Data Interchange Services client user

The following steps describe how to import data from a file.

1. Select the **Open Import File** function from the File menu. The Select Import File dialog appears.

2. In the Select Import File dialog, specify the name of an existing import file. Typically Data Interchange Services client export files have a file extension of *.eif*. The import file can be in any directory. Click **Open** when the name of the import file has been selected. The Import File window appears. This window displays the contents of the selected import file. If you decide this is not the file you want to import, you can click **Open** on the toolbar or select **Open** from the File menu to select an alternate file to open in the window.

3. Select one or more items to import from the Import File window, then click **Import** on the toolbar. Alternatively, you can select Import from the Actions menu once you have selected the files to import.

4. If you have more than one database defined in Data Interchange Services client, the Select a Database dialog appears. Select the database to which the items are to be imported, and then click **OK**.

5. The selected items begin importing. The Execution Status window appears showing the progress of the import. The Execution Status window can be closed at any time. You do not have to wait until the import is completed.

6. Close the Import File window when you are done with the import file.

The file read by the import process and created by the export process is interchangeably called an import file or an export file. Though the two terms are commonly used, they refer to the same file.

During the import process, you may receive a message indicating that an item you are importing already exists in the database. The message asks if you want to replace the existing item with item being imported. Indicating you do not want to replace the existing item will cancel the import process. Indicating that you do want replace the existing item will result in the existing item being replaced in the database. If there are associated or referenced items being imported as well, they will also be replaced.

When Data Interchange Services client is finished importing the selected items, a message appears in the Execution Status window indicating the import has completed. A check mark is placed next to the imported items in the tree view to indicate that these items have been imported. Parent nodes are marked as imported when all child nodes have been imported. A check mark next to the root node indicates that all items that may be imported in the file have been imported.

Importing XML schemas

The format of an XML document can be defined by an XML schema. Schemas can be obtained from various organizations or can be created by the user using an editor of some sort. Import any XML schema that you want to use in a data transformation map.

Schemas are stored in Data Interchange Services client within schema document definitions. Import an XML schema to create a new schema document definition when you plan to process an XML document defined by an XML schema.

Note: Schema document definitions are assigned to an XML dictionary. The XML Dictionary must be created before importing the schema into Data Interchange Services client.

The following steps describe how to import an XML schema.

1. Select **Open Import File** from the File menu. This opens the Select Import File dialog. This is similar to most common Windows dialogs used specify the name of the file.

2. Select XML Schema File in the Files of Type drop-down list. The window changes so only files with a type of "xsd" are displayed.

3. Enter the name of the file or use the dialog to select a directory and file name of the schema to be imported. Click **Open** when you have selected your schema file. The Select a Database dialog appears if you have more than one database defined to Data Interchange Services client. If you only have one database defined, the Import XML Schema dialog appears.

4. If the Select a Database dialog appears, select the Data Interchange Services client database you wish to import the schema into, then click **OK**. The Import XML Schema dialog appears.

5. On the Import XML Schema dialog, enter the appropriate information and click **Import**.

At this point the schema is imported into a schema document definition which is created as part of the import.

Note: The name of the schema document definition must be unique amongst schema document definitions and DTD document definitions within the same dictionary. An imported schema will replace any existing schema document definition or DTD document definition that has the same name.

Use the Schema Document Definition editor to view and update a schema document definition.

Data Interchange Services client scans the schema when it is imported looking for a targetNamespace attribute. If one is found, the Target Namespace field is filled with the corresponding value. The target namespace associated with the schema can be changed, but typically it should not be changed unless the import was not able to find it correctly. Setting the target namespace to a value that does not

match the `targetNamespace` attribute can prevent Data Interchange Services client from parsing the schema correctly, which results in errors when trying to use the schema document definition in a map, and it can also prevent the map from opening.

If the schema contains an "xmlns" attribute to identify a namespace, and the namespace is not already represented as a Namespace object within Data Interchange Services client, a new Namespace object is added to represent the namespace at the time the schema is imported. The prefix and other information associated with Namespace objects can be changed later.

Importing XML DTDs

The format of an XML document can be defined by a document type definition (DTD). DTDs can be obtained from various organizations or can be created by a user. Import any XML DTD into Data Interchange Services client that you want to use in a data transformation map.

Document Type Definitions (DTDs) are stored in Data Interchange Services client within DTD document definitions. Import an XML DTD to create a new DTD document definition when you plan to process an XML document defined by an XML DTD.

Note: DTD document definitions are assigned to an XML dictionary. The XML dictionary must be created before importing the DTD into Data Interchange Services client.

The following steps describe how to import an XML DTD.

1. Select **Open Import Fil**e from the File menu. This opens the Select Import File dialog. This is similar to most common Windows dialogs used specify the name of the file.
2. Select XML DTD File in the Files of Type drop-down list. The window changes so only files with a type of "DTD" are displayed.
3. Enter the name of the file or use the dialog to select a directory and file name of the DTD to be imported. Click **Open** when you have selected your DTD file. The Select a Database dialog appears if you have more than one database defined to Data Interchange Services client. If you only have one database defined, the Import XML DTD dialog appears.
4. If the Select a Database dialog appears, select the Data Interchange Services client database you want to import the DTD into and then click **OK**. The Import XML DTD dialog appears.
5. On the Import XML DTD dialog, enter the appropriate information and click **Import**.

At this point the DTD is imported into a DTD document definition which is created as part of the import.

Note: The name of the DTD document definition must be unique amongst DTD document definitions and Schema document definitions within the same dictionary. An imported DTD will replace any existing DTD document definition or schema document definition that has the same name.

Use the DTD Document Definition editor to view and update a DTD document definition.

Exporting data

The following steps describe how to export data from Data Interchange Server client.

1. Select one or more items from a list window.

2. Click **Export** on the toolbar or select the **Export To File** function from the Actions menu. The Select Export File dialog is displayed. This is similar to most common Windows dialogs used specify the name of the file. In this dialog, specify the name of the export file. Any existing file name can be selected or the name of a new file can be entered. Typically Data Interchange Services client export files have a file extension of *.eif*. If an existing file is specified, it must contain only valid data previously exported from Data Interchange Services client. The export file can be in any directory you desire. Click **Open** when the name of the export file has been entered or selected.

3. If a new file is specified, the Select File Format dialog appears. Use this dialog to identify the export format you wish to use. The recommended format is tagged. Click **OK** to continue. This dialog does not appear if you selected an existing export file. Data exported to an existing export file will be exported in the format specified when the file was created.

4. Certain types of information in Data Interchange Services client refer to other types of information or are dependent on other information. These are referred to as referenced types and associated types, respectively. If the items you are exporting have referenced or associated types, a dialog appears asking which referenced and associated types you want to export with the selected items. Click **OK** once you have specified any referenced or associated types to be exported with your selected items. If the items you are exporting do not have referenced or associated types, then no dialog appears. Each item you selected is exported. Referenced and associated types are exported with the selected items, depending on your selections. The Execution Status window appears showing the progress of the export. The Execution Status window can be closed at any time. You do not have to wait until the export is completed.

Once the selected items are exported, the export file can be used to perform the following tasks:

- Import information into another database
- Save information as a backup
- Share information with another Data Interchange Services client user

Note: The file created by the export process and input to the import process is interchangeably called an export file or an import file. Though the two terms are commonly used, they refer to the same file.

Using Release Migration

Release Migration is a feature that allows you to copy data from or to Data Interchange Services client. It is often used to copy data from one release of Data Interchange Services client to another. It can also be used to back up and restore data or to copy all data from one database to another.

The following steps describe how to use Release Migration.

1. From the main application window, select **Release Migration** from the View menu. The Release Migration wizard opens.

2. From the Export or Import page, select either **Export Migration Data** or **Import Migration Data**, then click **Next**.

3. From the Select Data Type page, select the data type you want to export or import, then click **Next**. The choices for exporting are: Database Data, Configuration Data, or Both. The choices for importing are: Database Data or Configuration Data.

4. From the Select Folder page, specify the folder that will hold the exported or imported data, then click **Next**.

5. From the Confirmation page, confirm that the export or import information is correct, then click **Finish**. The Execution Status window appears, displaying the data that is being migrated. A `Release migration completed` message appears at the bottom of the Execution Status window when the migration is complete.

Chapter 8. Generating reports

Data Interchange Services client allows you to create reports for maps, document definitions, and User Exit profiles.

Generating data transformation map reports

The Report function is used to display one or more data transformation maps selected in a Data Transformation Maps list window. The report is displayed in the default browser on your system. From there, you can choose to print the report, close the window, or save the report in an HTML file.

The following steps describe how to create a report for a data transformation map.

1. Open the Data Transformation Maps list window by clicking the Mapping button on the Data Interchange Services client navigator bar, then selecting the Data Transformation Maps tab.

2. Select one or more data transformation maps, and then select **Report** from the File menu or click **Report** on the toolbar. The report for each selected data transformation map is generated and displayed in a browser, and an Execution Status Window appears displaying the success or failure of each report generated.

3. Close the browser windows when you are done with the report.

Generating validation map reports

The Report function is used to display one or more validation maps selected in a Validation Maps list window. The report is displayed in the default browser on your system. From there, you can choose to print the report, close the window, or save the report in an HTML file.

The following steps describe how to create a report for a validation map.

1. Open the Validation Maps list window by clicking the Mapping button on the Data Interchange Services client navigator bar, then selecting the Validation Maps tab.

2. Select one or more validation maps, and then select Report from the File menu or click the Report button on the toolbar. The report for each selected validation map is generated and displayed in a browser, and an Execution Status Window appears displaying the success or failure of each report generated.

3. Close the browser windows when you are done with the report.

Generating functional acknowledgment map reports

The Report function is used to display one or more functional acknowledgment maps selected in a Functional Acknowledgment Maps list window. The report is displayed in the default browser on your system. From there, you can choose to print the report, close the window, or save the report in an HTML file.

The following steps describe how to create a report for a functional acknowledgment map.

1. Open the Functional Acknowledgment Maps list window by clicking the Mapping button on the Data Interchange Services client navigator bar, then selecting the Functional Acknowledgment Maps tab.

2. Select one or more functional acknowledgment maps, and then select Report from the File menu or click the Report button on the toolbar. The report for each selected functional acknowledgment map is generated and displayed in a browser, and an Execution Status Window appears displaying the success or failure of each report generated.

3. Close the browser windows when you are done with the report.

Generating EDI document definition reports

The Report function can be used to display one or more EDI document definitions selected in an EDI Document Definitions list window or the current EDI document definition in the EDI Document Definitions editor. The report is displayed in the default browser on your system. From there, you can choose to print the report, close the window, or save the report in an HTML file.

The following steps describe how to create a report for an EDI document definition.

1. Open the EDI Document Definitions list window by clicking **EDI** on the Data Interchange Services client navigator bar, then selecting the EDI Document Definitions tab.

2. Select one or more EDI document definitions, and then select **Report** from the File menu, or click **Report** on the toolbar. The report for each selected EDI document definition is generated and displayed in a browser, and an Execution Status Window appears displaying the success or failure of each report generated.

3. Close the browser windows when you are done with the report.

To generate a report for an EDI document definition opened in an editor, select **Report** from the File menu or click **Report** on the toolbar. The report is generated and then displayed in a browser window.

Generating ROD document definition reports

The Report function can be used to display one or more ROD document definitions selected in an ROD Document Definitions list window or the current ROD document definition in the ROD Document Definitions editor. The report is displayed in the default browser on your system. From there, you can choose to print the report, close the window, or save the report in an HTML file.

The following steps describe how to create a report for an ROD document definition.

1. Open the ROD Document Definitions list window by clicking **ROD** on the Data Interchange Services client navigator bar, then selecting the ROD Document Definitions tab.

2. Select one or more ROD document definitions, and then select **Report** from the File menu, or click **Report** on the toolbar. The report for each selected ROD document definition is generated and displayed in a browser, and an Execution Status Window appears displaying the success or failure of each report generated.

3. Close the browser windows when you are done with the report.

To generate a report for an ROD document definition opened in an editor, select **Report** from the File menu or click **Report** on the toolbar. The report is generated and then displayed in a browser window.

Generating user exit profile reports

The Report function is used to display one or more user exit profiles selected in a User Exits list window. The report is displayed in the default browser on your system. From there, you can choose to print the report, close the window, or save the report in an HTML file.

The following steps describe how to create a report for a User Exit profile.

1. Click **Mapping** on the Data Interchange Services client navigator bar to open the User Exits list window, then select the User Exits tab.
2. Select one or more User Exit profiles, and then select **Report** from the File menu or click **Report** on the toolbar. The report for each selected User Exit profile is generated and displayed in a browser, and an Execution Status Window appears displaying the success or failure of each report generated.
3. Close the browser windows when you are done with the report.

To generate a report for a User Exit profile opened in an editor, select **Report** from the File menu or click **Report** on the toolbar. The report is generated and then displayed in a browser window.

Chapter 9. Mapping reference

In addition to allowing you to map a source element to a target element by dragging and dropping, Data Interchange Services client includes a powerful set of mapping commands. This allows you to use conditional logic such as if/then/else, qualify repeating elements, save values in variables, create complex expressions, and many other functions.

This section describes the command language used in the Data Transformation Map editor to perform these tasks. When you create a map, you are basically creating a sequence of commands. Each command is attached to an object in the source or target document, such as a segment, data element, field, XML element, and so on. Whenever the translator finds an object in the source or target document, it executes any commands associated with it. You can also add commands before and after objects in the document. The translator executes these commands, even if the elements before and after are not found. You can create the commands by dragging and dropping a source element to a target element (or variable) or vice versa. This results in a **MapTo** (for source-based mapping) or **ForEach** (for target-based mapping) command or an assignment command being created. You can also create commands by right-clicking an element in the Mapping Command window, which opens the Mapping Command Editor.

The following sections provide examples of some of the supported commands and functions, as well as information about other important features of the mapping language.

Commands and functions

Mapping **commands** perform actions that are used to instruct Data Interchange Services client that a specific action is to occur, while mapping **functions** perform an action and return a result within an expression or assignment statement.

Mapping commands

Mapping commands instruct the translation process on how to move data from one element to another, how to manipulate data, how to process repeating compound elements, when to issue user specified errors, and how to perform conditional processing. Most commands require parameters, such as paths that identify where data is going to or coming from, variables, constants, and expressions.

Note: A command name is not case-sensitive. For example, the Error() command keyword could also be specified as ERROR().
An expression as a parameter in a command can resolve to any data type as long as the data type can be converted to the data type expected by the parameter. Parameters for a command are never modified by the command. In general, data transformation maps, validation maps, and functional acknowledgment maps support most commands. Exceptions will be documented with the command.

Mapping commands example

CloseOccurence: Use the CloseOccurrence command to close the current occurrence of a repeating element and force the creation of another instance of

the element. This command can be used for both source-based and target-based maps. The CloseOccurrence command has the following format:

CloseOccurrence(targetpath)

Where: **targetpath** specifies a repeating target element that is closed to any new elements in the current occurrence. Any additional mappings to this element or its children result in a new occurrence of the targetpath element. The CloseOccurrence command can be shortened to CloseOccur.

Example

Suppose your source document has the following structure:

Rec1

Field1

Field2

Field3

Field4

Suppose your target document has the following structure:

Seg (repeats)

Seg01

Seg02

Field1 and Field2 are to be mapped to the first occurrence of Seg (Seg01 and Seg02), and Field3 and Field4 to the second occurrence of Seg (again, Seg01 and Seg02).

The following steps describe how to accomplish this.

1. Move down to the node in the Mapping Command window where you want the commands to be executed.
2. Right click the node, select **Insert within**, **Insert Before**, or **Insert After** depending on where you want the commands to execute.
3. Select **Commands**, then **Assignment**.

   This results in the following: `Path = expression`
4. Map Field1 to Seg01, Field2 to Seg02 using the Assignment Command, and drag/drop each source path to the "path" and the target path to the "expression." Drag Seg01 in the target document and drop on "path." Drag Field1 in the source document and drop on "expression."

   Result: `\Seg\Seg01\\ = \Rec1\Field1\\.`
5. Click **Insert**. Drag Seg02 in the target document and drop on "path." Drag Field2 in the source document and drop on "expression."

   Result: `\Seg\Seg02\\ = \Rec1\Field2\\.`
6. Click **OK**

7. Insert a CloseOccurrence(Seg) command to close the first occurrence of Seg. Right-click the assignment statement \Seg\Seg01\\ = \Rec1\Field2\\.

8. Select **Insert After**.

9. Select **Command** then **CloseOccurrence**.

   This results in the following: **CloseOccurrence (targetPath)**.

10. Drag Seg and drop on targetPath.

11. Map Field3 to Seg01, Field4 to Seg02, which creates a second occurrence of Seg.

12. Right-click the CloseOccurrence command. Select **Insert After**

13. Select **Command** then **assignment** Repeat Step 4 with the same target document elements Seg01 and Seg02 and source document elements Field3 and Field4.

Mapping functions

Mapping functions perform an action and return a result within an expression or assignment statement. All functions take 0 or more parameters as input. The number of parameters and the data type of the return value vary from one function to the next. Appropriate type conversions are done implicitly if needed (and possible). Some functions have optional parameters. If the optional parameters are omitted from the function call, a default value will be used for that parameter.

Most functions can take an expression as a parameter, as long as the result of the expression is (or can be converted to) the correct data type. For example, the Char() function converts a value to a character string. The command:

Var1 = Char (1 + 2)

Is equivalent to:

Var1 = Char (3)

Note: Function names are not case-sensitive. The function name Char() is the same as CHAR().
In general, data transformation maps, validation maps, and functional acknowledgment maps support most functions. Exceptions are documented with the function.

Mapping function example

DateCnv: The DateCnv function is used to assist in converting one date format to another. The DateCnv function uses the following format:

```
char DateCnv(char sourceDate, char frommask, char tomask)
```

The actual syntax requires quotes.

For example:

```
Mapping Path = DateCnv('20021216', 'CCYYMMDD', 'CCYY-MM-DD')
```

Where: **sourceDate** is the string that contains the source date, **frommask** is the mask that identifies the format of the date in *sourceDate*, and **tomask** is the mask that identifies the format of the date desired in the result string.

Results: The converted date is a character string.

Example: Map the target field PODate in POHeader record from the source field PurchaseOrderDate in OrderRecord record. The following steps describe how to do this.

1. Move down to the node in the Mapping Command window where you want the commands to be executed.
2. Right-click the node, select **Insert within**, **Insert Before**, or **Insert After** depending on where you want the commands to execute.
3. Select **Commands**, then **Assignment**.

   This results in the following: `Path = expression`.
4. Drag the target path PODate to the "path."

   Results: `\POHeader\PODate\\ = expression`.
5. Right-click the expression. This results in a list of functions to select.
6. Select **DateConv**.

   This results in the following: `\POHeader\PODate\\ = DateCnv (sourceDate, fromMask, toMask)`.
7. Drag the source path to 'sourceDate.'

   Results: `\POHeader\PODate\\ = DateCnv (\OrderRecord\ PurchaseOrderDate\\, fromMask, toMask)`
8. Add the frommask and tomask in quotes: \POHeader\PODate\\ = DateCnv (\OrderRecord\PurchaseOrderDate\\, 'CCYYMMDD', 'CCYY-MM-DD')
9. Click **OK**. Results: if source date value = 20021216 target date value = 2002-12-16.

Assignment statements

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

An assignment statement has the following format:

target = expression

Where:

target–is a path identifying a simple element in the target document definition or any defined variable.

expression–is any valid expression.

Data Interchange Services client attempts to convert the result from expression to the same data type of the target, if needed. If it is unable to make the conversion, an error is issued.

One method of creating an assignment statement is to drag a simple element from the source document definition or the target document definition and drop it on a variable. This creates an assignment command at the appropriate position within the Mapping Command window:

- When a map is source-based, dragging a simple element from the source document definition to a variable results in the creation of an assignment command within the corresponding simple element in the Mapping Command window.

- If a map is source-based and a variable is dragged to a simple element in the target document definition, an assignment command is created at the selected node within the Mapping Command window.
- When a map is target-based, dragging a variable to a simple element in the target document definition results in the creation of an assignment command within the corresponding simple element in the Mapping Command window.
- If a map is target-based and a simple element in the source document definition is dragged to a variable, an assignment command is created at the selected node within the Mapping Command window.

An assignment statement can also be created by right-clicking on a node where you want to insert the statement. In the popup menu that appears, select an available **Insert** action, then select **Command**, and then **Assignment**.

Doing this opens the Mapping Command editor, where you will need to enter the target. Insert the target by dragging the appropriate variable or simple element from the target document definition to the target parameter in the Mapping Command editor. Enter the expression as needed, then click **OK**. This inserts the assignment statement where requested.

While in the Mapping Command editor, you can click **Insert** instead of **OK**. Clicking **OK** creates the command at the appropriate place in the Mapping Command window and then closes the Mapping Command editor. Clicking **Insert** creates the command at the appropriate place in the Mapping Command window, but the Mapping Command editor remains open. Another command can then be entered. When **OK** or **Repeat** is selected, the new command is inserted in the Mapping Command editor immediately after the previously inserted command. Click **OK** when there are no more assignment commands to insert at the current position, or click **Cancel** if the last command at the current position has already been inserted.

Note: Elements in the source document definition are read-only. You cannot update values in the source document. Elements in the target document definition are write-only. You may not read the value that has been assigned to an element in the target document.

Expressions

Expressions are commonly used in the mapping command language in data transformation maps, validation maps, and functional acknowledgment maps.

The basic form of an expression is:

token operator token [operator token [operator token [...]]]

Where:

token – can be one of the following:
- A local or global variable
- A path identifying a simple element in the source document definition
- A numeric constant, such as 1024
- A string constant, such as "ABCD"

operator – is one of the following types of operators:
- Logical operator

- Comparison operator
- Arithmetic operator
- Unary operator

Quotation marks must be used at the start and end of a string constant. Either single (') or double quotes (") may be used, but whichever is used to start the string constant must also be used to end it.

Data Interchange Services client processes expressions from left to right following a set order of precedence. Precedence can be overridden with parentheses embedded within an expression.

All variables, constants, and simple elements have a specific data type. Data Interchange Services client converts the value to the appropriate data type for the expression. If it cannot convert the value to the appropriate type because the types are incompatible (for example, Boolean to binary), an error is issued during the control string compilation. If it cannot convert the data value to the appropriate type because the values are incompatible, (for example "xyz" to real), an error is issued during translation or a default value such as 0 is used.

Data transformation document properties

This section lists the properties that can be set as part of the data transformation map in Data Interchange Services client and their corresponding WebSphere Partner Gateway attributes. If these properties are set in the map, they take precedence over the WebSphere Partner Gateway definition or envelope profile attributes. Attributes listed for Document definition also apply to the B2B capabilities of the partners.

Source document map properties and their corresponding attributes

During runtime the WebSphere Partner Gateway configured attributes get manifested into the Business Document Object (BDO). The BDO contains the metadata about the document being processed. The properties in this table apply to source documents. These are typically used by the source EDI Validator and will be used for obtaining information from the BDO. Other maps may also use these values.

Data Interchange Services client property	Obtains the attribute values from WebSphere Partner Gateway	WebSphere Partner Gateway object
Alphanum	Alphanumeric validation table	Document definition
Charset	Char set validation table	Document definition
GrpLvlFA	Generate group level information only in functional acknowledgment	Document definition
ServSegVal	Detailed validation segment	Document definition
ValLevel	Validation level	Document definition
ValErrLevel	Maximum validation error level	Document definition
ValMap	Validation map	Document definition

Target document map properties and their corresponding attributes

During runtime the WebSphere Partner Gateway configured attributes get manifested into the Business Document Object (BDO). The BDO contains metadata about the document being processed. The properties in this table apply to target documents. They are used in subsequent processing of the target document. They are set by the data transformation map, which overrides the value configured in WebSphere Partner Gateway.

Note: Some of the listed Data Interchange Services client properties have no corresponding WebSphere Partner Gateway attributes, but if set, they can be used by subsequent components. These are indicated under the WebSphere Partner Gateway corresponding attribute column as "Does not apply."

Data Interchange Services client property	WebSphere Partner Gateway corresponding attribute	WebSphere Partner Gateway object
AckReq	Acknowledgment request	Envelope profile
Alphanum	Alphanumeric validation table	Document definition
Charset	Char set validation table	Document definition
CtlNumFlag	Control numbers by Transaction Id	Envelope profile
DIProlog	Does not apply	Runtime
DocType	Document type	Business Document Object (runtime)
EdiDecNot (Decimal notation)	Decimal notation	Document definition
EdiDeDlm (Data element separator)	Data element delimiter	Document definition
EdiDeSep (Repeating data element separator)	Repeating data element separator	Document definition
EdiRlsChar (Release character)	Release character	Document definition
EdiSeDlm (Component data element separator)	Subelement delimiter	Document definition
EdiSegDlm (Segment terminator)	Segment delimiter	Document definition
EnvProfName	Envelope profile	Document definition
EnvType	Envelope type	Envelope profile
MaxDocs	Maximum transactions number	Envelope profile
Package	Package	Business Document Object (runtime)
Protocol	Protocol	Business Document Object (runtime)
ReceiverTP	Target partner	Business Document Object (runtime)
Reroute	Reroute	Business Document Object (runtime)
SegOutput	Segment output	Document definition
SenderTP	Sender partner	Business Document Object (runtime)

Data Interchange Services client property	WebSphere Partner Gateway corresponding attribute	WebSphere Partner Gateway object
ValLevel	Validation level	Document definition
ValErrLevel	Maximum validation error level	Document definition
ValMap	Validation map	Document definition

Additional Data Interchange Services client properties and relationship to WebSphere Partner Gateway

The following table describes Data Interchange Services client properties that are generic to the EDI Envelope and their relationship to WebSphere Partner Gateway attributes. During runtime the WebSphere Partner Gateway configured attributes get manifested into the Business Document Object (BDO). The BDO contains the metadata about the document being processed.

When processing the source EDI Interchange document, the Data Interchange Services client properties can be used for reading the WebSphere Partner Gateway attributes from the BDO. For the target EDI Interchange document, the Data Interchange Services client properties can be used to override the WebSphere Partner Gateway attributes or to provide additional information in the target BDO.

Some of the listed Data Interchange Services client properties are not set from a map but are provided by a runtime component into a BDO and can then be read by a map or another runtime component. An example of such a property is GrpTrxCnt, which is indicated under the "WebSphere Partner Gateway corresponding attribute" column as "Does not apply."

Data Interchange Services client property	WebSphere Partner Gateway corresponding attribute	WebSphere Partner Gateway object
IchgCtlNum	Interchange control number	Control number initialization or status
IchgSndrQl	Interchange sender qualifier	Document definition
IchgSndrId	Interchange sender ID	Document definition
IchgRcvrQl	Business Identifier (EDI Interchange Qualifier)	Document definition
IchgRcvrId	Business Identifier (EDI Interchange Identifier)	Document definition
IchgDate	Does not apply	Runtime
IchgTime	Does not apply	Runtime
IchgPswd	Interchange password	Envelope profile
IchgUsgInd	Interchange usage indicator	Document definition Envelope profile
IchgAppRef	Interchange application reference	Envelope profile
IchgVerRel	Interchange - depends on the standard	Envelope profile
IchgGrpCnt	Does not apply	Runtime
IchgCtlTotal	Does not apply	Runtime
IchgTrxCnt	Does not apply	Runtime

Data Interchange Services client property	WebSphere Partner Gateway corresponding attribute	WebSphere Partner Gateway object
GrpCtlNum	Group control number	Control number initialization or status
GrpFuncGrpId	Functional group ID	Envelope profile
GrpAppSndrId	Group application (GS) sender ID	Document definition Envelope profile
GrpAppRcvrId	Group application (GS) receiver ID	Document definition Envelope profile
GrpDate	Does not apply	Runtime
GrpTime	Does not apply	Runtime
GrpPswd	Group password	Envelope profile
GrpVer	Group version	Envelope profile
GrpRel	Group release	Envelope profile
GrpTrxCnt	Does not apply	Runtime
TrxCtlNum	Transaction control number	Control number initialization or status
TrxCode	Does not apply	Runtime
TrxVer	Transaction version	Envelope profile
TrxRel	Transaction release	Envelope profile
TrxSegCnt	Does not apply	Runtime

Examples:

SetProperty: The SetProperty command is used to set a special processing property of the target message. Various special properties are defined to control things such as the EDI envelope fields or the XML prolog. The SetProperty command uses the following format:

SetProperty(char propertyName, char propertyValue)

Where: **propertyName** is the property name to be set in the target message. propertyValue Is the value that it should be set to.

A list of target document properties may be obtained by right clicking the propertyName in the command.

Example: Override the segment terminator in a particular map. The following steps describe how to do this:

1. Right-click the node, selectfrom the following: **Insert within**, **Insert Before**, or **Insert After**, depending on where you want the commands to execute.
2. Select **Commands**, then **SetProperty**. This results in the following: SetProperty (propertyName, propertyValue)
3. Right-click propertyName for a list of available properties. Select Target Document Properties.
4. Select EdiSegDlm. Results: SetProperty (″EdiSegDlm″, propertyValue)
5. Enter the override value in the propertyValue in quotes. Result: '%'
6. Click **OK**.

GetProperty: The GetProperty function is used to get a property of the source message. This can be used to retrieve information such as EDI envelope header elements. The GetProperty function uses the following format:

char GetProperty(char propertyName)

Where: **propertyName** is the name of the property you want to retrieve. Results: The value associated with the specified message property. If the propertyName is not set in the source message, an empty string is returned.

Example: var1 = GetProperty("ISA04") will set var1 to the value that was in the ISA04 element in the X12 ISA segment. The following steps describe this.

1. Move down to the node in the Mapping Command window where you want the commands to be executed.
2. Create variable var1. To do this, right-click in the Local Variable Window and select New.
3. Right-click the node, select "Insert within, Insert Before, or Insert After" depending on where you want the commands to execute.
4. Select Commands, then Assignment. This results in the following: Path = expression.
5. Drag the local variable var1 to the "path." Results: var1 = expression.
6. Right-click the expression. This results in a list of functions to select.
7. Select GetProperty. Results: var1 = GetProperty (propertyName).
8. Right-click propertyName for a list of properties. Select Specific Envelope Properties, then ISAnn. Results: var1 = GetProperty ("ISAnn").
9. Type over nn with 04. Results: var1 = GetProperty ("ISA04").
10. Click **OK**.

Reserved words

In data transformation maps, validation maps, and functional acknowledgment maps, reserved words consist of all keywords, mapping commands and functions, logical operators, comparison operators, and some additional words.

Case does not matter. For instance, "Create" and "CREATE" are the same therefor they are both reserved. The following words are considered reserved:

AND	ASSERT	BREAK
BY	CASE	CHAR
CLOSEOCCUR	CLOSEOCCURRENCE	CONCAT
CREATE	CREATED	DATE
DATECNV	DEFAULT	DO
ELSE	ELSEIF	ENDASSERT
ENDFOR	ENDIF	ENDSELECT
EQ	ERROR	EXIT
FAERROR	FALSE	FIND
FOR	FOREACH	FOUND
IF	GE	GETPROPERTY
GT	HEXDECODE	HEXENCODE
HL	HLAUTOMAPPED	HLDEFAULT
HLLEVEL	ISEMPTY	ITERATE
LE	LEFT	LENGTH

LOOPBREAK	LOWER	LT
MAPCHAIN	MAPCALL	MAPFROM
MAPSWITCH	MAPTO	NE
NOT	NUMBER	NUMFORMAT
OCCUR	OCCURRENCE	OR
OTHERWISE	OVERLAY	PATTERN
QUALIFY	RD	RIGHT
ROUND	SCALE	SELECT
SETPROPERTY	STRCOMP	STRCOMPI
STRCOMPN	STRCOMPNI	SUBSTRING
TARGETROOT	THIS	TIME
TO	TRANSLATE	TRIMLEFT
TRIMRIGHT	TRUE	TRUNCATE
TU	UPDATED	UPPER
UNTIL	VALIDATE	WHILE

Terminology

Some of the terminology in the Data Interchange Services client product and documentation differs from the terminology in the WebSphere Partner Gateway product and documentation. The following table maps the terminology between the two products.

Terminology mapping between Data Interchange Services client and WebSphere Partner Gateway

This Data Interchange Services client term...	Is equivalent to this WebSphere Partner Gateway term
Dictionary (EDI dictionary, ROD dictionary, or XML dictionary) - Used to logically group a set of EDI transactions, ROD definitions, or XML schemas and DTDs.	**Document definition**- Protocol. An example is an EDI X12V2R1 or an ROD ADF-TO-EDI_DICT. METADICTIONARY attribute in the ROD Splitter Handler.
Document definition (EDI document definition, ROD document definition, DTD document definition, or Schema document definition) - Defines a particular business document, such as a purchase order or invoice. **Note:** DTD and Schema document definitions are both under the XML functional area.	**Document definition** - Document type. An example is an EDI 850 or an ROD DTADF-TO-EDI_ADF. METADOCUMENT attribute in the ROD Splitter Handler for ROD documents.
Syntax type - These values are displayed as "EDI," "Record Oriented Data," and "XML."	**METASYNTAX** attribute in the ROD Splitter Handler. Value to use is "rod." Can also be used in the Generic Document Type Handler. Values used internally besides "rod" by the components/handlers are "xml," "ediIchg," (EDI Interchange), and "ediTrx" (EDI transaction).

This Data Interchange Services client term...	Is equivalent to this WebSphere Partner Gateway term
Data transformation map	**Transformation map** - Can apply to the Data Interchange Services client and other types of maps such as XSLT for XML Transformation.
Validation map	**Validation map** - can apply to Data Interchange Services client and other types such as XML Schema.
Functional acknowledgment map	**Functional acknowledgment map**

Chapter 10. XML reference

When processing XML data there are some additional things that you must consider that are not relevant for other types of source and target documents. These include understanding how Data Interchange Services client resolves the schemas and DTDs that it uses to validate the XML documents and how it handles different encoding types such as UTF-8.

XML considerations

Some special content types that are defined by schemas have limitations on their mapping capabilities.

The following content types have some limitations on their mapping capabilities:

- **xsd:anyType** – This means that this element can contain any type of content, including child elements, simple elements, or mixed content. Data Interchange Services client allows this to be mapped only as if it were a simple string element.
- **xsd:any** – This means that any child element may appear in this position, sometimes subject to namespace restrictions. Data Interchange Services client does not allow you to map to or from this element.
- **xsd:anyAttribute** – This means that any attribute may appear on this element. Data Interchange Services client does not allow you to map to or from this attribute.
- **Substitution groups** – This means that one element may be substituted for another. Data Interchange Services client does not allow mapping to or from the substitution group elements.

XML namespace processing

Namespaces allow the use of multiple XML schemas definitions within an XML document or building a grammar from several different schemas, by providing a way to resolve name conflicts between the schemas. For example, one schema may define a "address" element to be a mailing address that includes street, city, state, and zip code elements. Another may define "address" to be an email address that contains a simple string. By using different namespaces, these can be uniquely identified.

For example, to show that the element "address" belongs to a particular namespace, a schema or instance document defines a prefix for the namespace. For instance, it might define xmlns:po="http://example.com/ns/POExample". Then when the qualified element appears in the data, the element appears as <po:address>, to show that it is part of that namespace. The "po" prefix in this example is just a shorthand way to represent the namespace "http://example.com/ns/POExample". Even though the schema might use prefix "po", instance documents may use different prefixes, for example "mypo", "purch", or it may even define it as the default namespace, so no prefix is used for this element. As long as these were defined to the "http://example.com/ns/POExample" namespace in the instance documents, they should all be considered equivalent.

The mapping and translation for XML schemas is namespace-aware. This means that it recognizes that the prefix represents a namespace, and gives it special

treatment instead of just treating it as part of the element name. Since the schema translation and mapping is namespace-aware, elements with the same name and namespace would be considered to be equivalent, even if they used different prefixes.

When a schema is imported, the schema is scanned for any "xmlns" attributes. If an "xmlns" attribute is found for a namespace that is not already represented by a Namespace object in the XML Dictionary, a new Namespace object is created using the prefix defined for the attribute.

Namespace Processing for Input XML documents

You must indicate in the WebSphere Partner Gateway Console that namespace processing is to be performed if you are doing schema validation or your XML schema uses namespace qualified elements or attributes. Namespace processing is required for schema validation because the attributes that help locate the schema definitions ("xsi:schemaLocation" and "xsi:noNamespaceSchemaLocation") use namespace qualification. If your source or target document is a schema that uses qualified elements or attributes, namespace processing is required so the internal names used by the translation process will match the internal names used in the map.

Namespace Processing for Output XML Documents

When generating output XML documents, qualified elements and attributes use the prefix specified by the namespace table entry for the URI. If no prefix is specified, or if no namespace entry exists for a namespace URI, then the element or attribute is written without a prefix (i.e., using the default namespace).

Some special attributes are sometimes needed in the XML output to identify the namespaces used and the schema location. Several mapping commands exist to specify these.

Refer to each of the following commands in the Data Interchange Services online help for additional information:
- SetNoNSSchemaLocation ()
- SetSchemaLocation ()
- SetNamespace ()

Chapter 11. Specifying Hierarchical Loop Levels

Hierarchical loops are a way for EDI transactions to dynamically define the looping structure. The standard defines all segments which might be a part of a hierarchical loop but it is the application/standard data being received which adds structure to those segments. For example, an HL loop would be defined to contain segments A, B, C, D, E, but it is the application/standard data which defines that segments A, B, C form one level 1 inner loop, that segments D, E form another level 1 inner loop, and that segments D, E might form a level 2 inner loop within the A, B, C loop. Theoretically there is no limit to the nesting levels.

HL loops may be qualified and mapped using the usual qualification and mapping methods. However, WebSphere Partner Gateway's HL support allows you to define Hierarchical levels and specify unique mapping instructions for each identifiable group of structures in a hierarchical level (loop).

Creating an HL Loop Level

The following steps describe how to create an HL loop level as a base level within an EDI source or target message.

1. Right-click the HL Loop in the Mapping Command window and select Add HL Qualification. The HL Qualification dialog box appears.
2. Select the HL level code from the drop down list. This results in an HLLevel()command.

The HLLevel command has the following format:

HLLevel(levelcode)

where: **levelcode** indicates the level code for this level. You must select this value from the drop-down list of level codes that are valid for the code list for HL03 (element 735). This value identifies the context of a series of segments following the current HL up to the next occurrence of an HL, or the end of the loop.

To insert an HL loop level as a child of the current HL loop level:

1. Right-click the HLLevel Command in the Mapping Command window, select HL Qualification, and select AddChild HL Qualification. The HLQualification dialog box appears.
2. Select the HL level code from the drop-down list.

To insert an HL loop level as a peer (sibling) of the current HL loop level:

1. Right-click the HLLevel Command in the Mapping Command window, select HL Qualification, and select AddPeer HL Qualification. The HL Qualification dialog box appears.
2. Select the HL level code from the drop-down list. With each HL loop level created, the HL loop is copied to its hierarchical location (as a child or peer) with an HLLevel command and displayed in the command window as if the hierarchy were explicitly defined in the standard.

Chapter 12. Mapping binary data

The ASC-X12 Specifications/Technical Information transaction set (841) defines a way for trading partners to exchange technical information the same way they exchange EDI transactions. This technical information, which can be graphic, image, or audio, can contain binary data. The binary data can assume any value in the range X'00' through X'FF'.

In the syntax of X12 transaction sets, data elements that are separated by delimiters are combined into a segment that is identified by a segment ID and terminated with a segment delimiter. The binary data introduced by the 841 transaction set causes problems for this syntax because the binary data may contain a value that matches a segment delimiter. Translators and networks that support the 841 transaction set must have a way to identify binary data and determine its length so that it does not interfere with parsing the rest of the envelope. Special care must be taken if you want to send and receive files between WebSphere Partner Gateway and translators on other platforms. Not all operating systems support the record types z/OS uses.

BIN segment ID

Binary data is identified with a BIN segment ID, which notifies the parser that data following the segment ID is binary. Although the parser must always treat the BIN segment as if it contained binary data, the segment can contain normal text. The first data element of the BIN segment contains the length of the binary data so that the parser knows the amount of data to pass without interference. The first character after the binary data should be BIN segment terminator. Any other value is a syntax error that ends parsing for the envelope.

The BIN segment ID triggers the special binary processing. Although the 841 transaction set is the only one that uses the BIN segment, binary processing is not limited to the X12 standard. In addition, WebSphere Partner Gateway applies this special processing to all envelope types.

Length of the BIN segment

The value in the length data element of the BIN segment can be up to 15 characters long, which means the maximum length of a BIN segment is 999,999,999,999,999 bytes (fifteen 9s). However, the maximum size that WebSphere Partner Gateway supports is 2,147,483,647 bytes, which is the maximum signed integer value that a fullword can hold. The binary segment is a repeating segment with an unlimited number of repetitions, therefore it is unlimited for EDI.

Data transformation for binary data

This section covers mapping binary data for data transformation to and from an application field.

Mapping the binary segment

The BIN segment, as mentioned earlier, can have a rather impressive length. A new segment BDS has been introduced to transmit binary data in X12V4R4, transaction 102 (Associated Data).

85

BIN and BDS segments

The binary data is identified with a BIN or BDS segment ID, which notifies the parser that data following the segment ID is binary. Although the parser must always treat the BIN and BDS segments as if it contained binary data, the segment can contain normal text.

The first data element of the BIN segment contains the length of the binary data so that the parser knows the amount of data to pass without interference.

The second data element of the BDS segment contains the length of the binary data so that the parser knows the amount of data to pass without interference.

The first character after the binary data should be BIN or BDS segment terminator. Any other value is a syntax error that ends parsing for the envelope.

The BIN and BDS segment IDs triggers the special binary processing.

Although using a file name is the primary way of providing data for a binary Segment, with data transformation maps you provide the data by simply passing it to WebSphere Partner Gateway in application fields, the way all other application data is passed. The binary data itself must be defined in the Data Format with a data type of BN. The application may provide the length of the binary data If the length of the binary data is not provided by the application the length will be the length of the application data field defining the binary data.

However, WebSphere Partner Gateway has a maximum length of 32767 bytes for application fields, which is significantly less than the 999,999,999,999,999 defined by the standard or the 2,147,483,547 that WebSphere Partner Gateway allows. WebSphere Partner Gateway knows that it is dealing with binary data, and that some special processing is needed to put it into a BIN or BDS segment. The easiest way to pass this binary data to WebSphere Partner Gateway is with a data format such as:

BINARY_LOOP1 999 Repeating loop

BINARY_STRUCTURE1 Record

LENGTH N0 15 15-byte length field

BINARY_DATA_LOOP 99 Repeating loop

BINARY_DATA1 Record

BINARY_FIELD BN 32767 Binary data field

Both BINARY_STRUCTURE1 and BINARY_DATA1 are passed separately records.

Now the application provides WebSphere Partner Gateway with a BINARY_STRUCTURE1 and as many BINARY_DATA1 records as necessary to satisfy the length specified in the LENGTH field. WebSphere Partner Gateway builds one BIN or BDS segment for each occurrence of BINARY_STRUCTURE1.

The BIN and BDS segment has a length value as specified by the LENGTH field and contains data from the BINARY_DATA1 records.

EFI segment

For the 841 transaction set, an Electronic Format Identification (EFI) segment precedes a group of repeating BIN segments. The EFI segment provides information about the binary data file.

WebSphere Partner Gateway provides special processing for only the following data elements in the EFI segment:
- File name
- Block type
- Record length
- Block length

File name: The EDI standard essentially defines file name as free format with a maximum length of 64 characters. Its format depends on the computer operating system being used. Accordingly, WebSphere Partner Gateway treats this element as a fully qualified data set name, including the owner ID. A file name value with a length greater than 44 characters (the maximum length of a data set name) will be ignored.

Block type: The standard defines block type as free format with a maximum length of 4 characters. Its value indicates the organization of data in the BIN segments. Examples are fixed length, variable length, and spanned.

However, the definition does not provide any codes to represent the organizations, such as F for fixed and V for variable. In the absence of standard codes, WebSphere Partner Gateway interprets the block type as a string of characters that can have the following values:

A: ASA printer control characters

B: Block

F: Fixed

M: Machine code printer control characters

S: Spanned

U: Undefined

V: Variable

Chapter 13. Mapping case study

Your company's business document processing for the trading partner agreement with Company C are as follows.

1. Company C will send a Multiple payment order message (UN/EDIFACT 99B PAYMUL). Your company (Company A) will need to respond with a function acknowledgment message CONTRL.

2. You will need to create the Data Interchange Services client mapping for the Multiple payment order message (UN/EDIFACT 99B PAYMUL). The business analyst has already received the standard implementation guide and has created the mapping worksheet below.

Mapping Worksheet: Commercial Payments (UN/EDIFACT PAYMUL) Inbound

Header Record

Description	Data Type	Length	Group source	Source	Logic
3. Record ID	CH	2		Forced to H F in ADF	
4. Application ID	CH	6		Forced to REMFAC Increment accumulator T0	
5. Code identifier	CH	1		Forced to **A**	
6. Transmitting bank account number	CH	23	6	Forced to AFII - C078/3194	If FII/3035=OR
7. Transmission date	CH	8	0	DTM - C507/2379	If C507/2005 = 137
8. Transmission time	CH	6		ZERO	
9. No order remise	CH	10		ZERO	
	CH	1		Not used	
	CH	11			
10. Currency	CH	3	5	MOA - C516/6345	If MOA - C516/5025=9
11. Reference Number	CH	16	4	RFF - C506/1154	If RFF - C506/1153 = AEK
12. Reserved	CH	329			

Supplier Record

Description	Data Type	Length	Group source	Source	Logic
13. Record ID	CH	2		Forced with FO in ADF	

Description	Data Type	Length	Group source	Source	Logic
14. SIRET number of the supplier	CH	14	13	NAD - C082/3039	NAD/3035=PE If exists else use NAD/3035=BE
15. VAT Code of the supplier	CH	20		Not filled	
16. Supplier number account number	CH	32	13	NAD-C080/3036 (truncated to 32 char.)	NAD/3035=PE If exists else use NAD/3035=BE
17. Address line 1	CH	32	13	NAD-C080/3036 (truncated to 32 char.)	NAD/3035=PE If exists else use NAD/3035=BE
18. Address line 2	CH	32	13	NAD-C059/3042 occur. 1 (truncated to 32 char.)	NAD/3035=PE If exists else use NAD/3035=BE
19. Address line 3	CH	32	13	NAD-C059/3042 occur. 2 (truncated to 32 char.)	NAD/3035=PE If exists else use NAD/3035=BE
20. Address line 4	CH	32	13	NAD-C059/3042 occur. 3 (truncated to 32 char.)	NAD/3035=PE If exists else use NAD/3035=BE
21. Address line 5	CH	32	13	NAD-C059/3042 occur. 4 (truncated to 32 char.)	NAD/3035=PE If exists else use NAD/3035=BE
22. Address line line 6 = 23. Postal code 24. City	CH CH	9 23	13	NAD-3251(Postal code) + NAD-3164(City)	NAD/3035=PE If exists else use NAD/3035=BE
25. Reserved	CH	23			
26. Method of payment	CH	3		Forced with COM	
27. Prepayment prohibited	CH	1		Forced with N	

Description	Data Type	Length	Group source	Source	Logic
28. Action for payment	CH	1	11	'I' 'P' else 'T'	NAD/3035=PE or BE If PAI C534/4461 = Z7 and INP-C522/4401=ZZB If PAI C534/4461 = Z7 and INP-C522/4401=ZZA or PAI/4461 = Z7 seul (<-> PAI/4461 = Z8)
29. Reserved	CH	12			
30. Reserved	CH	6			
31. Reserved	CH	32			
32. Country code	CH	3	13	NAD-3207	NAD/3035=PE If exists else use NAD/3035=BE
33. Telephone number	CH	10	13	COM-C076/3148	NAD/3035=PE If exists else use NAD/3035=BE and If COM/3155=TE
34. Can proposal be sent by fax?	CH	1	13	'Y' Else 'N'	NAD/3035=PE If exists else use NAD/3035=BE and If COM/3155=FX
35. Fax number	CH	10	13	COM-C076/3148	NAD/3035=PE If exists else use NAD/3035=BE and If COM/3155=FX
36. Can proposal be sent e-mail?	CH	1	13	'Y'	NAD/3035=PE If exists else use NAD/3035=BE and If COM/3155=EM
37. E-mail address	CH	50	13	COM-C076/3148 Substring "(at)" must be replaced by @	NAD/3035=PE If exists else use NAD/3035=BE and If COM/3155=EM
38. Beneficiary account number	CH	34	12	C078/3194	If FII/3035=BF

Description	Data Type	Length	Group source	Source	Logic
39. Beneficiary bank name	CH	11	12	C078/3433	If FII/3035=BF (check if needed for phase 1. Not managed for VMB in phase 1)
40. Bank's country code	CH	3	12	C078/3207	If FII/3035=BF (check if needed for phase 1. Not managed for VMB in phase 1)
41. Beneficiary contact	CH	35	12	CTA-C056/3412	NAD/3035=PE If exists else use NAD/3035=BE If CTA/3139 = IC

Beneficiary Record (Non-financial)

Description	Data Type	Length	Group source	Source	Logic
42. Record ID	CH	2		Forced with BE in ADF	If NAD/3035=PE exists Else use NAD/3035=BE
43. Beneficiary Identification	CH	35	13	NAD - C082/3039	If NAD/3035=PE exists Else use NAD/3035=BE
44. Address line 1	CH	32	13	NAD-C080/3036 (truncated to 32 char.)	If NAD/3035=PE exists Else use NAD/3035=BE
45. Address line 2	CH	32	13	NAD-C059/3042 occur. 1 (truncated to 32 char.)	If NAD/3035=PE exists Else use NAD/3035=BE
46. Address line 3	CH	32	13	NAD-C059/3042 occur. 2 (truncated to 32 char.)	If NAD/3035=PE exists Else use NAD/3035=BE
47. Address line 4	CH	32	13	NAD-C059/3042 occur. 3 (truncated to 32 char.)	If NAD/3035=PE exists Else use NAD/3035=BE
48. Address line 5	CH	32	13	NAD-C059/3042 occur. 4 (truncated to 32 char.)	If NAD/3035=PE exists Else use NAD/3035=BE

Description	Data Type	Length	Group source	Source	Logic
49. Address line 6 = 50. Postal code 51. City	CH CH	9 23	13	NAD-3251(Postal code) + NAD- NAD-3251(Postal code) + NAD-3164(City)	NAD/3035=PE exists Else use NAD/3035=BE
52. Country code	CH	3	13	NAD-3207	If NAD/3035=PE exists Else use NAD/3035=BE
53. Employee ID	CH	35	13	CTA-C056/3412	If NAD/3035=OY PE exists à NAD/3035 = BE and 3139 = IC
54. Telephone number	CH	20	13	COM-C076/3148	If NAD/3035=PE exists Else use NAD/3035=BE and If COM/3155=TE
55. Fax number	CH	20	13	COM-C076/3148	If NAD/3035=PE exists Else use NAD/3035 = BE and If COM/3155=FX
56. Email address	CH	50	13	COM-C076/3148 Substring "(at)" must be replaced by @	If NAD/3035=PE exists Else use NAD/3035 = BE and If COM/3155=EM

Client Transfer Record

Description	Data Type	Length	Group source	Source	Logic
57. Record ID	CH	2		Forced with 'DO' in ADF	If NAD/3035=OY exists
58. Client Identification	CH	35	13	NAD - C082/3039	If NAD/3035=OY exists
59. Address line 1 (name of order party)	CH	32	13	NAD-C080/3036 (truncated to 32 char.)	If NAD/3035=OY exists
60. Address line 2	CH	32	13	NAD-C059/3042 occur. 1 (truncated to 32 char.)	If NAD/3035=OY exists
61. Address line 3	CH	32	13	NAD-C059/3042 occur. 2 (truncated to 32 char.)	If NAD/3035=OY exists

Description	Data Type	Length	Group source	Source	Logic
62. Address line 4	CH	32	13	NAD-C059/3042 occur. 3 (truncated to 32 char.)	If NAD/3035=OY exists
63. Address line 5	CH	32	13	NAD-C059/3042 occur. 4 (truncated to 32 char.)	If NAD/3035=OY exists
64. Address line 6 = 65. Postal code 66. City	CH CH	9 23	13	NAD-3251(Postal code) + NAD-3164(City)	NAD/3035=OY exists
67. Country code	CH	3	13	NAD-3207	If NAD/3035=OY exists
68. Correspondent name	CH	35	13	CTA-C056/3412	If NAD/3035=OY exists and 3139 = IC
69. Telephone number	CH	20	13	COM-C076/3148	If NAD/3035=OY and If COM/3155=TE
70. Fax number	CH	20	13	COM-C076/3148	If NAD/3035=OY and If COM/3155=FX
71. Email address	CH	50	13	COM-C076/3148 Substring "(at)" must be replaced by @	If NAD/3035=OY and If COM/3155=EM

Expiry Record

Description	Data Type	Length	Group source	Source	Logic
Record ID	CH	2		Forced with EC in ADF	
72. Client reference number	CH	10		Informed automatically by VMB	
73. Due date	CH	8	4	(Same as what was put in the FO record) 'I' 'P' Else 'T'	C507/2005=203
74. Action for payment	CH	1	13		NAD/3035=PE or BE If PAI C534/4461 = Z7 and INP-C522/4401=ZZB If PAI C534/4461 = Z7 and INP-C522/4401=ZZA or PAI/4461 = Z7 only (<-> PAI/4461 = Z8)

Description	Data Type	Length	Group source	Source	Logic
75. Reserved	CH	12	13		
76. Reserved	CH	6	13		
77. Method of payment	CH	3	13	Forced with COM	
78. Bank account to be charged	CH	23	13	Not filled	
79. Bank account for commissions	CH	23	13	Not filled	
80. Bank account to be credited	CH	23	13	Not filled	
81. Currency	CH	3	11	MOA - C516/6345	If C516/5025=9
82. Customer reference number	CH	12	11	RFF - C506/1154	RFF C506/1153=CR
83. Payment reference number	CH	12	11	RFF - C506/1154	RFF C506/1153=PQ

Branch Record

Description	Data Type	Length	Group source	Source	Logic
84. Record ID	CH	2	17	Forced with FI in ADF	
85. Client Identification	CH	35	17	NAD - C082/3039	If NAD/3035=BY and not SG13/3035=OY
86. Address line 1 (name of order party)	CH	32	17	NAD-C080/3036 (truncated to 32 char.)	If NAD/3035=BY and not SG13/3035=OY
87. Address line 2	CH	32	17	NAD-C059/3042 occur. 1 (truncated to 32 char.)	If NAD/3035=BY and not SG13/3035=OY
88. Address line 3	CH	32	17	NAD-C059/3042 occur. 2 (truncated to 32 char.)	If NAD/3035=BY and not SG13/3035=OY
89. Address line 4	CH	32	17	NAD-C059/3042 occur. 3 (truncated to 32 char.)	If NAD/3035=BY and not SG13/3035=OY
90. Address line 5	CH	32	17	NAD-C059/3042 occur. 4 (truncated to 32 char.)	If NAD/3035=BY and not SG13/3035=OY
91. Address line 6 = 92. Postal code 93. City	CH	9 23	17	NAD-3251(Postal code) + NAD-3164(City)	NAD/3035=BY exists and not SG13/3035 = OY

Description	Data Type	Length	Group source	Source	Logic
94. Country code	CH	3	17	NAD-3207	If NAD/3035=BY and not SG13/3035=OY
95. Telephone number	CH	20	17	Not filled	
96. Fax number	CH	20	17	Not filled	
97. Email address	CH	50	19	FTX-C108/4440 (1 per occurrence) and Substring "(at)" must be replaced by @	If NAD/3035=BY exists and If FTX-4451=CTA and FTX-C107/4441=EM
98. Correspondent Name	CH	35	17	RFF-C506/1154	If NAD/3035=BY exists and If RFF-C506/1153=CTA

Invoice Record

Description	Data Type	Length	Group source	Source	Logic
99. Record ID	CH	2		Forced with FA in ADF	
100. Reserved	CH	2	17		
101. Customer Invoice Number	CH	20	17	RFF – C506/1154 (truncated to 20 char.) else blank	If C506/1153 = ALK
102. Supplier Invoice Number	CH	20	17	DOC – C503/1004 (truncated to 20 char.) If it isn't the customer invoice number) Else blank	
103. Currency	CH	3	17	MOA C516/6345	
104. Invoice amount	CH	15	17	MOA C516/5004	If C516/5025 = 12
105. Sign	CH	1	17	Forced with + If DOC-C002/1001= 380 Forced with - If DOC-C002/1001= 381	
106. Invoice date	CH	8	17	DTM – C507/2380	If DTM/2005 =137
107. Acknowledgment Reference Number	CH	20	17	Not filled	

Description	Data Type	Length	Group source	Source	Logic
108. Invoice title	CH	32	17	DOC-C002/1000	
109. Branch reference number	CH	35	17	RFF – C506/1154	If C506/1153 =AOK (defined by the customer)

Trailer Record

Field name	Description	Data Type	Length	Source	Logic
110. TOCENR	Record ID	CH	2	Forced with T	
111. TOTOTMT	Total amount	N2	15	Absolute value of the sum of all invoices = sum of MOA- C516/5004 (group 17)	If C516/5025 = 128 then MOA-C516/5004 (group 23) must be equal to absolute value of invoices (whatever way)
112. TONBR	Total number of records	N0	6	Number of lines written in output file (counting begin and end records)	If C270/6069 = 2 then CNT – C270/6066 must be equal to the number of lines in the input message (number of LIN segments in group 4)
113. TOFIL1	Reserved	CH	15		

Sample EDI source data

UNB+UNOB:1+35226440200046:5+1234567890123:5+040428:1620+20040428162011′

UNH+1+PAYMUL:D:96A:UN′

BGM+452+1+9′

DTM+137:20040428:102′

LIN+1′DTM+203:20040629:102′

RFF+AEK:0404280420110001′

MOA+9:30501:EUR′

FII+OR+3006610972000104453O335+CMCIFRPP:25:5:+FR′

SEQ++1′

MOA+9:15001:EUR′

RFF+CR:DF-0000001199′

PAI+::Z7′FCA+14′

FII+BF+FR5130077023000000250245P32+SMCTFR2AVAL:25:5:STE MARSEILLAISE
+FR′

NAD+BE+77946774500013++LABORATOIRE BIOLOGIE MEDICALE+13 RUE

FARNERIE::+VALENCE++26001+FR′

CTA+IC+:.′INP+YC7:+1:ZZA′PRC+8′

DOC+381:::1+1′MOA+12:999:EUR′

DTM+137:20040430:102′RFF+ALK:1′

RFF+AOK:TST_CAMP′RFF+CTA:.::′

NAD+BY+CZ++CROUZET AUTOMATISMES+2 RUE DOCTEUR ABEL:BP
59:+VALENCE CEDEX 09++26902+FR′

AJT+999′MOA+999:0′FTX+COM++EM::+THIERRY.LEBRETON(AT)FR.NON.
SCHNEIDER-ELECTRIC.COM+FRA′

DOC+380:::2+2′

MOA+12:1000:EUR′

DTM+137:20040430:102′\

RFF+ALK:2′RFF+AOK:TST_CAMP′

RFF+CTA:.::′NAD+BY+CZ++CROUZET AUTOMATISMES+2 RUE DOCTEUR
ABEL:BP 59:+VALENCE CEDEX 09++26902+FR′

AJT+999′

MOA+999:0′

FTX+COM++EM::+THIERRY.LEBRETON(AT)FR.NON.SCHNEIDER-
ELECTRIC.COM+FRA′

DOC+380:::1+1′

MOA+12:15000:EUR′

DTM+137:20040430:102′

RFF+ALK:1′RFF+AOK:TST_CAMP′

RFF+CTA:.::′

NAD+BY+SIRET F01++FILIALE 01 CAMPAGNE+ADR1 F01:ADR2 F01:+VILLE++01010+'AJT+999'

MOA+999:0'F

TX+COM++EM::+THIERRY.LEBRETON(AT)FR.NON.SCHNEIDER-ELECTRIC.COM+FRA'SEQ++2'

MOA+9:5500:EUR'

RFF+CR:DF-0000001200'

PAI+::Z7'FCA+14'

FII+BF+FR7613906001317060301400048+:25:5:CRCA LORIOL+FR'

NAD+BE+38482501400010++CGS PLASTIQUES SARL+ATELIER RELAIS N.4:Z-A LES

BLACHES:+LORIOL SUR DROME++26270+FR'

CTA+IC+:VCOM'

COM+TLEBRETON(AT)GNULINE.COM:EM'

INP+YC7:+1:ZZA'

PRC+8'DOC+380:::1+1'

MOA+12:2500:EUR'

DTM+137:20040430:102'

RFF+ALK:1'

RFF+AOK:TST_CAMP'RFF+CTA:VCOM::'

NAD+BY+CZ++CROUZET AUTOMATISMES+2 RUE DOCTEUR ABEL:BP 59:+VALENCE CEDEX 09++26902+FR'AJT+999'

MOA+999:0'

FTX+COM++EM::+THIERRY.LEBRETON(AT)FR.NON.SCHNEIDER-ELECTRIC.COM+FRA'DOC+380:::2+2'

MOA+12:3000:EUR'

DTM+137:20040430:102'

RFF+ALK:2'RFF+AOK:TST_CAMP'

RFF+CTA:VCOM::'

NAD+BY+CZ++CROUZET AUTOMATISMES+2 RUE DOCTEUR ABEL:BP 59:+VALENCE CEDEX 09++26902+FR'AJT+999'

MOA+999:0′

FTX+COM++EM::+THIERRY.LEBRETON(AT)FR.NON.SCHNEIDER-
ELECTRIC.COM+FRA′SEQ++3′MOA+9:5000:EUR′

RFF+CR:DF-0000001201′

PAI+::Z7′FCA+14′

FII+BF+FR7611978000010119410301077+:25:5:FACTOCIC PARIS+FR′

NAD+BE+217++FACTOCIC PC/GESPAC LE CRES+TOUR FACTO::+PARIS LA
DEFENSE

CEDEX++92988+FR′

CTA+IC+:FACTOCIC PC/GESPAC LE CRES′

COM+TLEBRETON(AT)GNULINE.COM:EM′INP+YC7:+1:ZZA′

PRC+8′

DOC+380:::1+1′

MOA+12:1000:EUR′

DTM+137:20040430:102′

RFF+ALK:1′RFF+AOK:TST_CAMP′

RFF+CTA:FACTOCIC PC/GESPAC LE CRES::′

NAD+BY+CZ++CROUZET AUTOMATISMES+2 RUE DOCTEUR ABEL:BP
59:+VALENCE CEDEX 09++26902+FR′

AJT+999′MOA+999:0′

FTX+COM++EM::+THIERRY.LEBRETON(AT)FR.NON.SCHNEIDER-
ELECTRIC.COM+FRA′

DOC+380:::2+2′

MOA+12:4500:EUR′

DTM+137:20040430:102′

RFF+ALK:2′RFF+AOK:TST_CAMP′

RFF+CTA:FACTOCIC PC/GESPAC LE CRES::′

NAD+BY+CZ++CROUZET AUTOMATISMES+2 RUE DOCTEUR ABEL:BP
59:+VALENCE CEDEX 09++26902+FR′

AJT+999′MOA+999:0′

FTX+COM++EM::+THIERRY.LEBRETON(AT)FR.NON.SCHNEIDER-
ELECTRIC.COM+FRA'

DOC+381:::3+3'

MOA+12:500:EUR'

DTM+137:20040430:102'

RFF+ALK:3'RFF+AOK:TST_CAMP'

RFF+CTA:FACTOCIC PC/GESPAC LE CRES::'

NAD+BY+CZ++CROUZET AUTOMATISMES+2 RUE DOCTEUR ABEL:BP
59:+VALENCE CEDEX 09++26902+FR'AJT+999'

MOA+999:0'

FTX+COM++EM::+THIERRY.LEBRETON(AT)FR.NON.SCHNEIDER-
ELECTRIC.COM+FRA'

SEQ++4'MOA+9:5000:EUR'

RFF+CR:DF-0000001202

'PAI+::Z7'FCA+14'

FII+BF+FR02100960010701078409S8A46+:25:5:LYONNAISE DE BANQ+FR'

NAD+PE+31184268600017++LES CHARPENTIERS DE BOURGOGNE+4, RUE
LAVOISIER- BP 99::+DIJON LONGVIC++21603+FR'CTA+IC+:.'

NAD+BE+DF-0000001202++LABORATOIRE BIOLOGIE MEDICALE+::++++'

COM+NON RENSEIGNE:EM'

INP+YC7:+1:ZZA'

PRC+8'DOC+380:::1+1'

MOA+12:10000:EUR'

DTM+137:20040430:102'

RFF+ALK:1'

RFF+AOK:TST_CAMP'

RFF+CTA:.::'

NAD+BY+SIRET F01++FILIALE 01 CAMPAGNE+ADR1 F01:ADR2
F01:+VILLE++01010+'

AJT+999'MOA+999:0'

FTX+COM++EM::+THIERRY.LEBRETON(AT)FR.NON.SCHNEIDER-
ELECTRIC.COM+FRA'

DOC+381:::2+2'MOA+12:5000:EUR'

DTM+137:20040430:102'

RFF+ALK:2'RFF+AOK:TST_CAMP'

RFF+CTA:.::'

NAD+BY+SIRET F01++FILIALE 01 CAMPAGNE+ADR1 F01:ADR2
F01:+VILLE++01010+'AJT+999'

MOA+999:0'

FTX+COM++EM::+THIERRY.LEBRETON(AT)FR.NON.SCHNEIDER-
ELECTRIC.COM+FRA'

CNT+2:1'

UNT+154+1'

UNZ+1+20040428162011'

Sample Record Oriented Data (Records are Truncated):

H REMFACA30066109720001044530335200404280000000000000000
EUR040428042FO77946774500013 LABORATOIRE BIOLOGIE MEDICALE
LABORATOIRE

EC 20040629P COM

FICZ CROUZET AUTOMATISMES 2 RUE DOCTE

FA 1 1 EUR000000000099900-200404301

FICZ CROUZET AUTOMATISMES 2 RUE DOCTE

FA 2 2 EUR000000000100000+200404302

FISIRET F01 FILIALE 01 CAMPAGNE ADR1 F01

FA 1 1 EUR000000001500000+200404301

FO38482501400010 CGS PLASTIQUES SARL CGS PLASTIQU

EC 20040629P COM

FICZ CROUZET AUTOMATISMES 2 RUE DOCTE

FA 1 1 EUR000000000250000+200404301

FICZ CROUZET AUTOMATISMES 2 RUE DOCTE

FA 2 2 EUR000000000300000+200404302

FO217 FACTOCIC PC/GESPAC LE CRES FACTOCIC PC/

EC 20040629P COM

FICZ CROUZET AUTOMATISMES 2 RUE DOCTE

FA 1 1 EUR000000000100000+200404301

FICZ CROUZET AUTOMATISMES 2 RUE DOCTE

FA 2 2 EUR000000000450000+200404302

FICZ CROUZET AUTOMATISMES 2 RUE DOCTE

FA 3 3 EUR000000000050000-200404303

FO3118426860001 LES CHARPENTIERS DE BOURGOGNE LES CHARPENT

BEDF-0000001202 LABORATOIRE BIOLOGIE MEDICALE

EC 20040629P COM

FISIRET F01 FILIALE 01 CAMPAGNE ADR1 F01

FA 1 1 EUR000000001000000+200404301

FISIRET F01 FILIALE 01 CAMPAGNE ADR1 F01

FA 2 2 EUR000000000500000-200404302

T 00000000434990000031

Sample mapping commands

The following sample mapping commands listed by reference number in the
mapping worksheet under the Description column or Field name column (Trailer
Record only).

Example #6:

6. Transmitting bank account number CH 23 6 FII - C078/3194 If FII/3035=OR

If (StrCompN (\Table 1\70 C FII Loop\70 C FII\1 M 3035\\, ″OR″, 2) = 0)

\HDR_REC\TRANS_BANK\ACCT\\ = \Table 1\70 C FII Loop\70 C FII\2 C
C078\1 C 3194\\

EndIf

Example #16:

16. Supplier number CH 32 13 NAD-C080/3036 (truncated to 32 char.)

NAD/3035=PE If exists else use NAD/3035=BE
1.   The NAD loop must be qualified by value:

*** Since this has an else and it is possible to get both NAD loops a Local Variable can be useful

Create a Local Variable SupplierNumber with scope Document Character and length 32

Qualify (StrComp (\Table 1\100 C NAD Loop\100 C NAD\1 M 3035\\, "PE") EQ 0)

Qualify (StrComp (\Table 1\100 C NAD Loop\100 C NAD\1 M 3035\\, "BE") EQ 0)

2. Under Qualifier "PE" under node NAD-C080/3036:

SupplierNumber = SubString (\Table 1\100 C NAD Loop\100 C NAD\4 C C080\1 M 3036\\, 1, 32)

\SUPPLIER_REC\SUPPLIER_NUM\\ = SupplierNumber

3. Under Qualifier "BE" under node NAD-C080/3036:

If (IsEmpty (SupplierNumber))

SupplierNumber = SubString (\Table 1\100 C NAD Loop\100 C NAD\4 C C080\1 M 3036\\, 1, 32)

\SUPPLIER_REC\SUPPLIER_NUM\\ = SupplierNumber

EndIf

Example #64

64. Address line 6 =

65. Postal code

66. City

NAD-3251(Postal code) + NAD-3164(City) NAD/3035=OY exists

1. Add Qualification by Value under NAD for "OY" value

Qualify (StrComp (\Table 1\100 C NAD Loop\100 C NAD\1 M 3035\\, "OY") EQ 0)

2. Under node NAD-3251:

\CLIENT_TRANS_REC\ADDR_6\\ = Concat (\Table 1\100 C NAD Loop\100 C NAD\8 C 3251\\,

\Table 1\100 C NAD Loop\100 C NAD\6 C 3164\\)

Example #71

71. Email address CH 50 13 COM-C076/3148 Substring "(at)" must be replaced by @ If

NAD/3035=OY and If COM/3155=EM

1. NAD Qualification for "OY" and under node COM /3155

*** Add Local Variable email scope document, character length 50.

*** Add Local Variable Position scope document, real initial value 0.

If (StrCompN (\Table 1\100 C NAD Loop\120 C COM\1 M C076\2 M 3155\\, "EM", 3) = 0)

Position = Find (\Table 1\100 C NAD Loop\120 C COM\1 M C076\1 M 3148\\, "(at)", 1)

If (Position > 0)

email = SubString (\Table 1\100 C NAD Loop\120 C COM\1 M C076\1 M 3148\\, 1, Position)

email = Concat (email, '@')

email = Concat (email, SubString (\Table 1\100 C NAD Loop\120 C COM\1 M C076\1 M 3148\\, Position+4))

Else

email = \Table 1\100 C NAD Loop\120 C COM\1 M C076\1 M 3148\\

EndIf

\Client_TRANS_REC\EMAIL\\ = email

EndIf

Example #112

112. TONBR Total number of records N0 6 Number of lines written in output file (counting begin and end records)

If C270/6069 = 2 then CNT - C270/6066 must be equal to the number of lines in the input message (number of LIN segments in group 4)

1. Count the number of LIN segments:

   *** Create a Local Variable LINCount Scope Document Integer with initial value 0

   Under the LIN Loop or Segment:

   LINCount = LINCount + 1

2. Under the node CNT – C270/6066 :

   If (\Table 1\860 C CNT\1 M C270\2 M 6066\\ != LINCount)

   Error (1, 5001, "Control Value and Line Item Count do not match")

   EndIf

Notices

This information was developed for products and services offered in the U.S.A.

IBM may not offer the products, services, or features discussed in this document in other countries. Consult your local IBM representative for information on the products and services currently available in your area. Any reference to an IBM product, program, or service is not intended to state or imply that only that IBM product, program, or service may be used. Any functionally equivalent product, program, or service that does not infringe any IBM intellectual property right may be used instead. However, it is the user's responsibility to evaluate and verify the operation of any non-IBM product, program, or service.

IBM may have patents or pending patent applications covering subject matter described in this document. The furnishing of this document does not grant you any license to these patents. You can send license inquiries, in writing, to:

IBM® Director of Licensing
IBM Corporation
North Castle Drive
Armonk, N.Y. 10504-1785
U.S.A.

For license inquiries regarding double-byte (DBCS) information, contact the IBM Intellectual Property Department in your country or send inquiries, in writing, to:

IBM World Trade Asia Corporation Licensing
2-31 Roppongi 3-chome, Minato-ku
Tokyo 106-0032, Japan.

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

Any references in this information to non-IBM Web sites are provided for convenience only and do not in any manner serve as an endorsement of those Web sites. The materials at those Web sites are not part of the materials for this IBM product and use of those Web sites is at your own risk.

IBM may use or distribute any of the information you supply in any way it believes appropriate without incurring any obligation to you.

Licensees of this program who wish to have information about it for the purpose of enabling: (i) the exchange of information between independently created programs and other programs (including this one) and (ii) the mutual use of the information which has been exchanged, should contact:

IBM Burlingame Laboratory Director
IBM Burlingame Laboratory
577 Airport Blvd., Suite 800
Burlingame, CA 94010
U.S.A

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.

COPYRIGHT LICENSE

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not

been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs.

Each copy or any portion of these sample programs or any derivative work, must include a copyright notice as follows:

Copyright (c) 1995-2007 International Business Machines Corporation and others All rights reserved.

If you are viewing this information softcopy, the photographs and color illustrations may not appear.

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.

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

Trademarks and service marks

The following terms are trademarks or registered trademarks of International Business Machines Corporation in the United States or other countries, or both:

IBM	DB2	IMS	MQIntegrator	Tivoli
the IBM logo	DB2 Universal Database	Informix	MVS	WebSphere
AIX		iSeries	OS/400	z/OS
CICS	Domino	Lotus	Passport Advantage	
CrossWorlds	IBMLink	Lotus Notes		
	i5/OS		SupportPac	

Microsoft, Windows, Windows NT, and the Windows logo are trademarks of Microsoft Corporation in the United States, other countries, or both.

MMX, Pentium, and ProShare are trademarks or registered trademarks of Intel Corporation in the United States, other countries, or both.

Solaris, Java, and all Java-based trademarks are trademarks of Sun Microsystems, Inc. in the United States, other countries, or both.

Linux is a trademark of Linus Torvalds in the United States, other countries, or both.

Other company, product or service names may be trademarks or service marks of others.

WebSphere Partner Gateway Enterprise and Advanced Editions includes software developed by the Eclipse Project (www.eclipse.org)

IBM ®

Printed in USA