SYBASE®

Reference Guide

ECMap™

Version 4.2

[Windows]

DOCUMENT ID: DC36332-01-0420-01

LAST REVISED: November 2004

Sybase, Inc., One Sybase Drive, Dublin, CA 94568.

Contents

About This Book

This document is a reference guide for ECMap™, a tool for building and understanding structured information messages.

Audience

While you certainly do not need to be a programmer to use ECMap, it is helpful to be familiar with certain technical concepts, such as the following:

- ECRTP™ (process engine)
- Electronic Data Interchange (EDI) and HIPAA concepts
- HIPAA transaction formats and usage

How to use this book

This guide presents general technical background information about specific topics that are relevant to understanding and using ECMap. It also provides a brief summary of ECMap features, with a particular emphasis on the newest additions to product functionality.

This guide describes how to use ECMap. It is organized into the following chapters:

- Chapter 1, “Informational Messages” describes the informational messages ECMap may encounter.
- Chapter 2, “Adapter Configuration Files” describes how to create the configuration files specified in ECMap’s Export Schema utility.
- Chapter 3, “Trading Partner and Log Database Formats” describes how ECMap’s trading partner information is stored in the trading partner database, and transaction and error logging are stored in a log database with their ODBC and non-ODBC versions.
- Chapter 4, “EDI Envelopes” describes the EDI envelope types.
- Chapter 5, “ASCII Character Chart” describes each of the ASCII characters that can be used in ECMap, such as decimal, binary and hexadecimal representations; description; abbreviation; and printable and non-printable characters.

Related documents

The following documents ship with ECMap:

- *ECMap New Features Guide*

- *ECMap Installation Guide*
- *Release Bulletin for ECMap*
- *ECMap Reference Guide*
- *ECMap User Guide*
- *ECMap Getting Started*

Additional documents are referred to in the ECMap documentation to supply you with specific information that supports this product:

- *ECRTP Reference Guide* to use the data transformation engine

Documentation that supports ECMap can be found on the Sybase Product Manuals web site. Go to Product Manuals at http://www.sybase.com/support/manuals, select ECMap from the drop-down list, and click Go!

Other sources of information

Use the Sybase Getting Started CD, the SyBooks Bookshelf CD, and the Sybase Product Manuals web site to learn more about your product:

- The Getting Started CD contains release bulletins and installation guides in PDF format, and may also contain other documents or updated information not included on the SyBooks Bookshelf CD. It is included with your software. To read or print documents on the Getting Started CD, you need Adobe Acrobat Reader, which you can download at no charge from the Adobe Web site using a link provided on the CD.
- The SyBooks Bookshelf CD is included with your software. It contains product manuals in a platform-independent bookshelf that contains fully searchable, HTML-based documentation.

  Some documentation is provided in PDF format, which you can access through the PDF directory on the SyBooks Bookshelf CD. To view the PDF files, you need Adobe Acrobat Reader.

  Refer to the *README.txt* file on the SyBooks Bookshelf CD for instructions on installing and starting SyBooks.
- The Sybase Product Manuals Web site is the online version of the SyBooks Bookshelf CD that you can access using a standard Web browser. In addition to product manuals, you will find links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

  To access the Product Manuals Web site, go to Product Manuals at http://www.sybase.com/support/manuals/.

Sybase certifications on the Web

Technical documentation at the Sybase Web site is updated frequently.

❖ **Finding the latest information on product certifications**

1. Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.
2. Select Products from the navigation bar on the left.
3. Select a product name from the product list and click Go.
4. Select the Certification Report filter, specify a time frame, and click Go.
5. Click a Certification Report title to display the report.

❖ **Creating a personalized view of the Sybase Web site (including support pages)**

Set up a MySybase profile. MySybase is a free service that allows you to create a personalized view of Sybase Web pages.

1. Point your Web browser to Technical Documents at http://www.sybase.com/support/techdocs/.
2. Click MySybase and create a MySybase profile.

Sybase EBFs and software maintenance

❖ **Finding the latest information on EBFs and software maintenance**

1. Point your Web browser to the Sybase Support Page at http://www.sybase.com/support.
2. Select EBFs/Maintenance. If prompted, enter your MySybase user name and password.
3. Select a product.
4. Specify a time frame and click Go. A list of EBF/Maintenance releases is displayed.

   Padlock icons indicate that you do not have download authorization for certain EBF/Maintenance releases because you are not registered as a Technical Support Contact. If you have not registered, but have valid information provided by your Sybase representative or through your support contract, click Edit Roles to add the “Technical Support Contact” role to your MySybase profile.
5. Click the Info icon to display the EBF/Maintenance report, or click the product description to download the software.

Typographic conventions

This documentation uses the following typographic conventions:

Item	Description
`Code`	SQL and program code displays in a mono-spaced (fixed-width) font.
`User entry`	Text entered by the user is shown in bold serif type.
emphasis	Emphasized words are shown in italic.
file names	File names are shown in italic.
database objects	Names of database objects, such as tables and procedures, are shown in sans serif type in print, and in italic online.
sybase\bin	A backward slash ("\") indicates cross-platform directory information. A forward slash ("/") applies to UNIX-specific information. Directory names appearing in text display in lowercase unless the system is case sensitive.
File > Save	Menu names and menu items are displayed in plain text. The angle bracket indicates how to navigate menu selections, such as from the File menu to the Save option.
`parse|put|get`	The vertical (pipe) bar indicates • Options available within code • Delimiter within message examples
segment	Bold text indicates a glossary term.

If you need help

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you cannot resolve a problem using the manuals or online help, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.

CHAPTER 1

Informational Messages

About this chapter

This chapter describes the informational messages ECMap may encounter.

Topics

This chapter contains the following topics:

Overview

Message format

The standard format for an ECMap informational message is the following:

```
(Message Number) Severity Level: Message
```

Example

```
(7015) ERROR: Output Files Reset Due to Transaction
Errors
```

Message number

The message number is a unique number assigned to an informational message. Messages are generally grouped by category of message, with a specific range of message numbers falling within a given category. See "System messages" on page 3.

Severity level

The following table describes an informational message's level of severity:

Message	Description of level of severity
OK	Informational message to the user that requires no action.
TRACE	Informational message to the user that requires no action.
DEBUG	Informational message to the user that requires no action
WARNING	Caution note to the user. Sybase recommends that you correct all WARNING conditions.
ERROR	A problem exists that you must correct. ERROR messages are frequently the result of a mistake in your setup, but they can also be simple data entry errors.
FATAL	A problem causes the program to terminate. FATAL errors can be the result of missing executables, not enough PC memory, internal program errors, or missing databases.

Message text

The message text is a brief explanation of message. Informational messages are found in the transaction log – *trlog* for those using an ODBC log and *translog.in* and *translog.out* for those using a non-ODBC log.

System messages

With two exceptions, system messages are the result of missing executables, inadequate PC memory, or internal program errors.

Error message exceptions

The two exceptions are the messages related to the User ABORT and STOP_RUN commands. The messages for these commands indicate that the user terminated the map run.

Error messages described as “Internal program error” are present to catch program logic errors and do not display. If they do display, note the message number and any associated error number, and call Sybase support for assistance immediately.

ECMap relies on the dynamic allocation and deallocation of memory. If memory allocation errors occur, call Sybase support for help in analyzing your transaction set databases and your PC memory limits.

Memory error work arounds

Two work arounds could allow work to continue when memory errors occur:

- If a Memory Allocation Error message displays during map run or compile, examine the data dictionary and rules databases, as well as the map flow database used with the current transaction. If particular rules, map flow levels, records, fields or memory variables are not needed for the current transaction, reduce the amount of memory required to process the transaction by creating a smaller rules, map flow or data dictionary database.
- If a Memory Allocation Error occurs when an option other than map run or compile was executed, restart ECMap and try running your map again.

Table 1-1 lists the format, cause, and solution (if applicable) for the ECMap system messages in numeric order by message number.

Table 1-1: ECMap system messages

Message	Cause	Solution
`(0003) FATAL: undefined database type- <internal program code>`	Internal program error.	Call support for assistance.
`(0021) OK: executed STOP_RUN command`	Information Only. User STOP_RUN rule was executed.	A command placed in the map by a user. Find where the command is located in the map and remove it if the command is not desired.

Message	Cause	Solution
(0100/<ERRNO-NO>) FATAL: call to program module- <program name>	An attempt to spawn “PROGRAM” failed. The ERRNO-NO is the error code from the ‘C’ “errno” table. The most likely reasons are “errno” number 2, which means that the executable file does could not be found on disk, or “errno” number 12, which means that there is not enough memory available to load the called program.	For “errno” value of 2, the corrective action is to do a DIR on the directory where the MAPPER executable are stored. WWIXMAP – The default installation directory for executables is “MAPEXE”. ECMAP – The default installation directory for executables is *Program Files\Sybase\ECMap*. If the file is not on the disk, reinstall the WWIX executable files using the installation disks. If the executable does exist on disk but was not found, then make sure that the WWIX executable directory is included in the DOS PATH environment variable. *For “errno” value of 12, call support for assistance.*
(0102) FATAL: program call- incorrect parameters	One or more wrong parameters was called.	Recheck the list of valid parameters and apply those parameters to the command line.
(0103) FATAL: parameter<parameter switch> exceeds Max Length number <maximum length>	You have too many parameters for your map.	Reduce the number of parameters.
(0105) FATAL: invalid or undefined input parameter(s)	Internal program errors. A parameter has been called or accessed that has not been defined, or is not defined correctly.	Call support for assistance.
(0200) FATAL: memory allocation	MAPPER was unable to allocate enough memory to execute a critical function.	Call support for assistance.

Message	Cause	Solution
(0202) FATAL: memory allocation; loading database- <database>	Indicates that there was not enough memory to create a temporary memory table from a database. If this message occurs while performing the Define Map Flow function, then the size of your Map Flow database has exceeded the size that can be loaded into memory. Examine the Map Flow to determine if all of the defined levels are required.	Call support for assistance.
(0208) FATAL: memory allocation; loading < > for Map: <filename>	Insufficient memory available to load map.	Call support for assistance.
(0209) ERROR: USING EDIFACT MAP with NON-EDIFACT ENVELOPE	Occurs when envelope information does not contain EDIFACT standards.	Call support for assistance.
(0210) ERROR: USING NON-EDIFACT MAP with EDIFACT ENVELOPE	Occurs when envelope information contains EDIFACT standards or a non-EDIFACT map.	Call support for assistance.
(0211) FATAL: memory allocation loading map	An internal program error. The length of the text keyed in for a rule operand by the user is greater than the space that has been allocated to store the text rule operand.	Call support for assistance.
(0212) FATAL: Trading Partner <trading partner> Has No Mailbox	No directory has been defined in the Trade Agreement record	Define the directory.
(0213) FATAL: Can't Set Transaction level		
(0214) FATAL: Can't Retrieve WIXSET Record	The wixset database is damaged or the wixset database has no record inside.	Check the wixset database
(0215) FATAL: Attempt to run < > with < > MAP	Attempt to run "inbound" with an "outbound" map.	

On map selection and directories

Some of the ECMap informational messages are related to map directories, application file directories, and ECMap environment variables.

ECMap provides great flexibility in deciding where to place directories and files on the disk. But with this flexibility comes the responsibility for accurately describing these locations to the mapping program.

- ECMap depends on the user to correctly define map directory sets for each inbound and outbound EDI transaction. This is done through setting project map directories on the Project/Map window.
- Each of the map directory sets defined is identified by a unique map name that is assigned on the Map window.
- ECMap also relies on the user to have entered valid directory locations for all application files using the Directories (Mailboxes), Files/Databases, and Records/Tables windows.

Table 1-2 lists the format, cause, and solution (if applicable) for the ECMap messages related to map selection and directories, by message number.

Table 1-2: ECMap messages related to map selection and directories

Message	Cause	Solution
`(1000) ERROR: can't find map- <mapname>`	The map name was not found in the database table of defined map names, MTABLE.	If this error displays as a result of direct window entry of a map name or at ECMap startup ensure that the map name is correctly spelled. If this error occurs when running an inbound or outbound map, examine the trade agreement records that are associated with the transaction set for the trading partner processed. One of the map names that has been specified in the trade agreement records is not a valid map name (i.e. is not contained in MTABLE). WWIXMAP – Valid map names can be viewed from the trade agreement record window by moving the cursor to the map name field and pressing <F2> (option to display valid map names).
`(1006) ERROR: Map name or directory database is incomplete`	The currently selected map name has not been defined completely.	Examine all of the directory entries for this map name. One of the directories has been left blank. Enter a directory name into this field.

Message	Cause	Solution
(1010) ERROR:[WWIXMAP] = environment variable is not SET	The environment variable must be set to an existing directory prior to invoking the MAPPER program. The MAPPER program checks this directory for control databases at start up. If the control databases do not exist they are automatically created. WWIXMAP default directory is MAPEXE	Control databases, such as the database table of defined map names, MTABLE, are located under the ECMap environment directory. Changing the value of ECMap causes a new MTABLE to be created under the new ECMap directory. This new MTABLE does not contain previously defined map names.
(1012) ERROR: [WWIXMAP] = environment variable invalid value (1013) ERROR: [WWIXMAP] = environment variable Disk Drive not Specified (1014) ERROR: [WWIXMAP] = environment variable Must have \"\\\" after Drive:	Messages 1012 – 1014 are displayed when the environment variable is either not set or set to a name that is not a valid directory name.	Check the setting and correct it. The values of environment variables can be checked at the DOS prompt by keying "SET" and pressing the ENTER key. At the DOS prompt, you can assign a new value to the environment variable by keying: SET [WWIXMAP] = "FULL PATH DIRECTORY NAME". For example, the DOS command, "SET [WWIXMAP]= C:\MAPDATA", sets the environment directory to the directory *C:\MAPDATA*. **Note** Place the SET command that assigns a full path directory name in the DOS *autoexec.bat* file.
(1015) ERROR: [WWIXMAP] = environment Must Be a Sub-Directory	ECMap cannot point to the root directory.	Modify environment variables to include drive and directory.
(1016) ERROR: Invalid Directory	ECMap was unable to locate the designated directory.	Modify variable to include drive and directory.
(1017) ERROR: can't get Current Working Directory	System cannot get current working directory or pathname can be longer than 260 characters (default for Windows).	Verify that all path directories should be less than 260 characters.
(1018) ERROR: putenv < > failed	Creating a new environment variable, modifying, or removing existing environment variables failed.	System error. Call Technical Support.

Message	Cause	Solution
(1020) FATAL: can't create - <directory name>	Cause 1: The MAPPER program was unable to create the designated directory. This message displays because either the directory name is invalid or the DOS limit has been reached for the number of directories on the PC.	Solution 1: Verify that the directory name is a valid directory for the PC. For example if the PC in use does not have a D drive, then attempts to create a directory on the D drive fails.
	Cause 2: This message can be the result of keying in an incorrect directory name.	Solution 2: Key in a correct directory name in the window field.
	Cause 3: Directory Names are stored in the User Files and in the map name Definition databases. When these directory names are entered they are verified as correct. However, changing the Configuration of a PC or copying data files between PC's could result in invalid directories in use.	Solution 3: Verify that all file directories are valid. When the current map name is highlighted, verify that the Map directories for this map name are correct
	Cause 4: If the directory name is valid, then the DOS limit for the number of directories on your hard disk could have been reached. For most hard disks, the total number of directories allowed depends on how the hard disk was formatted. Double sided 40 track diskettes can have a total of 112 directories, and high capacity double sided 80 track diskettes can have a total of 224 directories.	Solution 4: Delete a directory that is unrelated to the MAPPER to free up a directory entry space and execute the program again.
(1021) FATAL: Connection to <DSN> failed.	ODBC could not establish a connection to the database specified in the filename.	Check that the DSN specified is correct. Connection between ODBC drivers and the actual database has been broken. For example, the database server is down, and communication is not taking place.
(1022) FATAL: Bind Parameter Failed.	The internal record field definitions and the table structures do not match.	Re-import the table definitions from the database. Re-export and create a new table in the database based on the new table definition in ECMap.

Message	Cause	Solution
(1023) FATAL: Select Failed.	Cause 1: Select statement could not execute because maximum number of connections (cursors) has been exceeded. Cause 2: Select statement could not execute because database connection has been terminated.	Solution 1: Examine tables that establish the maximum number of cursors. Check Select Statement for correct syntax. Solution 2: Ensure that database has not gone inactive at the current location.
(1024) FATAL: Fetch without Select	Fetch was issued against a particular record type that had not been selected.	Examine rules to ensure that table has been "selected" prior to issuing the "fetch" command.
(1025) FATAL: Insert <record name> Failed	Cause 1: ODBC code failed to insert a record into the database because maximum number of connections (cursors) has been exceeded. Cause 2: ODBC code failed to insert a record into the database because database connection has been terminated.	Solution 1: Examine tables that establish the maximum number of cursors. Check Insert Statement for correct syntax. Solution 2: Ensure that database has not gone inactive at the current location.
(1026) FATAL: Update without Select.	ODBC could not establish a connection to the database specified in the filename.	Check that the DSN specified is correct.
(1027) FATAL: Update Failed.	Cause 1: Select statement could not execute because maximum number of connections (cursors) has been exceeded. Cause 2: Select statement could not execute because database connection has been terminated.	Solution 1: Examine tables that establish the maximum number of cursors. Check Update Statement for correct syntax. Solution 2: Ensure that database has not gone inactive at the current location.
(1028) FATAL: Get/SetCursorName Failed.		
(1029) FATAL: Invalid Cursor State	No ODBC connection or no cursor handler.	Call Support assistance.
(1030) FATAL: <ODBC> Error: <text explanation>	ODBC manager detected an error. Passed back to ECMap.	See the ODBC error messages in Section 15 – Microsoft Standard ODBC Error Messages.
(1031) ERROR: SQL RollBack Performed	A Bad Transaction has been processed. SQL statements that insert data into database are rolled back.	Informational Message accompanied by other error messages that detail the transaction errors that occurred.
(1032) OK: SQL RollBack Performed	A user-invoked SQL rollback has occurred.	Informational Message

Message	Cause	Solution
(1033) FATAL: Fetch Failed.	Record definition and database records do not match.	Ensure that the database definitions and the record definitions are synchronized.
(1034) FATAL: Procedure <Procedure Name> Failed	Cause 1: Procedure Call failed because of environment issue Cause 2: Procedure Call failed because of a bad parameter.	Solution 1: Examine procedure call in database to ensure that environment settings are correct Solution 2: Examine procedure call in database to ensure that a bad parameter was not passed to the program.
(1035) Fatal: Number of Fields in <record> Does Not Match Prior Map	Error occurs on Map Switching when using different maps for the same messages. The record definitions in one map do not match the Record Definitions in the initial map.	If different maps are used, ensure that they contain the same record definition for the same table.
(1036) TRACE: Database doesn't support a LOCK_TYPE.		
(1037) Fatal: Wixset Connection string not defined.	ODBC Trading Partner	Set up the company information.
(1038) Fatal: RUN_ID Table Empty		
(1039) Fatal: Cannot Obtain Mutex Lock		
(1040) Fatal: Try to write to stdin		
(1041) Fatal: Try to read from stdout		
(1042) FATAL: Failed to Write TRLOG – dumping log to < >		
(1043) FATAL: Commit Failed on < >		
(1050) ERROR: Data is too long, exceed the buffer length		
(1051) ERROR: Element name does not exist		
(1052) ERROR: Element name is too long		

Message	Cause	Solution
(1053) ERROR: The element name <element> does not match any record tag names		
(1054) ERROR: Failed to load attribute value		
(1055) ERROR: While reading data encountered an end of file		
(1056) ERROR: Tag length is 0		
(1057) ERROR: Attribute name is too long		
(1058) ERROR: Attribute name is missing		
(1059) ERROR: This is not a right position to get the attribute value		
(1060) ERROR: Missing ending quote when loading attribute value		
(1061) ERROR: Attribute value is too long		
(1062) ERROR: xml_input_convert() failed		
(1063) ERROR: Invalid switch		
(1064) ERROR: Invalid read operation from stdout		
(1065) ERROR: Invalid write operation to stdin		

Message	Cause	Solution
(1066) TRACE: Write XML start tag -- <start tag>		
(1067) TRACE: Write XML end tag -- <end tag>		
(1068) WARNING: Next XML Tag = <next tag>; Reading Record Tag = <current tag>	Tag in data does not match tag in Read Rule at that level or the following level	This is an informational message; you might not need to do anything. You can make sure the matching tag exists in the input data at this level or the following level.
(1069) WARNING: No more starting tag when reading record tag -- <current tag>	There are no more starting tags at the top of the loop in the input data and the end of the file has been reached.	Verify that the looping tag exists in the input data.
(1070) WARNING: The element name (<element>) does not exist in the map, skip to next	The element in the input data is not present in the map definition.	Verify that the input data is correct or element exists in the map.
(1071) WARNING: The attribute name (<attribute>) is not defined in the record, skip to next	The specified attribute in the input data is not present in the map definition.	Verify that the input data is correct or the specified attribute exists in the map.

On file I/O

Some of the messages in this section are the result of system errors from opening, reading, writing and closing files. Other errors result from users performing MAPPER program definitions for User Files, Map Directory Sets, Map I/O Rules or Map Flows.

Note Some of the messages in this section are described as internal program errors. Internal program error messages are not expected to display. If an internal error message occurs or if the suggested corrective action does not solve a problem, note the message number and any associated DBCIII error number and call Sybase support for assistance.

Table 1-3 lists the format, cause, and solution (if applicable) for the ECMap messages related to File I/O (including databases and indexes), by message number.

Table 1-3: ECMap messages related to File I/O

Message	Cause	Solution
(1101) FATAL: file <file name> has non-numeric < >		
(1102) FATAL: file <file name> has invalid zero delimiter		
(1104) FATAL: file number out of range - < >	A run time map message that is written to the error log along with other detailed information to identify the exact cause of the message. The message indicates that the generated map has a rule that is incorrectly referencing a file.	Regenerate the map for the current transaction. If message reoccurs, call support for assistance.
(1105) FATAL: filename is blank on read	Displays if a map is run and the generated tables contain a blank file name. This message is not expected to be displayed since the “Define Rules” option checks for valid file names.	Examine the Map Rules for any incorrect File I/O rules. Regenerate the map for the current transaction and examine the map generation error log. Correct any generation error log messages. If message reoccurs, call support for assistance.
(1106) FATAL: file- <file name> used both as <READ/WRITE> and KEYED	A run time map message that is written to the error log along with other detailed information to identify the exact cause of the message. This message occurs if the name file has been used as both a keyed index file and as a sequential user file.	Examine the file I/O rules that uses the FILE NAME in question. Correct the rules so that FILE NAME is not referenced as both a keyed file and a sequential file.

Message	Cause	Solution
(1110) FATAL: can't open file- < file name > (1113) FATAL: can't open index- <index name> database- <database>	Messages 1110 and 1113 are displayed as a result of a failed open file command. When a file cannot be opened, it could be due to file damage or the file might not exist on disk. The full path name of the file that cannot be opened displays in the error message.	Check that the file exists, and directory path exists. If the directory does not exist create it. If it cannot be determined why the database or index cannot be opened, call support for assistance. ECMap automatically creates the required directories; however, it is possible one of the needed directories has been deleted. If the directory exists but the file does not, determine what type of file it is by the file name and directory location. For example, the file could be a standards database that has not been loaded into the directory specified. Or perhaps the problem is a missing data file that has not been loaded into the directory. Or perhaps the map flow databases or the rules database have not been created, and the user is trying to generate a map.
(1118) FATAL: can't open EDI file- <EDI file name>	The specified EDI file cannot be opened. This Run Time map message is written to the error log along with other detailed information to identify the exact cause of the message.	Verify that the file exists on disk. If the file does exist, call support for assistance.
(1119) ERROR: not found- <file type>: <file name>	A map run time error message that occurs when a Look Up Table, or Keyed index file does not exist. The displayed file name contains the full path name of the needed file.	Place the needed file in the indicated directory or modify the map so that the file is not referenced and regenerate the map. If the file is a keyed index file, then use the "Define Rules" to remove file I/O rules using this keyed index file. If the file is a table lookup file, then "Mapping" options can be used to either create the look up table or delete any reference to it.

Message	Cause	Solution
(1120) WARNING: file not found- create?	Verifies that file creation is desired. MAPPER is designed to create some files when they are not present. For example, empty User file databases, and empty map flow databases are created by the MAPPER program. Before creation of these databases, MAPPER verifies that the user wants to create the new database for the currently selected map.	If the user responds with "Y", MAPPER creates the file. If the user responds with "N", MAPPER terminates the function.
(1122) ERROR: file <file name> not found	A file specified in the rules and/or directory structure is not found. This is a read only problem.	Check directory and filename specifications, to ensure that there is a file of that type present on the system.
(1124) FATAL: unexpected EOF while processing level <map flow level number>	An outbound map run time error message that occurs when a level's I/O rule is performed to read a user file and the read results in End Of File. The End of File is considered to be unexpected if the level has not been defined as "Optional" by the user in the "Mapper Flow" options, and the level is either not a repeating level or is a repeating level but the repeating level is executed for the first time in the loop. If this message displays, then either the user files for this transaction are incomplete or the map flow does not correctly describe the user files.	Call support for assistance.
(1125) FATAL: DBF Error	Displays when any errors have been detected by the Run Time while accessing dBASE files.	May indicate the Trading Partner file is not present. Ensure that directory location and Trading Partner status has been established.
(1126) FATAL: Directory <directory> Does Not Exist		

Message	Cause	Solution
(1130) FATAL: can't read file- <file name>	Messages 1130 through 1132 are displayed when an error occurs as a result of a read file command and the end of file has not been reached. Verify that the file name displayed exists on the disk. This message is usually displayed only after the file has been successfully opened, and if the file does exist on disk these messages should be regarded as internal program errors.	Call support for assistance.
(1140) FATAL: can't write to file- <file name> (1141) (DBCIII ERR #) FATAL: can't write to file- <file name>	Messages 1140 and 1141 are displayed if an attempt to write a record to an open database or file fails. Could be the result of a full disk, or not enough free memory.	Exit MAPPER and check for free disk space. If there is free disk space, restart MAPPER and retry the MAPPER option that caused the error. If the write fails again, call support for assistance.
(1142) ERROR: Rename <File 1> to <File2> Failed.	Rename command did not execute correctly.	Ensure that: • directory exists • filenames are correct and valid • File 2 does not already exist since the rename command can not update.
(1177) ERROR: on Breakfield	An internal program error that should never occur. If it does, it occurs when a outbound map is running, and the size of a breakfield for a user file exceeds the space that has been dynamically allocated to hold the breakfield value.	Call support for assistance.
(1181) ERROR: file- <file name> has multiple records, but record- <record name> doesn't have record type field.	A run time map message. A User file has been defined to contain several different records but the displayed RECORD NAME that belongs to FILE NAME file does not have a record type field.	Correct this by defining a record type field for the record or by changing the definition of the displayed FILE NAME to contain only one record.
(1190) ERROR: Missing <record name> Record in Map <map name>	Required type of record is not in map, such as outstat and flow record	Check flow, and/or record definitions to ensure that record is defined. Run Generate to regenerate the map.

Message	Cause	Solution
(1191) ERROR: Invalid header line in map <map name> at line <line number>	*.*MAP* file is incorrect. Somehow this file has been corrupted.	Run Generate to regenerate the map.
(1192) ERROR: Unexpected < > of < > records at line < line number> Map <map name>	Map was not completed correctly i.e.: the flow is missing,	Ensure that all map components are present (including the flow) and regenerate the map.
(1193) ERROR: Level <level number> Current Record Name Cannot be Blank	Outbound map flow has errors.	Current record is not defined in outbound flow. Define record and regenerate the map.
(1194) ERROR: Level <level number> Parent Record Name Cannot be Blank	Outbound map flow has errors.	Parent record is not defined in outbound flow. Define parent record and regenerate the map.
(1195) ERROR: Level <level number> Not Linked to Prior Level	Outbound map flow has errors.	Modify flow to include prior level and regenerate the map.
(1196) ERROR: Level <level number> Multiple Link Only's Attached to Same Prior Level	Outbound map flow has errors.	Modify flow to eliminate multiple “link onlys” attached to the same level. This is not a legal condition.
(1197) ERROR: Level <level number> Link Only Must Attach to Repeat or Master Level	Link only has been specified where it is not needed. Link only’s are only needed after repeats on the master level.	Delete the link only that is attached to the instance where this error message occurs and change the level numbers for the segments so that they are attached to the appropriate level.
(1198) ERROR: Level <level number> Should Be Placed Directly Beneath Parent Level	Outbound map flow has errors.	Check placement of entries in this flow table to ensure that parent and child levels flow correctly.
(1199) ERROR: Flow Errors In Outstate Map – Run Aborted	Outbound map flow has errors.	Check placement of entries in flow table.

On records

Some of the ECMap informational messages are the result of errors in adding, updating, reading and deleting database and index records. Other errors are the result of search or re-index failures. Some of the errors are the result of mistakes made in defining Map Flow records. Some of the messages in this section are described as internal program errors. Internal program error messages are not expected to be displayed. If an internal error message occurs or if the suggested corrective action does not solve a problem, note the message number and any associated DBCIII error number and call support for assistance.

Table 1-4 lists the format, cause, and solution (if applicable) for the ECMap messages related to records, in numeric order by message number.

Table 1-4: ECMap messages related to records

Message	Cause	Solution
`(1208) ERROR: invalid record- <EDI segment expected>`	Can occur when an inbound map is running and the inbound EDI X12 file has an invalid or missing control segment. The “EDI Segment Expected” part of the message could be “GS”, “ST” or another control segment name to indicate the control segment that the program was expecting.	Examine the error log and the bad transaction file for the inbound map run to determine where the error occurred.
`(1214) FATAL: can’t find key- <key value>`	Indicates that the index files for a MAPPER database have been corrupted.	Delete the DOS index files (**.NDX*) under the current map’s “Data Dictionary Tables” directory. When the MAPPER program is restarted, the index files under the “Data Dictionary Tables” automatically rebuilds. If this does not solve the problem, call support for assistance.
`(1222) FATAL: duplicate record- <record number> found in database- <database>`	Should not display. It can occur when a map is run but the generated tables have become damaged.	Regenerate the map for this transaction. If message occurs again, call support for assistance.
`(1223) FATAL: fseek failed during file read`	An internal error that could occur if, during a map run, an attempt to seek to a saved file location fails.	Call support for assistance.

Message	Cause	Solution
(1225) ERROR: <record name> in file <file name> doesn't have valid record type	An outbound map run error that occurs when processing a user file that contains multiple records with record type key fields, and a record is read that does not have a type key field value that matches any of the user defined type key field values.	Either correct the user file record so that it has a correct type key field value or modify the User File so that the record key field value of the user file record in error is valid.
(1226) FATAL: record number out of range- < >	Could occur when a map is run but the generated tables have become damaged.	Regenerate the map for this transaction. If message occurs again, call support for assistance.
(1227) FATAL: XML-READ gets into an infinite loop		
(1228) ERROR: No Tradstat for Tptnr <TRADING PARTNER NAME > Trans <TRANSACTION NUMBER > Vers <VERSION NUMBER> TstInd <TEST INDICATOR NUMBER>Agency <AGENCY NAME> Release <RELEASE NUMBER> – Skipping Forward	A Trade Agreement entry for this trading partner, for this version, for this test indicator, for this agency, for this release is not present.	Modify the Trading Partner Trade Agreement to link version transaction number for this trading partner. Ensure that agency, release and direction are correct.
(1230) FATAL: reading record from database- <database> (1231) FATAL: reading record- <record number> from database- <database>	Messages 1230 and 1231 display if a read of a database file fails. When this message occurs, the program has already found or added a record to the database and the read is attempting to retrieve a record that is known to exist in the database. Such an error could occur only if the database or index file has been damaged.	Check to see if the hard disk is full. Also, follow the database re-index procedures outlined in Error 1214 solution. If these corrective procedures fail, then the message can be the result of an internal program error. Call support for assistance.

Message	Cause	Solution
(1293) ERROR: Max Record Length of < > exceeded by Record < >	Occurs when the record you have just defined exceeds the system record length.	If you have requirement that a record exceeds the record boundary, you can define two records – when you write one you write the other. When defining the record, specify NONE as the end of record terminator for the first record.
(1294) ERROR: Maximum Length < > of EDI record exceeded	EDI file exceeds the maximum segment length that the system supports.	May be invalid data in the EDI file. Check the EDI file the ensure that segment terminator is present.
(1297) ERROR: Value- < > is not found in Table- <file name>.XRF	In performing a cross reference between an EDI value and a user value, the value is not present in the table.	Add the value to cross reference table and regenerate the map.
(1298) ERROR: Unknown Table Translation Name < file name >.XRF	A cross reference table has been specified but that table no longer exists. This could occur when you have specified that the table is in an external file, and the file cannot be found. Or a table has been deleted from the list of tables, but is still specified in the mapping under rules.	Verify file location if an external table is specified. If table is no longer needed, and has been deleted from the tables list, remove the rule that calls the table. If the table was erroneously removed, add it again and regenerate the map.
(1299) DEBUG: Line <line number> Read <record name> Defined Rec Len < >	Information message. Error occurs when debug mode is on. The record has been defined as a specific length, and the Record read exceeds the defined length	Informational message only.

On fields

Some of the ECMap informational messages are the result of an attempt to add a record whose key field already exists in a database. Some are the result of trying to reference a User File record field that does not exist in the User file database. Some are the result of User File record fields whose definitions in the User File have been changed since the User Field/Element Mapping defined. And, some of the messages in this section are described as internal program errors. Internal program error messages are not expected to ever be displayed. If an internal error message occurs or if the suggested corrective action does not solve a problem, note the message number and any associated DBCIII error number and call support for assistance.

Table 1-5 lists the format, cause, and solution (if applicable) for the ECMap messages related to fields, in numeric order by message number.

Table 1-5: ECMap messages related to fields

Message	Cause	Solution
(1316) FATAL: invalid field number - <number> in database - <database>	A run time map message that is written to the error log, which indicates that the map generated tables are incorrect.	Regenerate the map. If problem is not corrected, call support for assistance.
(1321) FATAL: User Set Flow Level <flow level number> is Not in Flow Map	User has overridden the flow to branch to different levels other than the defaults, and has specified a flow number that no longer exists.	Modify level number entries in outbound flow table.

On dates

Some of the ECMap informational messages are the result of problems with dates.

Table 1-6 lists the format, cause, and solution (if applicable) for the ECMap messages related to dates, in numeric order by message number.

Table 1-6: ECMap messages related to dates

Message	Cause	Solution
(1400) ??????: Invalid Century Minimum at line <>		
(1401) ???????: Invalid Date Flag at line <>		

Message	Cause	Solution
`(1402) ?????: < > Century Minimum Required To Run Map after 9/30/1999`	A result of Y2k logic.	Starting with version 4.5 and higher, Y2k date logic must be used. Choose the Utilities/Update Date option to assist you in this process.

On data and data manipulation

Many of these messages are map generation and map run time messages that occur because of invalid data, invalid Map Rules or incorrect Map Flow designs. Some of the error messages are the result of incomplete table look up databases. Some of the messages in this section are described as internal program errors. Internal program error messages are not expected to be displayed. If an internal error message occurs or if the suggested corrective action does not solve a problem, note the message number and any associated DBCIII error number and call support for assistance.

Table 1-7 lists the format, cause, and solution (if applicable) for the ECMap messages related to data and data manipulation, in numeric order by message number.

Table 1-7: ECMap messages related to data and data manipulation

Message	Cause	Solution
`(2000) ERROR: attempted to divide by zero`	A run time map message that is written to the error log along with other detailed information to identify the exact cause of the message. This message indicates that a Map Rule division failed because the denominator field had a zero value.	Check both data in User Files and values assigned to variables in the Map Rules.
`(2001) ERROR: numeric overflow`	A run time map message that is written to the error log along with other detailed information to identify the exact cause of the message. This indicates that the numeric value was too large to be stored in the User File field, or too large to be stored in the standard X12 EDI transaction file field.	Compare User File numeric field sizes against the maximum size for the standard X12 Field.

Message	Cause	Solution
(2002) ERROR: ILLEGAL USE OF DATE FIELDS IN ARITHMETIC RULE	Arithmetic operation requested on a date field cannot be performed. For example, adding a date field to a date field, or multiplying or dividing dates.	Use the "Define Rules" option to correct the rule. Allowable Date options are: • Date = Date + number • Date = Date - number • Number = Date-Date
(2028) ERROR: Level:<level number>-Loop count of exceeds maximum of <max loop occur>	A compliance error caused this message. The loop specified occurs more than the maximum number allowed.	Solution:
(2029) ERROR: Level<level number>, segment: <segment name> - occurred <number> times, exceeding maximum of <maximum segment occurrence>	A compliance map error caused this message. The segment specified occurred more than the maximum number of occurrences for that particular instance.	Solution:
(2030) ERROR: Segment <segment name> in loop <loop name> was Out-Of-Sequence.	A compliance map error caused this message. A segment in a particular loop came in the incorrect order or sequence.	Solution:
(2031) ERROR: Mandatory segment <segment name> in loop <loop name> was skipped.	A compliance map error. A mandatory segment is missing.	Look in the *Standards Guide* and compare it to your map to see what mandatory segment was left out.
(2032) ERROR: EDI Segment <segment name> not found in Transaction	One of the segments listed in the transaction should be in this transaction.	Remove the segment that is not required according to the *Standards Guide*.

Message	Cause	Solution
(2033) ERROR: EDI Segment <segment name> level <level number> not found in database- <database name>	Cause 1: An inbound run map message that is written to the error log. It occurs when the incoming X12 transaction file has a segment name that is not defined by the inbound Map at the indicated level. The run time result is that these incoming segments are ignored.	Solution 1: If that is desired, then no corrective action is necessary.
	Cause 2: The incoming X12 file has an invalid segment name. Also check the inbound Map Flow definition and the segment names that are defined for the Map. Perhaps some of the map segment names were incorrectly set to the IGNORE status, or some segment names need to be included in the map at additional levels.	Solution 2: Correct as indicated above and regenerate the map.
(2034) ERROR: end-of-file reached, while searching for control Segment	An inbound run map message that is written to the error log. This message occurs when the program reads the last record of the incoming X12 transaction file and the X12 transaction file does not have a correct ending control segment sequence, such as GE, IEA.	Examine the X12 incoming data file to check for valid control segments.
(2044) ERROR: Record Type Read <record type> Does Not Match Any In File	Occurs with a file with multiple record types. The message indicates that a record type is encountered in the data that does not match the file specifications.	Create a record for this record type and add it to your specifications.
(2045) ERROR: Invalid Date Field: <date read> format <specified date format>	An inbound run map message that indicates the date that was just read does not match the format specified.	Adjust format specification of change modify incoming data to meet specified format.
(2046) ERROR: Date Value <date read> is greater than field size <specified field size>	An inbound run map message that indicates the date that was just read exceeds the field size specified.	Define date field size and regenerate the map.

Message	Cause	Solution
(2047) WARNING: No Rule To Process Record Type <record type read>, Reading Forward	An inbound run map message that indicates that the rules for reading files for multiple record types does not define the record type just read.	Add record type to the list of records. Ensure that an I/O rule has been created that causes a read of the specified record type.
(2048) ERROR: Invalid Time Field: <record name> <field name>	Time value passed is not valid	Correct value.
(2049) ERROR: Invalid Field Value in SQL Where: <record name> <field name> <value>	Value is not valid for the Field name in the Where Clause.	Correct the rule or the data populating the value in the Where Clause.
(2050) ERROR: <record name> <field name> TimeStamp Missing Time	The time parameter in the TimeStamp has no data.	
(2051) ERROR: Select '?' and # Parameter Mismatch	Number of parameters and number of parameters passed do not match. This is usually because the SQL statement has been modified, but has not been recompiled.	Modify the SQL rule and recompile and store.
(2052) ERROR: Invalid Test Indicator: <test indicator value read>	An invalid test indicator was in the inbound data.	Look at the Test Indicator in the input data. The test indicator should be a P, T, or I in the ISA 15 section.
(2053) ERROR: No Test Indicator	An invalid test indicator was in the inbound data.	Look at the Test Indicator in the input data. The test indicator should be a P, T, or I in the ISA 15 section
(2054) ERROR: Memory variable can not be defined as a time stamp		
(2061) ERROR: Cannot switch between NCPDP Batch Versions	Cannot switch between NCPDP Batch Versions	
(2062) ERROR: Cannot Switch between Real Time and Batch	Cannot Switch between Real Time and Batch	

Message	Cause	Solution
(2063) ERROR: NCPDP MAP Has No Transmission Segments Marked	NCPDP MAP Has No Transmission Segments Marked	
(2064) ERROR: NCPDP MAP CANNOT USE DBC3 TP Files	NCPDP must use ODBC trading partner.	
(2065) ERROR: Cannot Process NCPPD Data with Non-NCPDP Map	Cannot Process NCPPD Data with Non-NCPDP Map	

On types, usage and type linking

The messages in this section focus on valid data entry and correct definitions of fields in the User Field/Element Map, User File, and Map rules. Some of the messages are displayed during data entry and other of the messages are written to error logs as the maps are generated and run. Some messages are described as internal program errors. Internal program error messages are not expected to ever be displayed. If an internal error message occurs or if the suggested corrective action for a message does not solve the problem, call support for assistance.

Table 1-8 lists the format, cause, and solution (if applicable) for the ECMap messages related to types, usage, and type linking, in numeric order by message number.

Table 1-8: ECMap messages related to types, usage, and type linking

Format	Cause	Solution
(0101) FATAL: invalid type – <field>	An arithmetic rule could be attempting to process alphanumeric field types. This error should be reported when Map generation takes place.	Modify arithmetic rule or modify data types to allow arithmetic operation on numeric fields.

Format	Cause	Solution
`(2102) ERROR: invalid Rule Parameters`	A run time map message that is written to the error log along with other detailed information to identify the exact cause of the message. This message is usually caused by a map rule with incorrect parameter type fields. This message is not expected to be displayed because the "Define Rules" option does data entry time checking on valid map rule parameters.	
`(2104) ERROR: invalid relational operator code`	A run time map message that is written to the error log along with other detailed information to identify the exact cause of the message. It is usually caused by a map rule with an incorrect relational operator code. This message is not expected to be displayed because the "Define Rules" option does data entry checking on valid relational operator codes.	If the cause is not apparent from the error log, call support for assistance.
`(2105) ERROR: invalid conditional check type`	A run time map message that is written to the error log along with other detailed information to identify the exact cause of the message. It is caused by a map rule making a comparison between incompatible field value types or by one of the field values compared not having been designated with a correct type of field. This message is not expected to be displayed because the because the "Define Rules" option and "User File" mapping options do data entry time checking on valid field types.	Examine the error log to identify the field name or map rule with the invalid field type. Then examine the User File or Map Rule definition of this field. Correct any User File or Map Rule errors. Regenerate the map. If the error occurs again, call support for assistance.

Format	Cause	Solution
(2106) ERROR: invalid EDI standard field type	A run time map message that is written to the error log along with other detailed information to identify the exact cause of the message. It is caused by a generated table field with an invalid field type. This message is not expected to be displayed because both the "User File" option and the generate map option check for valid field types.	Examine the error log to identify the field name with the invalid field type. Then examine the User File definition of this field. Correct any User File errors. Regenerate the map. If the error occurs again, call support for assistance.
(2107) ERROR: condition and field-must be a RECFLD or MEMVAR	A run time map message that is written to the error log along with other detailed information to identify the exact cause of the message. It is usually caused by an invalid outbound Conditional Check field specified during User Field/Element mapping. Conditional check fields must be defined as either record fields or memory variables in the User File database. This message is not expected to be displayed because the "User Field/Element" mapping option does data entry time checking on valid Conditional fields.	If the cause is not apparent from the error log, call support for assistance.
(2120) ERROR: unexpected authority qualifier <authority qualifier read>	In the inbound file, GS section, you have an authority qualifier that does not belong in this section.	In the inbound file, remove the authority qualifier.
(2121) ERROR: unexpected security qualifier <security qualifier read>	In the inbound file, GS section, you have a security qualifier that does not belong in this section.	In the inbound file, remove the security qualifier.
(2122) ERROR: unexpected sender qualifier <sender qualifier read>	In the inbound file, a bad sender ID qualifier is listed that is not in the Standard.	In the inbound file, verify that the ISA sender qualifier is correct.
(2123) ERROR: unexpected receiver qualifier <receiver qualifier read>	In the inbound file, a bad receiver ID qualifier is listed that is not in the Standard.	In the inbound file, verify that the ISA receiver qualifier is correct.

Format	Cause	Solution
`(2124) ERROR: unexpected end-of-segment while decoding <segment name> Segment`	An inbound run time map message that is written to the error log along with other detailed information to identify the exact cause of the message. It occurs if an end of segment delimiter is read while processing a GS or ST segment in the incoming transaction X12 file before all of the required EDI fields have been read.	Examine the error log and Incoming X12 transaction file to locate and correct the invalid GS or ST transaction file record.
`(2157) ERROR: first condition must be MEMVAR type`	Messages 2157 through 2159 are generated at run time map messages that are written to the error log along with other detailed information to identify the exact map records that caused the error message. They occur if the Record and Field names for the map condition fields in the User Field/Element maps are not correct. Outbound maps can have map condition fields at both the Segment and Sequence level. At the Segment level of an outbound map, the first condition must be either blank, a MEMVAR or a RECFLD. At the Sequence map record level, an outbound map must be blank or have a MEMVAR as the first map condition field. Inbound maps do not have map condition fields at the Segment level. At the Sequence map record level, an inbound map must have a blank, a MEMVAR or a RECFLD as the first map condition.	Regardless of the level or map direction, if a first map condition field is specified, the second map condition field must be MEMVAR, RECFLD, or STRVAR.

Format	Cause	Solution
`(2164) ERROR: on <READ/WRITE>, empty (M)andatory field- <field name>`	A run time map message that is written to the error log along with other detailed information to identify the exact cause of the message. It indicates that a field that has been defined as a mandatory User File field has a blank or zero value. For inbound maps, this message is generated when a record is read and a mandatory User File data field is blank. For outbound maps, this message is generated when a record is written and a mandatory User File data field is blank.	If the User File field does not need to be mandatory, the error can be corrected by simply changing the User File field's mandatory attribute status. If the field is truly a mandatory field, then for inbound maps correct the User data file that contains the blank field. For outbound maps, examine the mapping rules to determine a possible cause of the blank mandatory field. For outbound maps, it is the user defined map rules working with the User Field/Element map to place a non-blank value into a User record field before it is written.
`(2168) ERROR: must map to RECFLD or MEMVAR`	A run time map message that is written to the error log along with other detailed information to identify the exact cause of the message. It is caused by an inbound User Field/Element Map not having a RECFLD or MEMVAR defined in the Record field of a sequence record. For inbound maps, the Record field indicated where to store an inbound X12 transaction value.	Use the User Field/Element mapping option to correct the Record field name and regenerate the map.

Format	Cause	Solution
(2169) ERROR: empty value mapped to EDI mandatory field <record name> from <field name>	A run time map message that is written to the error log along with other detailed information to identify the exact cause of the message. It indicates that a field that has been defined as an mandatory EDI standard field has a blank or zero value. For inbound maps, this message is generated when an X12 record is read and a mandatory EDI field is blank. For outbound maps, this message is generated when a record is written to the X12 file and a mandatory X12 EDI field is blank.	For an inbound map, the inbound X12 transaction file needs to be corrected. For an outbound map, examine the error log, the user data file, the map rules and the User Field/Element map to determine and correct the cause of the blank EDI field value. If the error was caused by a missing or incorrect map rule or User Field/Element map, regenerate the map and run the map again.
(2170) ERROR: Mandatory <defined length> EDI Len < Min: Value <minimum specified standards length>	A mandatory EDI field is defined as less than the minimum length in the EDI standards.	Contact Trading Partner to update transaction to meet EDI specification
(2171) ERROR: Invalid <segment name> Segment <code> code	A run time map message that is written to the error log along with other detailed information to identify the exact cause of the message. For inbound maps, this message is generated when required information within the EDI elements such as the X12 version, are empty. In some cases, this message is generated for information only, and in most cases, processing continues.	Bad EDI file. Inform sender, or fix data.
(2172) ERROR: Mapping To Numeric EDI Field with Non-Numeric Data: <field value >	Occurs when mapping an alphanumeric field to a numeric EDI field. Inbound data received is not numeric.	Change the alphanumeric data to numeric or change the mapping specifications to meet the current data type or report the message to file sender regarding bad data.
(2173) ERROR: Incoming EDI Numeric Field Has Non-Numeric Data: < field value >	The EDI file read is invalid for this particular element.	The data for this element is bad. Report message to file sender.

Format	Cause	Solution
(2174) ERROR: Numeric Field Contains Non-Numeric Data < field value >	User field specified as numeric has non-numeric data.	Change the definition of the user file field or modify incoming data.
(2175) ERROR: Level <level number> Cond Check of Seg <segment name>, Non-Numeric Data: < >	Segment specified as numeric has non-numeric data.	Change the Segment definition to accommodate numeric values or modify incoming data.
(2176) ERROR: Level <level number> Cond Flow Check of Seg <segment name>, Non-Numeric Data: < >	Level specified as numeric has non-numeric data.	Change the definition or modify incoming data.
(2177) ERROR: Map Cond Check Numeric Field has Non-Numeric Data: < field value >	Condition specified as numeric has non-numeric data.	Change the definition or modify incoming data.
(2180) ERROR: Invalid Interchange Control Version Number: %s	Added for ISA error checking.	
(2181) ERROR: Invalid Interchange Control Standards Identifier: %s	Added for ISA error checking.	

On formats

The messages in this section are the result of either User File database fields defined incorrectly, or the data in an actual user file field not corresponding to the User File database definition.

Table 1-9 lists the format, cause, and solution (if applicable) for the ECMap messages related to formats, in numeric order by message number.

Table 1-9: ECMap messages related to formats

Format	Cause	Solution
`(2408) ERROR: length must be 4 or 8 for CD field type`	An interactive data entry error that occurs during User File definition if a field's length is less than 4 and the field is specified to be a computational field (i.e. CD).	Either change the field type or change the field length to 4.
`(2411) ERROR: Status MemVar Field Size too small for Err Code`	Indicates that a Keyed I/O Map Rule has been defined to use a MEMVAR variable as an error code field and the MEMVAR length is less than 2. The MEMVAR length must be at least 2 in order for the MEMVAR to be large enough to hold the error code.	Use the User File definition option to redefine the MEMVAR field length. Or use the Map Rule definition option to select a different MEMVAR. If this message occurred during a map run, the map must be regenerated after the MEMVAR length has been corrected.
`(2422) WARNING: field truncated`	A map run or map generation message that is written to the error log along with other detailed information to identify the exact cause of the message. It could be the result of a map addition rule that results in a value too large to be stored in the designated field. This message can also occur for inbound maps when the data field mapped to the User File field exceeds the defined User File field length and the User Field/Element map for the field does not explicitly specify that field truncation is allowed. For an outbound map this message displays when the data field mapped to an EDI transaction field exceeds the maximum size defined EDI field length.	Use the User Files options to change the lengths or specify that truncation is allowed.

Format	Cause	Solution
(2424) WARNING: <numeric type> overflow, <full number> truncated to <number>	A map run message that is written to the error log along with other detailed information to identify the exact cause of the message. It occurs when mapping a numeric field type results in truncation, and the field mapped to has not been specified as a field that can be truncated. Both the full number and the resulting truncated number are written as part of the message.	Use the User Files options to change the lengths or specify that truncation is allowed.
(2444) ERROR: Delimiter for < start position > Not Found in < field value >	Occurs when creating substring and/or concatenation rules. Rule is executed based on the presence of a specified delimiter, which cannot be found. For substring errors: Delimiter for start position is not found in Source Field Delimiter for Substring length is not found in Source Old For Concatenate errors: Concatenate Delimiter for Destination is not found in LHS Variable	Check data for presence of delimiter.

On mapping

The messages in this section focus on interactive User Field/Element mapping errors and interactive Map Flow errors. Many of these errors are caused by differences between EDI field definitions and User File field definitions.

The format, cause, and solution (if applicable) for the ECMap messages related to mapping are listed below, in numeric order by message number.

Table 1-10 lists the format, cause, and solution (if applicable) for the ECMap messages related to mapping, in numeric order by message number.

Table 1-10: ECMap messages related to mapping

Message	Cause	Solution
(3024) ERROR: Prior ST Transaction Removed From <file name> Due To < reason >	An abort transaction was executed either by the user or because of bad map information. User data was backed out.	If it does occur because of bad data, fix data and rerun the map. If abort transaction was erroneous placed in map flow, remove the command, and regenerate map.
(3026) ERROR: <element or subelement> Map to CNDVAR should not be Conditional	A conditional store was defined for an element or sub-element with conditions applied.	From the mapping window, modify the element and remove the conditional aspect for the conditional store or change the element from a conditional store element.
(3027) ERROR: Group Aborted by SysVar		
(3028) ERROR: Interchange Aborted by SysVar		

On rules

The messages in this section are primarily valid map rule and map condition messages.

Table 1-11 lists the format, cause, and solution (if applicable) for the ECMap messages related to rules, in numeric order by message number.

Table 1-11: ECMap messages related to rules

Message	Cause	Solution
(4000) ERROR: Rule out of valid range	The message indicates that one rule number (rule index in the rule table) is over the total rule number (in the rule table). It is not expected to occur, and would normally be caught at data entry time or map generated time.	If this error does occur, regenerate the map. If the error occurs again, call Support for assistance.
(4001) ERROR: Rule out of valid range for level - <level number> Segment - <segment name>	This message indicates that the rule with <rule number> with <segment number> is out of valid rule range that is defined in the map. It is not expected to occur and would normally be caught at map-generation-time.	Regenerate the map. If it occurs again, call Support for assistance.

Message	Cause	Solution
(4002) ERROR: Rule #(<beginning rule number> - <ending rule number>)out of valid range for level <level number>	These messages are run time map messages that are written to the error log along with other detailed information to identify the exact cause of the message. The message indicates that the generated map has an undefined rule number. This message is not expected to occur. Errors of this type would normally be caught at data entry time, or map generation time.	If it does occur, examine the error log to determine the rule number that was invalid. Then locate where this rule was used in the User Field/Element Map or the Map Flow. Correct by adding the Defining the Map rule or eliminating reference to it. Then regenerate the map.
(4006) ERROR: invalid conditional check value	An outbound run time map message that is written to the error log along with other detailed information to identify the exact cause of the message. It indicates that a map record had the first field of a map condition line defined correctly but did not have the required second field of the map condition defined as a valid RECFLD, MEMVAR or STRVAR.	Examine the error log to determine the map record in error, and use the User Field/Element option to correctly define the second map condition field. Then regenerate the map.
(4009) ERROR: Rule command undefined	Occurs during Map Rule definition if the user attempts to use an undefined rule number as part of a rule command.	Use Map Rule definition to define the rule to be used in another rule.
(4013) ERROR: Rules for SQL Stored Procedure are obsolete. Please run Generate Map	Occurs when running a map created in a previous release with the new release of ECMap. The rules for Stored Procedure in the new version have changed	Map rules must be modified to use the latest rule definitions. Then regenerate the map under the new release.
(4019) TRACE: Level <level number> -> <level number> processing Segment: <rule number >-No<line number>	Occurs before executing the OK Rule in the <level> to <level> processing and the rule number is <rule number>, line number is <line number>	
(4020) TRACE: Level <level number> processing Rule: <rule number >-No<line number>	Occurs before the command executed, which is at <line number> of <rule number> during the <level number> processing.	

Message	Cause	Solution
(4021) TRACE: Level <level number> No <Before/After> Rules to process	Indicates that the rule with rule <level number> does not have a “before and after” command to process.	
(4022) TRACE: Rule <rule number> Changed Level To <level number>		
(4023) TRACE: Performing Rule <rule number>	Indicates that at runtime the rule <rule number> is executed.	
(4024) TRACE: Rule <rule number> Line <line number> Start Perform While on Rule <rule number>	These informational messages occur when Verbose Trace is on. They report progress at particular points in time when running the map.	
(4040) ERROR: Parameter <parameter> is empty on assignment	You try to assign a parameter to a system variable or a memory variable but this parameter is empty.	Verify that the parameter in parameters tab at Run Map window of ECMap. Verify that the command that assigns the parameter to a system variable or a memory variable.
(4050) FATAL: MemVar <memory variable> has uncompleted sql statement		

On trading partners

These messages are generated due to invalid or mismatched Trading Partner numbers or fields.

Table 1-12 lists the format, cause, and solution (if applicable) for the ECMap messages related to trading partners, in numeric order by message number.

Table 1-12: ECMap messages related to trading partners

Message	Cause	Solution
`(5000) OK: executed ABORT_TRANS command for Trading Partner <internal trading partner number>`	A run time map message that is written to the error log. It indicates that the map transaction for the displayed Trading Partner number was aborted by the execution of an ABORT Transaction map rule.	If an error in compliance maps, this error also appears. Solve the earlier fatal errors listed before this message.
`(5001) ERROR: aborting- database Customer has no record for Trading Partner <internal trading partner number>`	An outbound run time map message that is written to the error log along with other detailed information to identify the exact cause of the message. It indicates that the outbound transaction for the displayed Trading Partner was aborted because the Trading Partner definition database, Customer, has no entry for the displayed Trading Partner number.	Either modify this Trading Partner number, or correct the user data file that contains the invalid Trading Partner number.
`(5002) ERROR: aborting- database <trade agreement database> Table <transaction code> has no Trading Partner <internal trading partner number>`	These are outbound run time map messages that are written to the error log along with other detailed information to identify the exact cause of the message. It indicates that the outbound transaction for the displayed Trading Partner was aborted because the Trading Partner definition database, TRADSTAT, has no entry for the displayed Trading Partner number and transaction set or Map Table code.	Either define this Trading Partner number and transaction set by using the Trading Partner menu option, or correct the user data file that contains the invalid Trading Partner number.
`(5004) ERROR: Trading Partner not made current`	An outbound run time map message that is written to the error log along with other detailed information to identify the exact cause of the message. It indicates that the Master Outbound Map Flow record has not been defined with a Trading Partner field. This error is not expected to happen since the Map Flow and map generation options check for this error condition.	Use the User File or Map Flow options to include the Trading Partner field and regenerate the map.

Message	Cause	Solution
(5006) ERROR: Trading Partner ID- <internal trading partner number> match not found	A run time map message that is written to the error log along with other detailed information to identify the exact cause of the message. It indicates that the displayed Trading Partner was not found in the Trading Partner definition databases. This could be due to invalid Trading Partner data or because the displayed Trading Partner has not been defined with the Trading Partner menu option.	Either define this Trading Partner number by using the Trading Partner menu option, or correct the user data file or X12 transaction file that contains the invalid Trading Partner number.
(5007) ERROR: Trading Partner is blank	An outbound run time map message that is written to the error log along with other detailed information to identify the exact cause of the message. It indicates that the Trading Partner user data file record was read and the Trading Partner field was blank.	Correct the user data file record to include a valid Trading Partner.
(5019) TRACE: Trading partner: <internal trading partner number> New Transaction <transaction code>	Cause:	
(5020) TRACE: New Trading Partner: <trading partner name>	Found a new trading partner in the data for an inbound or outbound map.	
(5021) TRACE: Changed Trading Partner: <trading partner name>	This message usually appears after 5020. The trading partner is now different than earlier.	
(5022) TRACE: Map change for Trading Partner <trading partner name> Map: <map name>	This message usually appears after 5020. The map is now different than earlier.	

Message	Cause	Solution
(5023) TRACE: Tradstat found for Tptnr <trading partner name> Trans <transaction code> Vers <version number> TstInd <test indicator value> Agency <agency> Release <release>, Overlay < >	These informational messages occur when Verbose Trace is on. They report that a new Trading Partner has been encountered during the mapping process.	
(5024) FATAL: Trading Partner Database Structure Needs To Be Updated	The map you ran has a previous version of the ODBC trading partner from an earlier version.	Open Map Properties. Reselect DSN and select OK and that updates your trading partner database.
(5025) FATAL: Map uses Repeat Elements but ISA_TYPE < 00402	Cause:	Solution:

On outbound map level changes

Table 1-13 lists the format, cause, and solution (if applicable) for the ECMap messages related to outbound map level changes, in numeric order by message number.

Table 1-13: ECMap messages related to trading partners

Message	Cause	Solution
(5030) TRACE: Level <level number>, combined files Break flag was set	This message occurs when Verbose Trace is on. Informational message indicates that when a read record was performed, the record did not match the defined record type and the Multiple Files switch has been turned off. Map continues to search for the correct level to go to, based on this record type.	
(5031) TRACE: Level <level number>, End-of-File Break, normal end of transaction	This message occurs when Verbose Trace is on. Informational message occurs at the Master Record level indicating the end of the file has been reached and the transaction has ended normally.	

Message	Cause	Solution
(5033) TRACE: Level <level number>, End-of-File Break	This message occurs when Verbose Trace is on. Informational message indicates the end of the file has been reached.	
(5034) TRACE: Level <level number>, generated Keyfield Break - Old Keyfield: <value> New Keyfield: <value> Length: <length>		
(5035) TRACE: Level <level number>, Keyfield Break - Old Keyfield: <value> New Keyfield: <value>		
(5036) TRACE: Level <level number>, generated Breakfield Break - Old Breakfield: <value> New Breakfield: <value> Length: <length>		
(5037) TRACE: Level <level number>, Breakfield Break - Old Breakfield: <value> New Breakfield: <value>		
(5038) TRACE: Level <level number>, generated Keyfield Linkfield Break, Keyfield: <value> Linkfield: <value> Length: <length>	Occurs when Verbose Trace is on. Informational message indicates a mismatch for the Keyfield and Linkfield. Values of the various fields are displayed in the message.	

Message	Cause	Solution
(5039) ERROR: Level <level number>, Keyfield does not match Linkfield and current record is not optional; Keyfield: <field name> Linkfield: <field name>	Is received when processing multiple files. A record has just been read that is defined as a key field and must match a field in another record, and the field is not optional.	Flow can be incorrect. Modify flow to indicate you are reading this record out of sequence.
(5040) TRACE: Level <level number>, Keyfield does not match Linkfield and current record is optional; Keyfield: <field name> Linkfield: <field name>	Occurs when Verbose Trace is on. Informational message occurs when reading multiple files where the key fields do not match. No error is generated since the record is optional.	
(5041) TRACE: after Keyfield/Linkfield Break	Occurs when Verbose Trace is on. Informational message reports progress after encountering a Keyfield / Linkfield Break.	
(5042) TRACE: Level <level number>, TOP-OF-LOOP processing <file type> files < > Loop Depth:- < > New Level: < > Brk Level: < >	Occurs when Verbose Trace is on. Informational message appears at the top of the Trace file that reports the level, the file type, the loop depth, the new level and break level.	
(5043) TRACE: Level <level number>, Processing LINK ONLY	Occurs when Verbose Trace is on. Informational message reports level number during link only process. The format, cause, and solution (if applicable) for the ECMap messages related to map level changes are listed below, in numeric order by message number.	
(5044) FATAL: Record Type Break and Level <level number>, Not Optional		

Message	Cause	Solution
(5045) ERROR: Level <level number>, Missing Optional Record		
(5046) FATAL: File <file name> Missing No Record Type Field	Received when processing multiple files in that you are reading two files and matching them up by fields. A record has just been read that is defined as a key field and must match a field in another record, and the field is not optional.	Solution 1: This record can in fact by optional. Change inbound flow so that reading this record is optional. Solution 2: Flow can be incorrect. Modify flow to indicate you are reading this record out of sequence. Solution 3: Using wrong field to link the two records. Change inbound flow.
(5047) TRACE: Reading File <file name> Record <record name>	Occurs when Verbose Trace is on. Informational message indicates current reading of a file and record. Also indicates end of file for the file name specified.	
(5048) TRACE: <I/O operation > File <file name> Record <record name>	Occurs when Verbose Trace is on. Informational message indicates reads, writes and SQL selects and inserts for a file and record.	
(5049) TRACE: Rule <rule number> Executing Procedure <procedure name>	Occurs when Verbose Trace is on. Informational message indicates execution of procedure.	
(5050) TRACE: < > Writing File <file name> Record <record name>		
(5051) FATAL: Combined File Missing Record Type		
(5052) TRACE: Writing File < > < > < >		

Message	Cause	Solution
(5999) ERROR: Failed to Get Status Information on File <file name>		

On inbound map control segments

Table 1-14 lists the format, cause, and solution (if applicable) for the ECMap messages related to inbound map control segments, in numeric order by message number.

Table 1-14: ECMap messages related to inbound map control segments

Message	Cause	Solution
(6000) TRACE: Reading < >- Seg Delim <segment delimiter> Elem Delim <element delimiter> Sub Delim <subelement delimiter> Test Ind <test indicator value>		
(6001) ERROR: expected < segment> read , Skipping Forward to Next	Expecting a control segment such as GS or ST and MAPPER is not finding it. It therefore, is reading forward to next occurrence of one of those records.	EDI data sent is corrupted. Report to sender.
(6002) ERROR: expected read , Aborting	Expecting a read of a particular segment type and did not find it. The MAPPER aborts the transaction.	EDI data sent is corrupted. Report to sender.
(6003) TRACE: Looking.. Segment (control)	Occurs when Verbose Trace is on. Informational message indicates processing is looking for a particular Segment control.	
(6004) TRACE: Level <level number>, Segment (control)	Occurs when Verbose Trace is on. Informational message indicates current level number and looking for a particular Segment control.	

Message	Cause	Solution
(6005) ERROR: errors in processing ST, skipping forward	Expecting a control segment such as GS or ST and MAPPER is not finding it. It therefore, is reading forward to next occurrence of one of those records.	EDI data sent is corrupted. Report to sender.
(6006) TRACE: Level <level number>, Segment (control) Rules	Occurs when Verbose Trace is on. Informational message indicates current level number and the execution of a particular control rule number.	
(6007) TRACE: Level <level number>, Segment (default) Rules	Occurs when Verbose Trace is on. Informational message indicates current level number and the execution of the default rule. This is used in the default flow, where you can change levels based on the default.	
(6008) TRACE: Level <level number>, Segment <element> Rules	This message occurs when Verbose Trace is on. Informational message indicates that at this Segment Name/Element number the execution of a particular rule number is performed.	
(6009) TRACE: Level <level number>, Segment (mapping)	Occurs when Verbose Trace is on. Informational message indicates that at this level, the Segment is mapped.	
(6010) TRACE: Level <level number>, Segment (conditional) <NOT MAPPED>	Occurs when Verbose Trace is on. Informational message indicates that at this level, the Segment is conditional and is not mapped.	
(6011) TRACE: Level<level number>, MEMVAR Rules	Occurs when Verbose Trace is on. Informational message indicates levels have changed based on a Memory variable rule.	

Message	Cause	Solution
(6012) TRACE: CNDVAR LOAD	Occurs when Verbose Trace is on. Informational message indicates the processing of a Segment loading the Conditional Store variables.	
(6013) ERROR: Not Terminated By	The EDI X12 envelope defines segments that must be followed by specific segments. The MAPPER has encountered a Segment that is not in the proper location, or has not been properly terminated before encountering another Segment	EDI data sent is corrupted. Report to sender.
(6014) ERROR: ISA # < > Not Equal to Expected # < >, Resetting ISA # to < > for Tptner <trading partner name >	If WWIXERR set, this message can appear. Error based on ISA control counters. Not finding correct control count and generating message accordingly.	Bad EDI data. Report to sender.
(6015) ERROR: Group Control # < > Not Equal to Expected Number < >	Group control count does not match expected resulted.	Bad EDI data. Report to sender. The following messages are related to the BIN segment, that is the binary segment. (allows you to place binary data within an X12 transmission)
(6016) ERROR: Incoming EDI BIN seg Numeric Field Has Non-Numeric Data: < > Transaction Aborted	Number of bytes in file in not numeric.	Bad Data. Report to Sender.
(6017) ERROR: Incoming EDI BIN segment with Non-Zero Char Count is Missing File Transaction Aborted	Actual data after the character count can not be found.	Bad Data. Report to Sender.
(6018) ERROR: Incoming EDI BIN File <binary file name> EOF After < > Chars	Number of characters transmitted does not match expected number of characters.	Bad Data. Report to Sender.
(6019)`WARNING: BIN FILE <binary file name> is EMPTY	Binary file has no data.	Bad Data. Report to Sender.

Message	Cause	Solution
(6020) WARNING: BIN FILE NAME BLANK	Name of binary file has not been entered. Parameter is blank.	Bad Data. Report to Sender.
(6021) ERROR: Can't Open BIN File <binary file name> Transaction Aborted	Binary segment was transmitted and system could not create a file definition to place binary data.	Bad Data. Report to Sender.
(6022) ERROR: Reset EDI File to First Seg Delim in BIN FILE after < > Chars	Binary segment was transmitted and system could not create a file definition to place binary data. Length of binary file does not match the BIN segment specifications.	Bad Data. Report to Sender.
(6023) ERROR: Read Unexpected < >		
(6024) ERROR: < > Not Terminated by < >		
(6025) ERROR: ISA Control Number < > exceeds max length		
(6026) ERROR: ISA Control Number not numeric using < >		
(6029) TRACE: Level <level number>, <amount of RAM available> of RAM Memory Free	Occurs when Verbose Trace is on. Informational message indicates the amount of RAM Memory available at this level.	
(6030) TRACE: Level <level number>, Segment<element > Edit Rule	Occurs when Verbose Trace is on. Informational message indicates that at this Level, Segment, Element, this rule is performed.	
(6031) TRACE: Level <level number>, Segment (mapping)	Occurs when Verbose Trace is on. Informational message indicates an element is mapped at this Level, Segment.	
(6032) TRACE: Level <level number>, Segment (conditional) <NOT MAPPED>	Occurs when Verbose Trace is on. Informational message indicates an element is not mapped at this Level, Segment, because it is conditional	

Message	Cause	Solution
(6033) TRACE: Level <level number>, <#> processing RPTMAP Segment	Occurs when Verbose Trace is on. Informational message indicates a Repeat Map function is processed for this Level, Segment.	
(6034) TRACE: Level<level number>, Loading Next <#> CNDVAR	Occurs when Verbose Trace is on. Informational message indicates that the next Conditional Store Variable is loaded for this Level.	
(6050) ERROR: Segment <segment name> exceeds Max Elements	Occurs on Inbound when the expected number of elements for this segment is exceeded.	Check incoming data.
(6051) ERROR: Level <level number>, Ignored < > Contains Data		
(6060) ERROR: Invalid source string		
(6061) ERROR: Invalid search string		
(6062) ERROR: Invalid number of occurrence string		
(6063) ERROR: Invalid replace string		
(6064) ERROR: The replace string is too big		
(6080) ERROR: Batch Group Count *<value>* Does not Match Trailer Count *<value>*	The number of records in the batch does not match the count in the trailer record.	
(6081) ERROR: Batch Control Number *<value>* Does Not Match Trailer *<value>*	The control number in the batch header does not match the batch trailer control number.	
(6082) ERROR: EOF Reached - Expecting %s	EOF (end of file) is received unexpectedly.	
(6082) ERROR: EOF Reached - Expecting %s	Errors prevented the Batch file from being processed.	
(6084) OK: Received Batch Transmission Error Header	An error occurred in the batch transmission header.	

Message	Cause	Solution
(6085) ERROR: Batch Header Number Not Numeric: *<value>*	The batch header number was not a valid numeric value.	
(6086) ERROR:Batch Transmission Type Invalid: *<value>*	Batch Transmission Type Invalid	
(6087) ERROR: Batch Header Has Version: <value> Expecting	An unexpected Batch Version is sent.	
(6088) ERROR: Expected Batch Trailer, Read Batch Header	Expected Batch Trailer, Read Batch Header.	
(6089) ERROR:Invalid Transmission Header, Searching Forward	The transmission header is invalid. The program searches ahead for the next header.	
(6090) ERROR: Transmission Bin Header Number Not Numeric: *<value>*	Transmission Bin Header Number Not Numeric: <value>	
(6091) ERROR: Mandatory NCPDP Field %s is Blank	A mandatory field is missing.	
(6092) ERROR: NCPDP Telecom Version <value> unexpected	An unsupported Telecommunications version is trying to be processed.	
(6093) ERROR: Read Batch Header, Expected Group or Trailer	Read Batch Header, Expected Group or Trailer	
(6094) ERROR: Invalid TP Lookup Choice for NCPDP	An inbound error. The type of TP lookup is invalid for NCPDP, because NCPDP requires an ODBC trading partner.	
(6095) ERROR: Invalid NCPDP Segment Name %s	Invalid NCPDP Segment Name %s	
(6096) ERROR: Transmission Hdr Transaction Count *<value>* But Actual Count is *<value>*, Aborting Trans	The actual count of transactions in a transmission does not match the value specified in the file.	
(6097) ERROR: Expected Segment Sep After Group Sep, Aborting Trans	Expected Segment Sep After Group Sep, Aborting Trans	

Message	Cause	Solution
(6098) FATAL: Transaction Header Not Found, Aborting	Transaction Header Not Found, Aborting	
(6099) ERROR: Transaction Count %ld Does Not Match Transactions	Transaction Count %ld Does Not Match Transactions	
(9000) ERROR: NCPDP switch does not match TP lookup switch	A TP lookup is used that does not support the inbound type (Batch or Telecom) selected.	
(6070) ERROR: Invalid string offset or length		
(6071) ERROR: Exceeds the string boundary		
(7001) ERROR: No Table Ref Destination Field	Cross reference rule executed to a table but there is no field for it to go into.	Check cross reference table definitions to ensure that destination field has been defined.
(7002) OK: Transaction for Trade Partner<trading partner name>Copied To File<file name>		
(7003) OK: Destination File <file name>	Information only. Transaction was copied to a file instead of passing through a map. 7002 displays what transaction was copied to what file. 7003 displays destination filename.	This is controlled through the Trade Agreement records where you instruct the system to copy the EDI data to another file while it is passing through a map.
(7004) WARNING: No Flow Records Defined for < map name >	Indicates that flow has not been defined.	Define the levels and flow information and regenerate the map.
(7005) WARNING: Map Line Count Does Not Match HDR Count	Indicates the *.MAP file is invalid or has somehow been corrupted.	Regenerate the map. If the error occurs again, call support for assistance.

Message	Cause	Solution
(7006) FATAL: < xref map table > Index Out of Range	Occurs during map generation when loading the map into memory. This internal check occurs when table header information such as the total number of tables in map does not match the number of tables in the map.	Regenerate map. If the error occurs again, call support for assistance
(7007) FATAL: Invalid record in map at <line number> line	Indicates that map definition contains errors.	Check map definition and modify. regenerate map. If the error occurs again, call support for assistance.
(7008) FATAL: No Trading Partner Fields Were Defined	No Trading Partner field has been defined in the application.	First record read on an outbound map must contain a field where the attribute has been set to Trading Partner. Check Trading Partner Status to ensure that field attribute has been selected.
(7009) FATAL: Map Error Keyed IO Rule with No Filename	A keyed- I/O rule was executed with no filename specified.	Modify the rule in question.
(7010) ERROR: < Backout > point < point position > out of range	The number of checkpoints or backout points is over the maximum number of system-defined checkpoints.	Call Support for assistance.
(7011) ERROR: < Files/X12 > Backout point < point number > Called When File Closed	System error message for the Checkpoint Backout command.	Call Support for assistance.
(7012) ERROR: < >Backout point < > With No Prior Save	Backout command was requested before a checkpoint command was issued.	Modify map to insert a checkpoint prior to doing a backout. All backouts must have prior checkpoints, as you are backing out to the prior checkpoint.
(7013) OK: EDI Backout point < point number > Executed.	Information only. The backout was executed.	

Message	Cause	Solution
(7014) FATAL: Failed Backout Pt < point number > TpNo <internal trading partner number>	System error message. Backout failed for this trading partner.	Call support for assistance.
(7015) ERROR: Output Files Reset Due To Transaction Errs	Indicates that the system encountered a Fatal error and backed out the transaction.	Data is corrupted and has been backed out. Modify data or report to sender.
(7016) OK: Output Files Reset to Backout Point < point number >	Information only. Output Files Reset to Backout Point.	
(7017) FATAL: Destination File Switch and not ST Seg	Occurs on Inbound when the EDI Out Destination File field has been checked in the Trade Agreement record. Performs mapping rules on incoming data. The ST segment initiates this action.	Call support for assistance.
(7018) FATAL: Exceeded max Field size < system-defined maximum size of a field > searching for Field delim, EDI Rec # < inbound number of X12 records >	Missing delimiters.	Modify file and regenerate the map.
(7019) FATAL: Failed trying to open < original number > File, errno = < >	Insufficient number of file handles.	Set system's config.sys file to allow for a sufficient number of files to be opened. **Note** If you are mapping multiple record types, the system creates a new file for each record type.
(7020) FATAL: Could Not Load < dynamic link library name>	Cannot find <dynamic link library> in system paths.	Double check whether the <dynamic link library> path is located in the system path.
(7021) FATAL: Could Not Get Address for Procedure <procedure name>	To use User exit command, the *USEREXIT.DLL* file must be available.	Ensure that *USEREXIT.DLL* file is in current path.

On inbound map checkpoints

Table 1-15 lists the format, cause, and solution (if applicable) for the ECMap messages related to inbound map checkpoints, in numeric order by message number.

Table 1-15: ECMap messages related to inbound map checkpoints

Message	Cause	Solution
(7022) OK: Output Files Checkpoint < > at < > < >	Cause:	
(7023) OK: EDI OUT File < > Chkpt < > at < >	Information only. Advises that checkpoint has been set.	
(7024) ERROR: Invalid Date Assignment	Moving date from one field to another field using an invalid operation.	Modify rule to perform operation.
(7025) ERROR: < > Control Count Mismatch < > vs < >	Control counts on the GE do not match the GS.	Bad EDI data. Inform sender.
(7026) ERROR: < > Reference Mismatch: < > vs < >	Control counts on the GE do not match the GS.	Bad EDI data. Inform sender.
(7031) ERROR: Non-Numeric < > Control Count: < >	Control counts are not numeric for the SE Segment	Bad EDI data. Inform sender.
(7034) ERROR: Missing < > Control Count: < >	Control counts are missing for the SE Segment	Bad EDI data. Inform sender
(7037) FATAL: Rule <rule number> L < > MemVar File Name > 256	Performed an operation on a file where a defined memory variable is a file name and the size of the memory variable file name is greater than 256 characters.	Modify extended file name to meet the maximum limit imposed.
(7038) ERROR: No Open File: <file name>	Occurs when a file management rule has been issued to close a file that is not opened.	Modify file management rule.
(7039) TRACE: < > < > < >	A generic trace message that occurs when Verbose Trace is on.	
(7040) FATAL: Nested Perform Limit Exceeded	Runtime program has a limit of 50 nested rules. This message can occur when a rule calls itself within a nested rule.	Modify perform rule.

On running the EDI product as an adapter

Table 1-16 lists the format, cause, and solution (if applicable) for the ECMap messages related to running the EDI adapter, in numeric order by message number.

Table 1-16: ECMap messages related to running the EDI adapter

Message	Cause	Solution
`(8000) ERROR: ADK does not support %s data type of %s field`	The specified field is an unsupported data type for use with NDO metadata. Currently, the only unsupported field type is Packed Decimal, F_Comp3.	For an unsupported field type, assigned a supported field type in another record. Then use that record for the NDO.
`(8001) ERROR: Put NDO object into NDO queue failed`	The internal queue fails to hold another NDO object.	System error. Call the vendor.
`(8002) ERROR: Incompleted NDO structure parent node missing Transaction Abort.`	The current record placed in the NDO data tree is does not have its specified parent already in the NDO tree.	Correct the map so that the NDO_WRITE command of a parent record is always performed before using NDO_WRITE for a child record.
`(8003) ERROR: Cannot find record %s's parent %s in data tree, skip this node.`	When reading a NDO Data tree, this message occurs if the parent child relationships in the data tree do not match the parent child relationships defined in the map.	Check that the repository schema matches the parent child relationship described in the map.
`(8004) ERROR: < field name > field has invalid date < date value >.`	Date format of <date value> is invalid.	Check input data field and data type of this <field name>
`(8050) ERROR: Does not support %s type of %s field.`	When reading the NDO data tree, the node data type trying to be read is not a supported conversion type. Currently, conversion is not supported for DT_VoidPTR and DT_Binary.	Do not use DT_VoidPtr or DT_Binary as field type for NDO data trees that are to be read by the EDI Adapter.
`(8051) ERROR: <Schema name> does not match <filename>.`	The schema name of the NDO Data Tree Being read from a Queue must match the schema name defined in the map. Note in the map the schema name is the file name.	Schema name of either the map or the NDO data tree must be changed so that both are identical.

Message	Cause	Solution
(8052) ERROR: Record %s undefined in map file, skip all leaf nodes.	A container node name does not match any record name in the map	Correct either the NDO schema or the map. All container NDO node names should be defined as a record in the map.
(8053) ERROR: <Field name> undefined in record <record name> in the map file.	When reading an NDO data tree, the field name of one of the nodes is not matched by a map field name in the map.	Redefine map or schema so that all field names in the schema matches a field name in the map.
(9000) ERROR: NCPDP switch does not match TP lookup switch	A TP lookup is used that does not support the inbound type (Batch or Telecom) selected.	

Microsoft standard ODBC error messages

SQLError returns SQLSTATE values as defined by the X/Open and SQL Access Group SQL CAE specification (1992). SQLSTATE values are strings that contain five characters.

The following table lists SQLSTATE values that a driver can return for SQLError. The character string value returned for an SQLSTATE consists of a two character class value followed by a three character subclass value. A class value of “01” indicates a warning and is accompanied by a return code of SQL_SUCCESS_WITH_INFO. Class values other than “01”, except for the class “IM”, indicate an error and are accompanied by a return code of SQL_ERROR. The class “IM” is specific to warnings and errors that derive from the implementation of ODBC itself. The subclass value “000” in any class is for implementation defined conditions within the given class. The assignment of class and subclass values is defined by ANSI SQL-92.

Note Although successful execution of a function is normally indicated by a return value of SQL_SUCCESS, the SQLSTATE 00000 also indicates success.

Table 1-17: SQLSTATE values that SQLError returns

SQLError	Returns SQLSTATE value
01000 General warning All ODBC functions except: SQLAllocEnv	SQLError
01002 Disconnect error SQLDisconnect	
01004 Data truncated SQLBrowseConnect	SQLColAttributes SQLDataSources SQLDescribeCol SQLDriverConnect SQLDrivers SQLExecDirect SQLExecute SQLExtendedFetch SQLFetch SQLGetCursorName SQLGetData SQLGetInfo SQLNativeSql SQLPutData SQLSetPos
01006 Privilege not revoked SQLExecDirect	SQLExecute
01S00 Invalid connection string attribute SQLBrowseConnect	SQLDriverConnect

SQLError	Returns SQLSTATE value
01S01 Error in row SQLExtendedFetch	SQLSetPos
01S02 Option value changed SQLSetConnectOption	SQLSetStmtOption
01S03 No rows updated or deleted SQLExecDirect	SQLExecute SQLSetPos
01S04 More than one row updated or deleted SQLExecDirect	SQLExecute SQLSetPos
07001 Wrong number of parameters SQLExecDirect	SQLExecute
07006 Restricted data type attribute violation SQLBindParameter	SQLExtendedFetch SQLFetch SQLGetData
08001 Unable to connect to data source SQLBrowseConnect	SQLConnect SQLDriverConnect
08002 Connection in use SQLBrowseConnect	SQLConnect SQLDriverConnect SQLSetConnectOption
08003 Connection not open SQLAllocStmt	SQLDisconnect SQLGetConnectOption SQLGetInfo SQLNativeSql SQLSetConnectOption SQLTransact
08004 Data source rejected establishment of connection SQLBrowseConnect	SQLConnect SQLDriverConnect
08007 Connection failure during transaction SQLTransact	

SQLError	Returns SQLSTATE value
08S01 Communication link failure SQLBrowseConnect	SQLColumnPrivileges SQLColumns SQLConnect SQLDriverConnect SQLExecDirect SQLExecute SQLExtendedFetch SQLFetch SQLForeignKeys SQLFreeConnect SQLGetData SQLGetTypeInfo SQLParamData SQLPrepare SQLPrimaryKeys SQLProcedureColumns SQLProcedures SQLPutData SQLSetConnectOption SQLSetStmtOption SQLSpecialColumns SQLStatistics SQLTablePrivileges SQLTables
21S01 Insert value list does not match column list SQLExecDirect	
21S02 Degree of derived table does not match column list SQLExecDirect	SQLPrepare SQLSetPos
22001 String data right truncation SQLPutData	
22003 Numeric value out of range SQLExecDirect	SQLExecute SQLExtendedFetch SQLFetch SQLGetData SQLGetInfo SQLPutData SQLSetPos
22005 Error in assignment SQLExecDirect	SQLExecute SQLGetData SQLPrepare SQLPutData SQLSetPos

SQLError	Returns SQLSTATE value
22008 Datetime field overflow SQLExecDirect	SQLExecute SQLGetData SQLPutData SQLSetPos
22012 Division by zero SQLExecDirect	SQLExecute SQLExtendedFetch SQLFetch
22026 String data, length mismatch SQLParamData	
23000 Integrity constraint violation SQLExecDirect	SQLExecute SQLSetPos
24000 Invalid cursor state SQLColAttributes	SQLColumnPrivileges SQLColumns SQLDescribeCol SQLExecDirect SQLExecute SQLExtendedFetch SQLFetch SQLForeignKeys SQLGetData SQLGetStmtOption SQLGetTypeInfo SQLPrepare SQLPrimaryKeys SQLProcedureColumns SQLProcedures SQLSetCursorName SQLSetPos SQLSetStmtOption SQLSpecialColumns SQLStatistics SQLTablePrivileges SQLTables
25000 Invalid transaction state SQLDisconnect	
28000 Invalid authorization specification SQLBrowseConnect	SQLConnect SQLDriverConnect
34000 Invalid cursor name SQLExecDirect	SQLPrepare SQLSetCursorName
37000 Syntax error or access violation SQLExecDirect	
3C000 Duplicate cursor name SQLSetCursorName	SQLNativeSql SQLPrepare

SQLError	Returns SQLSTATE value
40001 Serialization failure SQLExecDirect	SQLExecute SQLExtendedFetch SQLFetch
42000 Syntax error or access violation SQLExecDirect	SQLExecute SQLPrepare SQLSetPos
70100 Operation aborted SQLCancel	
IM001 Driver does not support this function All ODBC functions except:SQLAllocConnect	SQLAllocEnv SQLDataSources SQLDrivers SQLError SQLFreeConnect SQLFreeEnv SQLGetFunctions
IM002 Data source name not found and no default driver specified SQLBrowseConnect	SQLConnect SQLDriverConnect
IM003 Specified driver could not be loaded SQLBrowseConnect	SQLConnect SQLDriverConnect
IM004 Driver's SQLAllocEnv failed SQLBrowseConnect	SQLConnect SQLDriverConnect
IM005 Driver's SQLAllocConnect failed SQLBrowseConnect	SQLConnect SQLDriverConnect
IM006 Driver's SQLSetConnect-Option failed SQLBrowseConnect	SQLConnect SQLDriverConnect
IM007 No data source or driver specified; dialog prohibited SQLDriverConnect	
IM008 Dialog failed SQLDriverConnect	
IM009 Unable to load translation DLL SQLBrowseConnect	SQLConnect SQLDriverConnect SQLSetConnectOption
IM010 Data source name too long SQLBrowseConnect	SQLDriverConnect
IM011 Driver name too long SQLBrowseConnect	SQLDriverConnect
IM012 DRIVER keyword syntax error SQLBrowseConnect	SQLDriverConnect
IM013 Trace file error All ODBC functions.	
S0001 Base table or view already exists SQLExecDirect	SQLPrepare
S0002 Base table not found SQLExecDirect	SQLPrepare
S0011 Index already exists SQLExecDirect	SQLPrepare
S0012 Index not found SQLExecDirect	SQLPrepare
S0021 Column already exists SQLExecDirect	SQLPrepare
S0022 Column not found SQLExecDirect	SQLPrepare

SQLError	Returns SQLSTATE value
S0023 No default for column SQLSetPos	
S1000 General error All ODBC functions except:SQLAllocEnv	
S1001 Memory allocation failure All ODBC functions except:SQLAllocEnv	SQLError SQLFreeConnect SQLFreeEnv
S1002 Invalid column number SQLBindCol	SQLColAttributes SQLDescribeCol SQLExtendedFetch SQLFetch SQLGetData
S1003 Program type out of range SQLBindCol	SQLBindParameter SQLGetData
S1004 SQL data type out of range SQLBindParameter	SQLGetTypeInfo
S1008 Operation canceled All ODBC functions that can be processed	asynchronously:SQLColAttributes SQLColumnPrivileges SQLColumns SQLDescribeCol SQLDescribeParam SQLExecDirect SQLExecute SQLExtendedFetch SQLFetch SQLForeignKeys SQLGetData SQLGetTypeInfo SQLMoreResults SQLNumParams SQLNumResultCols SQLParamData SQLPrepare SQLPrimaryKeys SQLProcedureColumns SQLProcedures SQLPutData SQLSetPos SQLSpecialColumns SQLStatistics SQLTablePrivileges SQLTables

SQLError	Returns SQLSTATE value
`S1009 Invalid argument value SQLAllocConnect`	SQLAllocStmt SQLBindCol SQLBindParameter SQLExecDirect SQLForeignKeys SQLGetData SQLGetInfo SQLNativeSql SQLPrepare SQLPutData SQLSetConnectOption SQLSetCursorName SQLSetPos SQLSetStmtOption

SQLError	Returns SQLSTATE value
S1010 Function sequence error SQLBindCol	SQLBindParameter SQLColAttributes SQLColumnPrivileges SQLColumns SQLDescribeCol SQLDescribeParam SQLDisconnect SQLExecDirect SQLExecute SQLExtendedFetch SQLFetch SQLForeignKeys SQLFreeConnect SQLFreeEnv SQLFreeStmt SQLGetConnectOption SQLGetCursorName SQLGetData SQLGetFunctions SQLGetStmtOption SQLGetTypeInfo SQLMoreResults SQLNumParams SQLNumResultCols SQLParamData SQLParamOptions SQLPrepare SQLPrimaryKeys SQLProcedureColumns SQLProcedures SQLPutData SQLRowCount SQLSetConnectOption SQLSetCursorName SQLSetPos SQLSetScrollOptions SQLSetStmtOption SQLSpecialColumns SQLStatistics SQLTablePrivileges SQLTables SQLTransact
S1011 Operation invalid at this time SQLGetStmtOption	SQLSetConnectOption SQLSetStmtOption

SQLError	Returns SQLSTATE value
S1012 Invalid transaction operation code specified SQLTransact	
S1015 No cursor name available SQLGetCursorName	
S1090 Invalid string or buffer length SQLBindCol	SQLBindParameter SQLBrowseConnect SQLColAttributes SQLColumnPrivileges SQLColumns SQLConnect SQLDataSources SQLDescribeCol SQLDriverConnect SQLDrivers SQLExecDirect SQLExecute SQLForeignKeys SQLGetCursorName SQLGetData SQLGetInfo SQLNativeSql SQLPrepare SQLPrimaryKeys SQLProcedureColumns SQLProcedures SQLPutData SQLSetCursorName SQLSetPos SQLSpecialColumns SQLStatistics SQLTablePrivileges SQLTables
S1091 Descriptor type out of range SQLColAttributes	
S1092 Option type out of range SQLFreeStmt	SQLGetConnectOption SQLGetStmtOption SQLSetConnectOption SQLSetStmtOption
S1093 Invalid parameter number SQLBindParameter	SQLDescribeParam
S1094 Invalid scale value SQLBindParameter	
S1095 Function type out of range SQLGetFunctions	
S1096 Information type out of range SQLGetInfo	
S1097 Column type out of range SQLSpecialColumns	
S1098 Scope type out of range SQLSpecialColumns	

SQLError	Returns SQLSTATE value
S1099 Nullable type out of range SQLSpecialColumns	
S1100 Uniqueness option type out of range SQLStatistics	
S1101 Accuracy option type out of range SQLStatistics	
S1103 Direction option out of range SQLDataSources	SQLDrivers
S1104 Invalid precision value SQLBindParameter	
S1105 Invalid parameter type SQLBindParameter	
S1106 Fetch type out of range SQLExtendedFetch	
S1107 Row value out of range SQLExtendedFetch	SQLParamOptions SQLSetPos SQLSetScrollOptions
S1108 Concurrency option out of range SQLSetScrollOptions	
S1109 Invalid cursor position SQLExecute	SQLExecDirect SQLGetData SQLGetStmtOption SQLSetPos
S1110 Invalid driver completion SQLDriverConnect	
S1111 Invalid bookmark value SQLExtendedFetch	

SQLError	Returns SQLSTATE value
`S1C00 Driver not capable SQLBindCol`	SQLBindParameter SQLColAttributes SQLColumnPrivileges SQLColumns SQLExecDirect SQLExecute SQLExtendedFetch SQLFetch SQLForeignKeys SQLGetConnectOption SQLGetData SQLGetInfo SQLGetStmtOption SQLGetTypeInfo SQLPrimaryKeys SQLProcedureColumns SQLProcedures SQLSetConnectOption SQLSetPos SQLSetScrollOptions SQLSetStmtOption SQLSpecialColumns SQLStatistics SQLTablePrivileges SQLTables SQLTransact

SQLError	Returns SQLSTATE value
S1T00 Timeout expired SQLBrowseConnect	SQLColAttributes SQLColumnPrivileges SQLColumns SQLConnect SQLDescribeCol SQLDescribeParam SQLDriverConnect SQLExecDirect SQLExecute SQLExtendedFetch SQLFetch SQLForeignKeys SQLGetData SQLGetInfo SQLGetTypeInfo SQLMoreResults SQLNumParams SQLNumResultCols SQLParamData SQLPrepare SQLPrimaryKeys SQLProcedureColumns SQLProcedures SQLPutData SQLSetPos SQLSpecialColumns SQLStatistics SQLTablePrivileges SQLTables

CHAPTER 2

Adapter Configuration Files

About this chapter

This chapter describes how to create the configuration files specified in the Export Schema utility in ECMap.

Topics

This chapter contains the following topics:

Overview

The EDI product can be used as a plug-in adapter (called EDIadapter) with these core integration products:

- e-Biz 2000
- e-Biz Integrator
- MQSI

The configuration files for ECMap use two modes:

- Schema
- Schema_Remove

Configuration file samples

The following are sample configuration files.

Note The keys and values in the configuration files are case sensitive.

Schema Mode samples

The mode of the configuration file is Schema Mode. Schema Mode is used during design time to create formats for storing data structure definitions in a repository. The adapter plug-in library defines all schemas for the integration servers to use to parse and format messages. Each schema is equal to one New Data Object (NDO). Comments indicate the lines that must be added or on which information must be changed.

Sample configuration file for e-Biz 2000

This is a sample of a Schema Mode configuration file for e-Biz 2000, NNT41SchemaLoader, NNT51SchemaLoader, and how to remove a schema from the formatter. This file exports to e-Biz 2000.

```
Adapter
    adapter=schemamode
    mode=schema
```

```
    data=NDO

    # e-Biz 2000 schema load settings
Ebiz2K.Client.Name=
                    # Name of the machine where the client is setup, plus
                    # the name of the client
Ebiz2K.Message.Type=MessageType25
    SchemaLoader.Factory=eBiz2kSchemaLoader_Factory
    SchemaLoader.Library=adk33ebiz2ksl
    set.oob.options= false
ebiz2K.Outgoing_Schema = True
    in_file_name=ectree.txt
prefix=             # Prefix for the schema
    clash.avoid=true
    continue.format.exists=true
testinput
    file.delimiter=;
    default.nodedatatype=DT_String
    default.treedatatype=FALSE
    schemadata.file=ECTREE.TXT
    outputlog.Verbose=false
    outputlog.Warning=false
    outputlog.Error=true
    outputlog.file=ecmap.log
    outputerror.file=ecmap.err
    autoappendchild.maintree=true
    autoappendchild.maintree.error=true
    allow.same.name.dif.tree=true
```

Sample configuration file for Sybase Formatter 4.11,e-Biz Integrator, or MQSI

This configuration file exports to the Sybase Formatter 4.11, e-Biz Integrator, or MQSI.

```
Adapter
    clash.avoid=TRUE
    continue.format.exists=TRUE
    adapter=schemamode
    SchemaLoader.Factory=NNT41SchemaLoader_Factory    SchemaLoader.Library=
                    # Name of the library for the database    prefix=
                    # Prefix used for this schema
    mode=SCHEMA
    data=NDO
    repository.dir=C:\nnsy\NNSYContentRepository
    schema.key=ECMap    session.server=
            # Name of the server where database is located session.username=
```

```
            # User ID for the database    session.password=
            # Password for the database    session.database=
             # Name of the databasetestinput
    file.delimiter=;default.nodedatatype=DT_String
    default.treedatatype=FALSE
    schemadata.file=ECTREE.TXT
    outputlog.Verbose=false
    outputlog.Warning=false
    outputlog.Error=true
    outputlog.file=ecmap.log
    outputerror.file=ecmap.err
    autoappendchild.maintree=true
    autoappendchild.maintree.error=true
    allow.same.name.dif.tree=true
```

Sample configuration file for Sybase Formatter 5.1, e-Biz Integrator, and MQSI

This configuration file exports to Sybase Formatter 5.1, e-Biz Integrator, and MQSI.

```
Adapter
    clash.avoid=TRUE
    continue.format.exists=TRUE
    adapter=schemamodeSchemaLoader.Factory=NNT51SchemaLoader_Factory
    SchemaLoader.Library=adk33nnt51sl    prefix=
                    # Prefix for the schema
    mode=SCHEMA
    data=NDO    repository.dir=C:\nnsy\NNSYContentRepository
    schema.key=ECMap
    session=ECSchemaSession.ECSchema    NNOT_SHARED_LIBRARY=
            # Library needed to connect to database    NNOT_FACTORY_FUNCTION=
            # Function specific to each database    NN_SES_SERVER=
            # Name of the server    NN_SES_USER_ID=
            # User ID for the database    NN_SES_PASSWORD=
            # Password for the database    NN_SES_DB_NAME=
            # Name of the databasetestinput
    file.delimiter=;
    default.nodedatatype=DT_String
    default.treedatatype=FALSE
    schemadata.file=ECTREE.TXT
    outputlog.Verbose=false
    outputlog.Warning=false
    outputlog.Error=true
    outputlog.file=ecmap.log
    outputerror.file=ecmap.err
    autoappendchild.maintree=true
```

```
    autoappendchild.maintree.error=true
    allow.same.name.dif.tree=true
```

Schema_Remove Mode sample

This file allows schemas to be removed from the formatter.

```
Adapter
    clash.avoid=TRUE
    adapter=NNADKStubPlugIn
    SchemaLoader.Factory=NNT41SchemaLoader_Factory
    SchemaLoader.Library=
        # Name of the appropriate library for your database
    mode=SCHEMA_REMOVE
    data=NDO    prefix=
       # Prefix for the schema(s) you are going to remove    session.server=
       # Name of the server    session.username=
       # User ID for the database    session.password=
       # Password for the database    session.database=
       # Name of the database
    remove.by.prefix=true    remove.schema.keys=
        # Name(s) of the schema(s) you are going to remove
```

For more information about the EDI adapter, see *Adapter Runtime Environment for EDI User Guide*.

CHAPTER 3

Trading Partner and Log Database Formats

About this chapter

In ECMap, trading partner information is stored in the trading partner database, and transaction and error logging are stored in a log database. Both databases have an ODBC and non-ODBC version.

Topics

This chapter contains the following topics:

Overview

The trading partner database consists of three tables that contain information about the company, its trading partners, and the trade agreements that have been set up between them.

In the non-ODBC version, the company information is stored in an ASCII flat file, and the trading partner and trade agreement information are stored in Access tables and dBaseIII tables. (For map development, the program uses the data in Access tables, but at runtime the program uses the data in dBaseIII tables.)

In the ODBC version, the three tables can be stored in any ODBC-compliant database. The user must assign a data source name (DSN) that points to the trading partner database and use the appropriate ODBC driver for that specific database.

File	Description
wixset.dat	Contains company information for non-ODBC databases. See details in Table 3-1 on page 79.
wixset	Contains company information for ODBC databases. See details in Table 3-2 on page 83.
customer.dbf customer.mdb	Contains trading partner information for non-ODBC databases. See details in Table 3-3 on page 87.
	Contains trading partner information for ODBC databases. See details in Table 3-4 on page 96.
tradstat.dbf tradstat.mdb	Contains trade agreement information for non-ODBC databases. See details in Table 3-5 on page 109.
tradstat	Contains trade agreement information for ODBC databases. See details in Table 3-6 on page 114.

The log database contains information logged during the map execution. For non-ODBC users, transaction logging is written to one of two ASCII flat files—*translog.in* for inbound processing and *translog.out* for outbound processing. For ODBC users, transaction logging is written to one trlog table. The user must assign a data source name (DSN) that points to the log database and use the appropriate ODBC driver for that specific database.

File	Description
translog.in translog.out	Contains transaction logging for non-ODBC databases. See details in Table 3-7 on page 120.
trlog	Contains transaction logging for ODBC databases. See details in Table 3-8 on page 129.

The transaction log has one user-defined field (USERID_ENT) with a corresponding system variable (SYS_USER_FIELD), which can be used in any way the user wants.

For related information, see Chapter 4, "EDI Envelopes," the System Variables chapter in the *ECMap User Guide*, and the *ECRTP Reference Guide*.

This chapter contains the formats for each of the files and tables in the two databases. The following information is included in the description of each format: Field Number, Field Name, Type, Width/Precision, and Description. The Description contains a textual explanation of the field contents and where applicable, an explanation of the relationship between the field itself, other fields in the database, EDI envelope fields, and related system variables.

The program actions involving database fields differ for inbound and outbound processing. Some important processing actions are highlighted in the following sections. See "In outbound processing" and "In inbound processing" on page 78.

In outbound processing

I/O Rule

When the program performs the first I/O Rule associated with the first flow level, it increments the value in the isa_out_no field in the trading partner table and the value in the gs_no field in the trade agreement table and stores these values in memory. At this point, it loads information from the company file/table into WIX_* system variables and information from the trading partner database into the system variables that hold trading-partner and trade agreement information.

Before Rule

In the Before Rule associated with the first flow level, the user can assign values to system variables, overriding the system-assigned values and causing the program to use the user-assigned values instead. This gives the user control over envelope fields that are loaded from system variables; some fields are loaded from other program variables.

After Rule

Before performing the After Rule associated with the first flow level, the program populates the outbound EDI envelope fields with values from either system variables (user-assigned or system-assigned) or other program variables.

During processing

During processing – each time that the trading partner changes (regardless of whether the transaction changes), the program loads the interchange and group control numbers stored in memory into the appropriate database fields (ISA_OUT_NO and GS_NO) and increments the values stored in memory by 1.

Each time that the transaction changes (regardless of whether the trading partner changes), the program loads the group control number stored in memory into the GS_NO field in the trade agreement database and increments the value stored in memory by 1.

In inbound processing

Loading company data

The program loads information from the company file/table into the WIX_* system variables and information from the trading partner database into the system variables that hold trading-partner and trade agreement information.

Looking up the trading partner

The program performs a trading partner lookup to determine which map to run. Based on the search option chosen (switch selected/parameter used), the program looks for a match between the values in specific system variables (which were loaded from fields in the incoming envelope) and corresponding fields in the trading partner database.

Reading the ISA envelope

Each time the program reads a new ISA envelope, it updates the ISA_IN_NO field in the trading partner table in the trading partner database using the value from the following areas:

- Interchange Control Number (ISA 13) field in X12 envelopes
- Interchange Control Reference (UNB S004 0020) field in EDIFACT envelopes
- File Control ID (FHS 00091) field in HL7 envelopes

Reading the GS envelope

Each time it reads a new GS envelope, the program updates the GS_NO field in the trade agreement table in the trading partner database using the value from the following areas:

- Functional Group Header Control Number (GS 06) field in X12 envelopes
- Interchange Control Reference (UNG S004 00248 field in EDIFACT envelopes
- Batch Control ID (BHS 00091) field in HL7 envelopes

Database tables and logs

The following sections describe the database tables and logs.

Company table in non-ODBC trading partner database

The wixset.dat file is a fixed-length sequential file that contains one record.

Note With non-ODBC, the user cannot create multiple company profiles.

This file is created from the internal wixset table, which the program creates from the entries on the Company window, when the map runs. The location for wixset.dat is the trading partner directory, but for outbound processing you can override this directory location at runtime using the -dw <*directory*> switch.

Table 3-1: Company table in non-ODBC trading partner database

Name	Type	Width	Description
WIX_COMPANY_NAME	Character	35	Internal application name for the company. Inbound – Loads the contents of this field into the WIX_COMPANY_NAME system variable.
<filler>	Character	12	Not used.

WIX_GSID	Character	35	Main code used to identify the group level default sender on outbound maps – used only if the SND_GSID field in the trading partner file is blank.
			• Outbound– Loads the contents of this field into the WIX_GSID system variable. If the APP_SEND_CODE system variable is blank, the program loads WIX_GSID into APP_SEND_CODE, then loads APP_SEND_CODE into the following envelope fields:
			  • Application Sender's Code – GS 02
			  • Application Sender ID – UNG S006 0040
			  • Sending Application – MSH 00004
			  • File Sending Application – FHS 00070
			  • Batch Sending Application – BHS 00084
			• Inbound – Loads the contents of this field into the WIX_GSID system variable.

Name	Type	Width	Description

WIX_IDQUAL	Character	4	Qualifier that specifies the type of main code used to identify the interchange level default sender on outbound maps – used only if the SND_IDQUAL field in the trading partner file is blank.
			• Outbound – Loads the contents of this field into the WIX_IDQUAL system variable. If the SEND_QUAL system variable is blank, the program loads WIX_IDQUAL into SEND_QUAL, then loads SEND_QUAL into these envelope fields:
			  • Interchange Sender ID Qualifier – ISA 05
			  • Interchange Sender ID Code Qualifier – UNB S002 0007
			• Inbound – Loads the contents of this field into the WIX_IDQUAL system variable.

WIX_IDCODE	Character	35	Main code used to identify the interchange level default sender on outbound maps – used only if SND_IDCODE field in the trading partner file is blank.
			• Outbound – Loads the contents of this field into the WIX_IDCODE system variable. If the SEND_CODE system variable is blank, the program loads WIX_IDCODE into SEND_CODE, then loads SEND_CODE into the following envelope fields:
			  • Interchange Sender ID Code – ISA 06
			  • Interchange Sender ID – UNB S002 0004
			  • Sending Facility – MSH 00003
			  • File Sending Facility – FHS 00069
			  • Batch Sending Facility – BHS 00083
			• Inbound – Loads the contents of this field into the WIX_IDCODE system variable.

WIX_AUTH_QUAL	Character	2	Qualifier that specifies the type of code used to authenticate the company at the interchange level – used only if the AUTH_QUAL field in the trading partner file is blank. • Outbound – Loads the contents of this field into the WIX_AUTH_QUAL system variable. If the AUTH_QUAL system variable is blank, the program loads WIX_AUTH_QUAL into AUTH_QUAL, then loads AUTH_QUAL into this envelope field: Authorization Information Qualifier – ISA 01 • Inbound – Loads the contents of this field into the WIX_AUTH_QUAL system variable.

Name	Type	Width	Description
WIX_AUTH_CODE	Character	10	Code used to authenticate the company at the interchange level – used only if the AUTH_CODE field in the trading partner file is blank. • Outbound – Loads the contents of this field into the WIX_AUTH_CODE system variable. If the AUTH_CODE system variable is blank, the program loads WIX_AUTH_CODE into AUTH_CODE, then loads AUTH_CODE into this envelope field: Authorization Information Qualifier – ISA 02 • Inbound – Loads the contents of this field into the WIX_AUTH_CODE system variable.
WIX_SECU_QUAL	Character	2	Qualifier that specifies the type of code used to grant the company security clearance at the interchange level – used only if the SECU_QUAL field in the trading partner file is blank. • Outbound – Loads the contents of this field into the WIX_SECU_QUAL system variable. If the SECU_QUAL system variable is blank, the program loads WIX_SECU_QUAL into SECU_QUAL, then loads SECU_QUAL into the following envelope fields: • Security Information Qualifier – ISA 03 • Recipient Reference/Password Qualifier – UNB S005 0025 • Inbound – Loads the contents of this field into the WIX_SECU_QUAL system variable.
WIX_SECU_CODE	Character	10	Code used to grant the company security clearance at the interchange level – used only if the SECU_CODE field in the trading partner file is blank. • Outbound – Loads the contents of this field into the WIX_SECU_CODE system variable. If the SECU_CODE system variable is blank, the program loads WIX_SECU_CODE into SECU_CODE, then loads SECU_CODE into the following envelope fields: • Security Information – ISA 04 • Recipient Reference/Password – UNB S005 0022 • Inbound – Loads the contents of this field into the WIX_SECU_CODE system variable.

Name	Type	Width	Description
WIX_SNDR_ROUTE	Character	14 (35 for Syntax 4)	Internal code used to identify the interchange level default sender on outbound maps – used only if the SNDR_ROUTE field in the trading partner file is blank. • Outbound – Loads the contents of this field into the WIX_SNDR_ROUTE system variable. If the SEND_REV_ROUTE system variable is blank, the program loads WIX_SNDR_ROUTE into SEND_REV_ROUTE, then loads SEND_REV_ROUTE into this envelope field: Interchange Sender Internal ID – UNB S002 0008 • Inbound – Loads the contents of this field into the WIX_SNDR_ROUTE system variable.
WIX_SNDR_SUBID	Character	35	Internal sub-code used to identify the interchange level default sender on outbound maps – used only if the SNDR_SUBID field in the trading partner file is blank. (EDIFACT Syntax 4 only) • Outbound – Loads the contents of this field into the WIX_SNDR_SUBID system variable. If the SNDR_SUBID system variable is blank, the program loads WIX_SNDR_SUBID into SNDR_SUBID, then loads SNDR_SUBID into this envelope field: Interchange Sender Internal Sub-ID – UNB S002 0042 • Inbound – Loads the contents of this field into the WIX_SNDR_SUBID system variable.
WIX_APP_SND_QL	Character	4	Qualifier that specifies the type of main code used to identify the group level default sender on outbound maps – used only if the APP_SND_QL field in the trading partner file is blank. • Outbound – Loads the contents of this field into the WIX_APP_SND_QL system variable. If the APP_SEND_QUAL system variable is blank, the program loads WIX_APP_SND_QL into APP_SEND_QUAL, then loads APP_SEND_QUAL into this envelope field: Application Sender ID/ID Code Qualifier – UNG S006 0007 • Inbound – Loads the contents of this field into the WIX_APP_SND_QL system variable.

Company table in ODBC trading partner database

The wixset table is created from the entries on the Company window and can contain multiple company profiles, as shown in Table 3-2.

Table 3-2: Company table in ODBC trading partner database

Name	Type	Precision	Description
RECORD_NO	SQL_SMALLINT	4	Unique identifier used to create multiple profiles for the company.

GSID	SQL_VARCHAR	35	Main code used to identify the group level default sender on outbound maps – used only if the SND_GSID field in the trading partner table is blank.
			• Outbound – Loads the contents of this field into the WIX_IDQUAL system variable. If the APP_SEND_CODE system variable is blank, the program loads WIX_IDQUAL into APP_SEND_CODE, then loads APP_SEND_CODE into the following envelope fields:
			  • Application Sender's Code – GS 02
			  • Application Sender ID – UNG S006 0040
			  • Sending Application – MSH 00004
			  • File Sending Application – FHS 00070
			  • Batch Sending Application – BHS 00084
			• Inbound – Loads the contents of this field into the WIX_IDQUAL system variable.

NAME	SQL_VARCHAR	35	Internal application name for the company. Inbound – Loads the contents of this field into the WIX_COMPANY_NAME system variable.

IDQUAL	SQL_VARCHAR	4	Qualifier that specifies the type of main code used to identify the interchange level default sender on outbound maps – used only if the SND_IDQUAL field in the trading partner table is blank.
			• Outbound – Loads the contents of this field into the WIX_IDQUAL system variable. If the SEND_QUAL system variable is blank, the program loads WIX_IDQUAL into SEND_QUAL, then loads SEND_QUAL into the following envelope fields:
			  • Interchange Sender ID Qualifier – ISA 05
			  • Interchange Sender ID Code Qualifier – UNB S002 0007
			• Inbound – Loads the contents of this field into the WIX_IDQUAL system variable.

Name	Type	Precision	Description

IDCODE	SQL_VARCHAR	35	Main code used to identify the interchange level default sender on outbound maps – used only if the SND_IDCODE field in the trading partner table is blank.
			• Outbound – Loads the contents of this field into the WIX_IDCODE system variable. If the SEND_CODE system variable is blank, the program loads WIX_IDCODE into SEND_CODE, then loads SEND_CODE into the following envelope fields:
			  • Interchange Sender ID Code – ISA 06
			  • Interchange Sender ID – UNB S002 0004
			  • Sending Facility – MSH 00003
			  • File Sending Facility – FHS 00069
			  • Batch Sending Facility – BHS 00083
			• Inbound – Loads the contents of this field into the WIX_IDCODE system variable.

AUTH_QUAL	SQL_VARCHAR	2	Qualifier that specifies the type of code used to authenticate the company at the interchange level – used only if the AUTH_QUAL field in the trading partner table is blank. • Outbound – Loads the contents of this field into the WIX_AUTH_QUAL system variable. If the AUTH_QUAL system variable is blank, the program loads WIX_AUTH_QUAL into AUTH_QUAL, then loads AUTH_QUAL into this envelope field: Authorization Information Qualifier – ISA 01 • Inbound – Loads the contents of this field into the WIX_AUTH_QUAL system variable.
AUTH_CODE	SQL_VARCHAR	10	Code used to authenticate the company at the interchange level, used only if the AUTH_CODE field in the trading partner table is blank. • Outbound – Loads the contents of this field into the WIX_AUTH_CODE system variable. If the AUTH_CODE system variable is blank, the program loads WIX_AUTH_CODE into AUTH_CODE, then loads auth_code into this envelope field: Authorization Information Qualifier – ISA 02 • Inbound – Loads the contents of this field into the WIX_AUTH_CODE system variable.

Name	Type	Precision	Description

SECU_QUAL	SQL_VARCHAR	2	Qualifier that specifies the type of code used to grant the company security clearance at the interchange level – used only if the SECU_QUAL field in the trading partner table is blank.
			• Outbound – Loads the contents of this field into the WIX_SECU_QUAL system variable. If the SECU_QUAL system variable is blank, the program loads WIX_SECU_QUAL into SECU_QUAL, then loads SECU_QUAL into the following envelope fields:
			  • Security Information Qualifier – ISA 03
			  • Recipient Reference/Password Qualifier –UNB S005 0025
			• Inbound – Loads the contents of this field into the WIX_SECU_QUAL system variable.

SECU_CODE	SQL_VARCHAR	10	Code used to grant the company security clearance at the interchange level – used only if the SECU_CODE field in the trading partner table is blank.
			• Outbound – Loads the contents of this field into the WIX_SECU_CODE. If the SECU_CODE system variable is blank, the program loads WIX_SECU_CODE into SECU_CODE, then loads secu_code into the following envelope fields:
			  • Security Information – ISA 04
			  • Recipient Reference/Password – UNB S005 0022
			• Inbound – Loads the contents of this field into the WIX_SECU_CODE system variable.

SNDR_ROUTE	SQL_VARCHAR	14 (35 for Syntax 4)	Internal code used to identify the interchange level default sender on outbound maps – used only if the SNDR_ROUTE field in the trading partner table is blank. • Outbound – Loads the contents of this field into the WIX_SNDR_ROUTE system variable. If the SEND_REV_ROUTE system variable is blank, the program loads WIX_SNDR_ROUTE into SEND_REV_ROUTE, then loads SEND_REV_ROUTE into this envelope field: Interchange Sender Internal ID – UNB S002 0008 • Inbound – Loads the contents of this field into the WIX_SNDR_ROUTE system variable.

Name	Type	Precision	Description
SNDR_SUBID	SQL_VARCHAR	35	Internal sub-code used to identify the interchange level default sender on outbound maps – used only if the SNDR_SUBID field in the trading partner table is blank. (EDIFACT Syntax 4 only) • Outbound – Loads the contents of this field into the WIX_SNDR_SUBID system variable. If the SNDR_SUBID system variable is blank, the program loads WIX_SNDR_SUBID into SNDR_SUBID, then loads SNDR_SUBID into this envelope field: Interchange Sender Internal Sub-ID – UNB S002 0042 • Inbound – Loads the contents of this field into the WIX_SNDR_SUBID system variable.
APP_SND_QL	SQL_VARCHAR	4	Qualifier that specifies the type of main code used to identify the group level default sender on outbound maps – used only if the APP_SND_QL field in the trading partner table is blank. • Outbound – Loads the contents of this field into the WIX_APP_SND_QL system variable. If the APP_SEND_QUAL system variable is blank, the program loads WIX_APP_SND_QL into APP_SEND_QUAL and then loads APP_SEND_QUAL into this envelope field: Application Sender ID/ID Code Qualifier – UNG S006 0007 • Inbound – Loads the contents of this field into the WIX_APP_SND_QL system variable.
B_SEND_ID	SQL_VARCHAR	24	Batch sender ID
BIN_NUMB	SQL_SMALLINT	6	Bin number
PROC_NUMB	SQL_VARCHAR	10	Processing control number
SERV_QUAL	SQL_VARCHAR	2	Service provider ID qualifier
SERV_ID	SQL_VARCHAR	15	Service provider ID
SOFT_ID	SQL_VARCHAR	10	Software/vendor ID

Trading partner file in non-ODBC trading partner database

The *customer.dbf* file contains trading partner information for non-ODBC databases as shown in Table 3-3.

Table 3-3: Trading partner file in non-ODBC trading partner database

Name	Type	Width	Description
CUSTNO	Character	35	Internal application identifier for the trading partner, used to link the trading partner table (*customer.dbf*), trade agreement (*tradstat.dbf*) table, and the application data field that has the attribute "Trading Partner ID".
<filler>	Numeric	1	No longer used (formerly TYPE_OWNER).
NAME	Character	35	Internal application name for the trading partner.
IDQUAL	Character	4	Qualifier that specifies the type of main code used to identify the interchange level default receiver on outbound maps. (The value in this field is overridden by a value in the RCV_IDQUAL field of the trade agreement table for outbound processing.) Outbound – Loads the contents of this field into the RECV_EQUAL system variable, then loads RECV_EQUAL into these envelope fields: • Interchange Receiver ID Qualifier – ISA 07 • Interchange Recipient ID Code Qualifier – UNB S003 0007
IDCODE	Character	35	Main code used to identify the interchange level default receiver on outbound maps. The value in this field is overridden by a value in the RCV_IDCODE field of the trade agreement table for outbound processing. Outbound – Loads the contents of this field into the RECV_CODE system variable, then loads RECV_CODE into the following envelope fields: • Interchange Receiver ID Code – ISA 08 • Interchange Receiver ID – UNB S003 0010 • Receiving Facility – MSH 00005 • File Receiving Facility – FHS 00071 • Batch Receiving Facility – BHS 00085
AUTH_QUAL	Character	2	Qualifier that specifies the type of code used to authenticate the trading partner at the interchange level. If this field is blank, the program uses the value in the WIX_AUTH_QUAL field of the company file for outbound processing. Outbound – Loads the contents of this field into the AUTH_QUAL system variable, then loads AUTH_QUAL into this envelope field: Authorization Information Qualifier – ISA 01

Name	Type	Width	Description
AUTH_CODE	Character	10	Code used to authenticate the trading partner at the interchange level. If this field is blank, the program uses the value in the WIX_AUTH_CODE field of the company file for outbound processing. Outbound – Loads the contents of this field into the AUTH_CODE system variable, then loads AUTH_CODE into the following envelope fields: • Authorization Information – ISA 02 • Application Password – UNG S008 0058
SECU_QUAL	Character	2	Qualifier that specifies the type of code used to grant security clearance to the trading partner at the interchange level. If this field is blank, the program uses the value in the WIX_SECU_QUAL field of the company file for outbound processing. Outbound – Loads the contents of this field into the SECU_QUAL system variable, then loads SECU_QUAL into the following envelope fields: • Security Information Qualifier – ISA 03 • Recipient Reference/Password Qualifier – UNB S005 0025
SECU_CODE	Character	10	Code used to grant security clearance to the trading partner at the interchange level. If this field is blank, the program uses the value in the WIX_SECU_CODE field of the company file for outbound processing. Outbound – Loads the contents of this field into the SECU_CODE system variable, then loads SECU_CODE into these envelope fields: • Security Information – ISA 04 • Recipient Reference/Password – UNB S005 0022

Name	Type	Width	Description

GSID	Character	35	Main code used to identify the group level default receiver on outbound maps and the group sender lookup value on inbound maps. The value in this field is overridden by a value in the RCV_GSID field of the trade agreement table for outbound processing.
			• Outbound – Loads the contents of this field into the APP_RECV_CODE system variable, then loads APP_RECV_CODE into the following envelope fields:
			  • Application Receiver's Code – GS 03
			  • Application Recipient ID – UNG S007 0044
			  • Receiving Application – MSH 00006
			  • File Receiving Application – FHS 00072
			  • Batch Receiving Application – BHS 00086
			• Inbound – When this field is part of the trading partner lookup, the program compares the contents of this field with the contents of the APP_SEND_CODE system variable, loaded from the following envelope fields:
			  • Application Sender's Code – GS 02
			  • Application Sender ID – UNG S006 0040
			  • Sending Application – MSH 00004
			  • File Sending Application – FHS 00070
			  • Batch Sending Application – BHS 00084

SHIPQUAL	Character	2	Qualifier that specifies Ship To identification code of the trading partner. Outbound – Loads the contents of this field into the SYS_SHIPQUAL system variable, where it is available for use in mapping.
SHIPIDEN	Character	15	Ship To identification code of the trading partner. Outbound – Loads the contents of this field into the SYS_SHIPIDEN system variable, where it is available for use in mapping.
BILLQUAL	Character	2	Qualifier that specifies Bill To identification code of the trading partner. Outbound – Loads the contents of this field into the SYS_BILLQUAL system variable, where it is available for use in mapping.
BILLIDEN	Character	15	Bill To identification code of the trading partner. Outbound – Loads the contents of this field into the SYS_BILLIDEN system variable, where it is available for use in mapping.
ADDR1	Character	35	Street address at which the trading partner is located. Outbound – Loads the contents of this field into the SYS_ADDR1 system variable, where it is available for use in mapping.

Name	Type	Width	Description
ADDR2	Character	35	Additional street address at which the trading partner is located. Outbound – Loads the contents of this field into the SYS_ADDR2 system variable, where it is available for use in mapping.
CITY	Character	19	City in which the trading partner is located. Outbound – Loads the contents of this field into the SYS_CITY system variable, where it is available for use in mapping.
STATE	Character	15	State in which the trading partner is located. Outbound – Loads the contents of this field into the SYS_STATE system variable, where it is available for use in mapping.
COUNTRY	Character	25	Country in which the trading partner is located. Outbound – Loads the contents of this field into the SYS_COUNTRY system variable, where it is available for use in mapping.
ZIP	Character	9	Zip code at which the trading partner is located. Outbound – Loads the contents of this field into the SYS_ZIP system variable, where it is available for use in mapping.
CONTACT1	Character	35	Name of the trading partner contact. Outbound – Loads the contents of this field into the CONTACT1 system variable, where it is available for use in mapping.
TELEPHONE1	Character	22	Telephone number of the trading partner contact. Outbound – Loads the contents of this field into the TELEPHONE1 system variable, where it is available for use in mapping.
CONTACT2	Character	35	Name of an additional trading partner contact. Outbound – Loads the contents of this field into the CONTACT2 system variable, where it is available for use in mapping.
TELEPHONE2	Character	22	Telephone number of an additional trading partner contact. Outbound – Loads the contents of this field into the TELEPHONE2 system variable, where it is available for use in mapping.

Name	Type	Width	Description
ISA_IN_NO	Character	14	Interchange-level control reference number for inbound processing. Inbound – To eliminate unnecessary processing time, the program updates this field only when the trading partner changes. The program reads the incoming EDI envelope, performs a trading partner lookup to select the map to run (using criteria specified by the user), writes an entry to the log, then the contents of these envelope fields into the INT_HEAD_NUM system variable: • Interchange Control Number – ISA 13 • Interchange Control Reference – UNB S004 0020 • File Control ID – FHS 00077 The program also stores the contents of the INT_HEAD_NUM system variable in an internal storage location at this time. When the trading partner changes, the program loads the contents of the internal storage location into this field. Users can load an override value into this field on the General tab of the Trading Partner window.
ISA_OUT_NO	Character	14	Interchange-level control reference number for outbound processing. Outbound – To eliminate unnecessary updates during processing, the program updates the value in this field initially and then only when the trading partner changes. The program performs the internal update when it looks for a trade agreement match to select the map to be run. When it finds a match, it increments the value in this field by 1, loads the contents of the field into the INT_HEAD_NUM system variable, then loads INT_HEAD_NUM into these envelope fields: • Interchange Control Number – ISA 13 • Interchange Control Reference – UNB S004 0020 • File Control ID – FHS 00077 Whenever the trading partner changes during processing, the program again increments the value in this field by 1, loads the contents of the field into the INT_HEAD_NUM system variable, and loads INT_HEAD_NUM into the appropriate envelope field. Users can load an override value into this field on the General tab of the Trading Partner window.

Name	Type	Width	Description

SND_GSID	Character	35	Main code used to identify the group level override receiver on outbound maps and the group receiver lookup value on inbound maps. If this field is blank, the program uses the value in the WIX_IDQUAL field of the company file for outbound processing.
			• Outbound – Loads the contents of this field into the APP_SEND_CODE system variable, then loads APP_SEND_CODE into the following envelope fields:
			  • Application Sender's Code – GS 02
			  • Application Sender ID – UNG S006 0040
			  • Sending Application – MSH 00004
			  • File Sending Application – FHS 00070
			  • Batch Sending Application – BHS 00084
			• Inbound – When this field is part of the trading partner lookup, the program compares the contents of this field with the contents of the APP_RECV_CODE system variable, loaded from the following envelope fields:
			  • Application Receiver's Code – GS 03
			  • Application Recipient ID – UNG S007 0044
			  • Receiving Application – MSH 00006
			  • File Receiving Application – FHS 00072
			  • Batch Receiving Application – BHS 00086

SND_IDQUAL	Character	4	Qualifier that specifies the type of main code used to identify the interchange level override sender on outbound maps. If this field is blank, the program uses the value in the WIX_IDQUAL field of the company file for outbound processing. Outbound – Loads the contents of this field into the SEND_QUAL system variable, then loads SEND_QUAL into the following envelope fields: • Interchange Sender ID Qualifier – ISA 05 • Interchange Sender ID Code Qualifier – UNB S002 0007

Name	Type	Width	Description
SND_IDCODE	Character	35	Main code used to identify the interchange level override sender on outbound maps. (If this field is blank, the program uses the value in the WIX_IDCODE field of the company file for outbound processing.) Outbound – Loads the contents of this field into the SEND_CODE system variable, then loads SEND_CODE into the following envelope fields: • Interchange Sender ID Code – ISA 06 • Interchange ID – UNB S002 0004 • Sending Facility – MSH 00003 • File Sending Facility – FHS 00069 • Batch Sending Facility – BHS 00083
SUB_DELIMT	Character	3	Special character used by the trading partner to override the default X12 sub-element delimiter. Outbound – If there is a value in this field, the program uses it to create outbound X12 data. If not, it uses a default value.
ELE_DELIMT	Character	3	Special character used by the trading partner to override the default X12 element delimiter. Outbound – If there is a value in this field, the program uses it to create outbound X12 data. If not, it uses a default value.
SEG_DELIMT	Character	3	Special character used by the trading partner to override the default X12 segment delimiter. Outbound – If there is a value in this field, the program uses it to create outbound X12 data. If not, it uses a default value.
RELEASE_CH	Character	3	Special character used by the trading partner to override the default X12 release character. Outbound – If there is a value in this field, the program uses it to create outbound X12 data. If not, it uses a default value.
X12_REPEATS	Character	3	Special character used by the trading partner to override the default X12 repeat character. Outbound – If there is a value in this field, the program uses it to create outbound X12 data. If not, it uses a default value.
<filler>	Character	1	No longer used. (formerly DEL_CODE)
EDIF_SUBDL	Character	3	Special character used by the trading partner to override the default EDIFACT sub-element delimiter. Outbound – If there is a value in this field, the program uses it to create outbound EDIFACT data. If not, it uses a default value.

Name	Type	Width	Description
EDIF_ELEDL	Character	3	Special character used by the trading partner to override the default EDIFACT element delimiter. Outbound – If there is a value in this field, the program uses it to create outbound EDIFACT data. If not, it uses a default value.
EDIF_SEGDL	Character	3	Special character used by the trading partner to override the default EDIFACT segment delimiter. Outbound If there is a value in this field, the program uses it to create outbound EDIFACT data. If not, it uses a default value.
EDIF_RELCH	Character	3	Special character used by the trading partner to override the default EDIFACT release character. Outbound – If there is a value in this field, the program uses it to create outbound EDIFACT data. If not, it uses a default value.
EDIF_REPEA	Character	3	Special character used by the trading partner to override the default EDIFACT repeat character. Outbound – If there is a value in this field, the program uses it to create outbound EDIFACT data. If not, it uses a default value.
HL7_SEGDL	Character	3	Special character used by the trading partner to override the default HL7 segment delimiter. Outbound – If there is a value in this field, the program uses it to create outbound HL7 data. If not, it uses a default value.
HL7_ELEDL	Character	3	Special character used by the trading partner to override the default HL7 element delimiter. Outbound – If there is a value in this field, the program uses it to create outbound HL7 data. If not, it uses a default value.
HL7_SUBDL	Character	3	Special character used by the trading partner to override the default HL7 sub-element delimiter. Outbound – If there is a value in this field, the program uses it to create outbound HL7 data. If not, it uses a default value.
HL7_SUBSUB	Character	3	Special character used by the trading partner to override the default HL7 component delimiter. Outbound – If there is a value in this field, the program uses it to create outbound HL7 data. If not, it uses a default value.
HL7_RELCH	Character	3	Special character used by the trading partner to override the default HL7 release character. Outbound – If there is a value in this field, the program uses it to create outbound HL7 data. If not, it uses a default value.
HL7_REPEAT	Character	3	Special character used by the trading partner to override the default HL7 repeat character. Outbound – If there is a value in this field, the program uses it to create outbound HL7 data. If not, it uses a default value.

Name	Type	Width	Description
EXPORT_FLAG	Character	1	Special character used to designate that flagged trading partner records can be moved from one database to another.
MBOX_NAME	Character	35	Internal name of the trading partner mailbox - used only as a label on windows or reports. The value in this field can be overridden by a value in the MBOX_NAME field in the trade agreement table.
MAILBOX	Character	100	Full-path name of the trading partner mailbox folder, or directory. The value in this field can be overridden by a value in the dest field in the trade agreement table.
CURR_FMT	Character	1	Character used to indicate whether a period or comma is used as the decimal character • C – Comma decimal character • D – Period decimal character
POS_LTR	Character	1	Reserved for future use with packed decimal.
SNDR_ROUTE	Character	14 (35 for Syntax 4)	Internal code used to identify the interchange level override sender on outbound maps. Outbound – Loads the contents of this field into the SEND_REV_ROUTE system variable, then loads SEND_REV_ROUTE into this envelope field: Interchange Sender Internal ID – UNB S002 0008
SNDR_SUBID	Character	35	Internal sub-code used to identify the interchange level override sender on outbound maps. (EDIFACT Syntax 4 only) Outbound – Loads the contents of this field into the SNDR_SUBID system variable, then loads SNDR_SUBID into this envelope field: Interchange Sender Internal Sub-ID – UNB S002 0042
RCVR_ROUTE	Character	14 (35 for Syntax 4)	Internal code used to identify the interchange level default receiver on outbound maps. Outbound – Loads the contents of this field into the RCVR_ROUTE system variable, then loads RCVR_ROUTE into this envelope field: Interchange Receiver Internal ID – UNB S003 0014
RCVR_SUBID	Character	35	Internal sub-code used to identify the interchange level default receiver on outbound maps. (EDIFACT Syntax 4 only) Outbound – Loads the contents of this field into the RCVR_SUBID system variable, then loads RCVR_SUBID into this envelope field: Interchange Receiver Internal Sub-ID – UNB S003 0046

Name	Type	Width	Description
APP_SND_QL	Character	4	Qualifier that specifies the type of main code used to identify the group level override sender on outbound maps. (If this field is blank, the program uses the value in the WIX_APP_SND_QL field of the company file for outbound processing.) Outbound – Loads the contents of this field into the APP_SEND_QUAL system variable, then loads APP_SEND_QUAL into this envelope field: Application Sender ID/ID Code Qualifier – UNG S006 0007
APP_RCV_QL	Character	4	Qualifier that specifies the type of main code used to identify the group level default receiver on outbound maps. (The value in this field will be overridden by a value in the app_rcv_ql field of the trade agreement table for outbound processing.) Outbound – Loads the contents of this field into the APP_RECV_QUAL system variable, then loads APP_RECV_QUAL into this envelope field: Application Recipient ID/ID Code Qualifier – UNG S007 0007
B_SEND_ID	Character	24	Batch sender ID
BIN_NUMB	Integer	6	Bin number
PROC_NUMB	Character	10	Processing control number
SERV_QUAL	Character	2	Service provider ID qualifier
SERV_ID	Character	15	Service provider ID
SOFT_ID	Character	10	Software/vendor ID
ACK_TYPE	Character	1	Acknowledgement flag
TPKEY	Integer	10	Unique auto-increment field used to: Update the ISA_IN_NO and ISA_OUT_NO control numbers Prevent simultaneous update of the trading partner database by multiple users

Trading partner table in ODBC trading partner database

The *tp* file contains trading partner information for ODBC databases as illustrated in Table 3-4.

Table 3-4: Trading partner table in ODBC trading partner database

Name	Type	Precision	Description
CUSTNO	SQL_VARCHAR	35	Internal application identifier for the trading partner, used to link the trading partner table (tp), trade agreement (tradstat) table, and the application data field that has the attribute *Trading Partner* ID

Name	Type	Precision	Description
<filler>	SQL_VARCHAR	1	No longer used (formerly TYPE_OWNER)
NAME	SQL_VARCHAR	35	Internal application name for the trading partner

IDQUAL	SQL_VARCHAR	4	Qualifier that specifies the type of main code used to identify the interchange level default receiver on outbound maps and the interchange sender lookup value on inbound maps. (The value in this field will be overridden by a value in the RCV_IDQUAL field of the trade agreement table for outbound processing.)
			• Outbound – Loads the contents of this field into the RECV_QUAL system variable, then loads RECV_QUAL into the following envelope fields:
			  • Interchange Receiver ID Qualifier – ISA 07
			  • Interchange Recipient ID Code Qualifier –UNB S003 0007
			• Inbound – Loads the contents of this field into the SEND_QUAL system variable. When this field is used as part of the trading partner lookup, the program compares the contents of SEND_QUAL with the contents of the following envelope fields:
			  • Interchange Sender ID Qualifier – ISA 05
			  • Interchange Sender ID Code Qualifier – UNB S002 0007

IDCODE	SQL_VARCHAR	35	Main code used to identify the interchange level default receiver on outbound maps and default sender on inbound maps. The value in this field is overridden by a value in the RCV_IDCODE field of the trade agreement table for outbound processing. Outbound – Loads the contents of this field into the RECV_CODE system variable, then loads RECV_CODE into the following envelope fields: • Interchange Receiver ID Code – ISA 08 • Interchange Receiver ID–UNB S003 0010 • Receiving Facility – MSH 00005 • File Receiving Facility – FHS 00071 • Batch Receiving Facility – BHS 00085

Name	Type	Precision	Description
IDCODE	SQL_VARCHAR	35	Inbound – When this field is part of the trading partner lookup, the program compares the contents of this field with the contents of the SEND_CODE system variable loaded from these envelope fields: • Interchange Sender ID Code – ISA 06 • Interchange Sender ID – UNB S002 0004 • Sending Facility – MSH 00003 • File Sending Facility – FHS 00069 • Batch Sending Facility – BHS 00083
AUTH_QUAL	SQL_VARCHAR	2	Qualifier that specifies the type of code used to authenticate the trading partner at the interchange level. If this field is blank, the program uses the value in the WIX_AUTH_QUAL field in the company table for outbound processing. Outbound – Loads the contents of this field into the AUTH_QUAL system variable, then loads AUTH_QUAL into this envelope field: Authorization Information Qualifier – ISA 01
AUTH_CODE	SQL_VARCHAR	10	Code used to authenticate the trading partner at the interchange level. If this field is blank, the program uses the value in the WIX_AUTH_CODE field in the company table for outbound processing. Outbound – Loads the contents of this field into the AUTH_CODE system variable, then loads AUTH_CODE into the following envelope fields: • Authorization Information – ISA 02 • Application Password – UNG S005 0058
SECU_QUAL	SQL_VARCHAR	2	Qualifier that specifies the type of code used to grant security clearance to the trading partner at the interchange level. If this field is blank, the program uses the value in the WIX_SECU_QUAL field in the company table for outbound processing. Outbound – Loads the contents of this field into the SECU_QUAL system variable, then loads SECU_QUAL into these envelope fields: • Security Information Qualifier – ISA 03 • Recipient Reference/Password Qualifier – UNB S005 0025

Name	Type	Precision	Description
SECU_CODE	SQL_VARCHAR	10	Code used to grant security clearance to the trading partner at the interchange level. If this field is blank, the program uses the value in WIX_SECU_CODE field in the company table for outbound processing. Outbound – Loads the contents of this field into the SECU_CODE system variable, then loads SECU_CODE into these envelope fields: • Security Information – ISA 04 • Recipient Reference/Password – UNB S005 0022

GSID	SQL_VARCHAR	35	Main code used to identify the group level default receiver on outbound maps and default sender on inbound maps.
			• Outbound – Loads the contents of this field into the APP_RECV_CODE system variable, then loads APP_RECV_CODE into the following envelope fields:
			  • Application Receiver's Code – GS 03
			  • Application Recipient ID – UNG S007 0044
			  • Receiving Application – MSH 00006
			  • File Receiving Application – FHS 00072
			  • Batch Receiving Application – BHS 00086
			• Inbound – When this field is part of the trading partner lookup, the program compares contents of this field with the contents of the APP_SEND_CODE system variable, loaded from the following envelope fields:
			  • Application Sender's Code – GS 02
			  • Application Sender ID - UNG S006 0040
			  • Sending Application – MSH 00004
			  • File Sending Application – FHS 00070
			  • Batch Sending Application – BHS 00084

SHIPQUAL	SQL_VARCHAR	2	Qualifier that specifies Ship To identification code of the trading partner. Outbound – Loads the contents of this field into the SYS_SHIPQUAL system variable, where it is available for use in mapping.
SHIPIDEN	SQL_VARCHAR	15	Ship To identification code of the trading partner. Outbound – Loads the contents of this field into the SYS_SHIPIDEN system variable, where it is available for use in mapping.

Name	Type	Precision	Description
BILLQUAL	SQL_VARCHAR	2	Qualifier that specifies Bill To identification code of the trading partner. Outbound – Loads the contents of this field into the SYS_BILLQUAL system variable, where it is available for use in mapping.
BILLIDEN	SQL_VARCHAR	15	Bill To identification code of the trading partner. Outbound – Loads the contents of this field into the SYS_BILLIDEN system variable, where it is available for use in mapping.
ADDR1	SQL_VARCHAR	35	Street address at which the trading partner is located. Outbound – Loads the contents of this field into the SYS_ADDR1 system variable, where it is available for use in mapping.
ADDR2	SQL_VARCHAR	35	Additional street address at which the trading partner is located. Outbound – Loads the contents of this field into the SYS_ADDR2 system variable, where it is available for use in mapping.
CITY	SQL_VARCHAR	19	City in which the trading partner is located. Outbound – Loads the contents of this field into the SYS_CITY system variable, where it is available for use in mapping.
STATE	SQL_VARCHAR	15	State in which the trading partner is located. Outbound – Loads the contents of this field into the SYS_STATE system variable, where it is available for use in mapping.
COUNTRY	SQL_VARCHAR	25	Country in which the trading partner is located. Outbound – Loads the contents of this field into the SYS_COUNTRY system variable, where it is available for use in mapping.
ZIP	SQL_VARCHAR	9	Zip code at which the trading partner is located. Outbound – Loads the contents of this field into the SYS_ZIP system variable, where it is available for use in mapping.
CONTACT1	SQL_VARCHAR	35	Name of the trading partner contact. Outbound – Loads the contents of this field into the CONTACT1 system variable, where it is available for use in mapping.

Name	Type	Precision	Description
TELEPHONE1	SQL_VARCHAR	22	Telephone number of the trading partner contact. Outbound – Loads the contents of this field into the TELEPHONE1 system variable, where it is available for use in mapping.
CONTACT2	SQL_VARCHAR	35	Name of an additional trading partner contact. Outbound – Loads the contents of this field into the CONTACT2 system variable, where it is available for use in mapping.
TELEPHONE2	SQL_VARCHAR	22	Telephone number of an additional trading partner contact. Outbound – Loads the contents of this field into the TELEPHONE2 system variable, where it is available for use in mapping.
ISA_IN_NO	SQL_VARCHAR	14	Interchange-level control reference number for inbound processing. Inbound – The program reads the incoming EDI envelope, performs a trading partner lookup to select the map to be run (using criteria specified by the user), writes an entry to the log, then the contents of these envelope fields into the INT_HEAD_NUM system variable: • Interchange Control Number – ISA 13 • Interchange Control Reference –UNB S004 0020 • File Control ID – FHS 0077 The program then also stores the contents of INT_HEAD_NUM in an internal storage area. When the trading partner changes, the program loads the contents of the internal storage location into this field. Users can load an override value into this field on the Mailbox tab of the Trading Partner window.

Name	Type	Precision	Description
ISA_OUT_NO	SQL_VARCHAR	14	Interchange-level control reference number for outbound processing. Outbound – The program first looks for a trade agreement match to select the map to be run. Once it finds a match, it increments the value in this field by 1, loads the field contents into the INT_HEAD_NUM system variable, then loads INT_HEAD_NUM into these envelope fields: • Interchange Control Number – ISA 13 • Interchange Control Reference–UNB S004 0020 • File Control ID – FHS 0077 When the trading partner changes during processing, the program increments the value in this field by 1, loads the contents of the field into the INT_HEAD_NUM system variable, and loads INT_HEAD_NUM into the appropriate envelope field. Users can load an override value into this field on the Mailbox tab of the Trading Partner window.

SND_GSID	SQL_VARCHAR	35	Main code used to identify the group level override sender on outbound maps and group receiver lookup value on inbound maps. If this field is blank, the program uses the value in the WIX_IDQUAL field in the company table for outbound processing.
			• Outbound – Loads the contents of this field into the APP_SEND_CODE, then loads APP_SEND_CODE into the following envelope fields:
			  • Application Sender's Code – GS 02
			  • Application Sender ID – UNG S006 0040
			  • Sending Application – MSH 00004
			  • File Sending Application – FHS 00070
			  • Batch Sending Application – BHS 00084
			• Inbound – When this field is part of the trading partner lookup, the program compares contents of this field with the contents of APP_RECV_CODE loaded from the following envelope fields:
			  • Application Receiver's Code – GS 03
			  • Application Recipient ID – UNG S007 0044
			  • Receiving Application – MSH 00006
			  • File Receiving Application – FHS 00072
			  • Batch Receiving Application – BHS 00086

Name	Type	Precision	Description

SND_IDQUAL	SQL_VARCHAR	4	Qualifier that specifies the type of main code used to identify the interchange level override sender on outbound maps and interchange receiver lookup value on inbound maps. If this field is blank, the program uses the value in the WIX_IDQUAL field of the company table for outbound processing.
			• Outbound – Loads the contents of this field into the SEND_QUAL system variable, then loads SEND_QUAL into the following envelope fields:
			  • Interchange Sender ID Qualifier – ISA 05
			  • Interchange Sender ID Code Qualifier – UNB S002 0007
			• Inbound – When this field is part of the trading partner lookup, the program compares the contents of this field with the contents of the RECV_QUAL system variable, loaded from the following envelope fields:
			  • Interchange Receiver ID Qualifier – ISA 07
			  • Interchange Recipient ID Code Qualifier – UNB S003 0007

SND_IDCODE	SQL_VARCHAR	35	Main code used to identify the interchange level override sender on outbound maps and interchange receiver lookup value on inbound maps. If this field is blank, the program uses the value in the WIX_IDCODE field of the company table for outbound processing.
			• Outbound – Loads the contents of this field into the SEND_CODE, then loads SEND_CODE into the following envelope fields:
			  • Interchange Sender ID Code – ISA 06
			  • Interchange ID – UNB S002 0004
			  • Sending Facility – MSH 00003
			  • File Sending Facility – FHS 00069
			  • Batch Sending Facility – BHS 00083
			• Inbound – When this field is part of the trading partner lookup, the program compares the contents of this field with the contents of RECV_CODE, loaded from the following envelope fields:
			  • Interchange Receiver ID Code – ISA 08
			  • Interchange Receiver ID – UNB S003 0010
			  • Receiving Facility – MSH 00005
			  • File Receiving Facility – FHS 00071
			  • Batch Receiving Facility – BHS 00085

Name	Type	Precision	Description
SUB_DELIMT	SQL_VARCHAR	3	Special character used by the trading partner to override the default X12 sub-element delimiter. Outbound – If there is a value in this field, the program uses it to create outbound EDI data. If not, it uses a default value.
ELE_DELIMT	SQL_VARCHAR	3	Special character used by the trading partner to override the default X12 element delimiter. Outbound – If there is a value in this field, the program uses it to create outbound EDI data. If not, it uses a default value.
SEG_DELIMT	SQL_VARCHAR	3	Special character used by the trading partner to override the default X12 segment delimiter. Outbound – If there is a value in this field, the program uses it to create outbound EDI data. If not, it uses a default value.
RELEASE_CH	SQL_VARCHAR	3	Special character used by the trading partner to override the default X12 release character. Outbound – If there is a value in this field, the program uses it to create outbound EDI data. If not, it uses a default value.
X12_REPEAT	SQL_VARCHAR	3	Special character used by the trading partner to override the default X12 repeat character. Outbound – If there is a value in this field, the program uses it to create outbound EDI data. IfIf not, it uses a default value.
<filler>	SQL_VARCHAR	1	No longer used. (formerly DEL_CODE)
EDIF_SUBDL	SQL_VARCHAR	3	Special character used by the trading partner to override the default EDIFACT sub-element delimiter. Outbound – If there is a value in this field, the program uses it to create outbound EDI data. If not, it uses a default value.
EDIF_ELEDL	SQL_VARCHAR	3	Special character used by the trading partner to override the default EDIFACT element delimiter. Outbound – If there is a value in this field, the program uses it to create outbound EDI data. If not, it uses a default value.
EDIF_SEGDL	SQL_VARCHAR	3	Special character used by the trading partner to override the default EDIFACT segment delimiter. Outbound – If there is a value in this field, the program uses it to create outbound EDI data. If not, it uses a default value.
EDIF_RELCH	SQL_VARCHAR	3	Special character used by the trading partner to override the default EDIFACT release character. Outbound – If there is a value in this field, the program uses it to create outbound EDI data. If not, it uses a default value.

Name	Type	Precision	Description
EDIF_REPEA	SQL_VARCHAR	3	Special character used by the trading partner to override the default EDIFACT repeat character. Outbound – If there is a value in this field, the program uses it to create outbound EDI data. If not, it uses a default value.
HL7_SEGDL	SQL_VARCHAR	3	Special character used by the trading partner to override the default HL7 segment delimiter. Outbound – If there is a value in this field, the program uses it to create outbound EDI data. If not, it uses a default value.
HL7_ELEDL	SQL_VARCHAR	3	Special character used by the trading partner to override the default HL7 element delimiter. Outbound – If there is a value in this field, the program uses it to create outbound EDI data. If not, it uses a default value.
HL7_SUBDL	SQL_VARCHAR	3	Special character used by the trading partner to override the default HL7 sub-element delimiter. Outbound – If there is a value in this field, the program uses it to create outbound EDI data. If not, it uses a default value.
HL7_SUBSUB	SQL_VARCHAR	3	Special character used by the trading partner to override the default HL7 component delimiter. Outbound – If there is a value in this field, the program uses it to create outbound EDI data. If not, it uses a default value.
HL7_RELCH	SQL_VARCHAR	3	Special character used by the trading partner to override the default HL7 release character. Outbound – If there is a value in this field, the program uses it to create outbound EDI data. If not, it uses a default value.
HL7_REPEAT	SQL_VARCHAR	3	Special character used by the trading partner to override the default HL7 repeat character. Outbound – If there is a value in this field, the program uses it to create outbound EDI data. If not, it uses a default value.
EXPORT_FLAG	SQL_VARCHAR	1	Special character used to designate that flagged trading partner records can be moved from one database to another.
MBOX_NAME	SQL_VARCHAR	35	Internal name of the trading partner mailbox – used only as a label on windows or reports. The value in this field can be overridden by a value in the MBOX_NAME field in the trade agreement table.
MAILBOX	SQL_VARCHAR	100	Full-path name of the trading partner mailbox folder, or directory. The value in this field can be overridden by a value in the dest field in the trade agreement table.

Name	Type	Precision	Description
CURR_FMT	SQL_VARCHAR	1	Character used to indicate whether a period or comma is used as the decimal character • C – comma decimal character • D – period decimal character
POS_LTR	SQL_VARCHAR	1	Reserved for future use with packed decimal.
SNDR_ROUTE	SQL_VARCHAR	14 (35 for Syntax 4)	Internal code used to identify the interchange level override sender on outbound maps and interchange receiver lookup value on inbound maps. • Outbound – Loads the contents of this field into the SNDR_ROUTE system variable, then loads SNDR_ROUTE into this envelope field: Interchange Sender Internal ID – UNB S002 0008 • Inbound – When this field is part of the trading partner lookup, the program compares the contents of this field with the contents of the RCVR_ROUTE system variable, loaded from this envelope field: Interchange Receiver Internal ID – UNB S003 0014
SNDR_SUBID	SQL_VARCHAR	35	Internal sub-code used to identify the interchange level override sender on outbound maps and interchange receiver lookup value on inbound maps. (EDIFACT Syntax 4 only) • Outbound – Loads the contents of this field into the SNDR_SUBID system variable, then loads SNDR_SUBID into this envelope field: Interchange Sender Internal Sub-ID – UNB S002 0042 • Inbound – When this field is part of the trading partner lookup, the program compares the contents of this field with the contents of the RCVR_SUBID system variable, loaded from this envelope field: Interchange Receiver Internal Sub-ID – UNB S003 0046

Name	Type	Precision	Description
RCVR_ROUTE	SQL_VARCHAR	14 (35 for Syntax 4)	Internal code used to identify the interchange level default receiver on outbound maps and interchange sender lookup value on inbound maps. • Outbound – Loads the contents of this field into the RCVR_ROUTE system variable, and contents of RCVR_ROUTE into this envelope field: Interchange Receiver Internal ID – UNB S003 0014 • Inbound – When this field is part of the trading partner lookup, the program compares the contents of this field with the contents of the SNDR_ROUTE system variable, loaded from this envelope field: Interchange Sender Internal ID – UNB S002 0008
RCVR_SUBID	SQL_VARCHAR	35	Internal sub-code used to identify the interchange level default receiver on outbound maps and interchange sender lookup value on inbound maps. • Outbound – Loads the contents of this field into the RCVR_SUBID system variable, then loads RCVR_SUBID into this envelope field: Interchange Receiver Internal Sub-ID – UNB S003 0046 • Inbound – When this field is part of the trading partner lookup, the program compares the contents of this field with the contents of the SNDR_SUBID system variable, loaded from this envelope field: Interchange Sender Internal Sub-ID-UNB S002 0042
APP_SND_QL	SQL_VARCHAR	4	Qualifier that specifies type of code used to identify the group level override sender on outbound maps and group receiver lookup value on inbound maps. • Outbound – Loads the contents of this field into the APP_SEND_QUAL system variable, then loads APP_SEND_QUAL into this envelope field: Application Sender ID/ID Code Qualifier – UNG S006 0007 • Inbound – When this field is part of the trading partner lookup, the program compares the contents of this field with the contents of the APP_RECV_QUAL system variable, loaded from this envelope field: Application Recipient ID/ID Code Qualifier – UNG S007 0007

Name	Type	Precision	Description
APP_RCV_QL	SQL_VARCHAR	4	Qualifier that specifies the type of main code used to identify the group level default receiver on outbound maps and group sender lookup value on inbound maps. • Outbound – Loads the contents of this field into the APP_RECV_QUAL system variable, then loads APP_RECV_QUAL into this envelope field: Application Recipient ID/ID Code Qualifier – UNG S007 0007 • Inbound – When this field is part of the trading partner lookup, the program compares the contents of this field with the contents of the APP_SEND_QUAL system variable, loaded from this envelope field: Application Sender ID/ID Code Qualifier – UNG S006 0007
B_SEND_ID	SQL_VARCHAR	24	Batch sender ID
B_RECV_ID	SQL_VARCHAR	24	Batch receiver ID
BIN_NUMB	SQL_INTEGER	6	Bin number
PROC_NUMB	SQL_VARCHAR	10	Processing control number
SERV_QUAL	SQL_VARCHAR	2	Service provider ID qualifier
SERV_ID	SQL_VARCHAR	15	Service provider ID
SOFT_ID	SQL_VARCHAR	10	Software/vendor ID
ACK_TYPE	SQL_VARCHAR	1	Acknowledgement flag
TPKEY	SQL_INTEGER	10	Unique auto-increment field used to: • Update the ISA_IN_NO and ISA_OUT_NO control numbers • Prevent simultaneous update of the trading partner database by multiple users

Trade agreement table in non-ODBC trading partner database

The *tradstat.dbf* file contains trade agreement information for non-ODBC databases, as shown in Table 3-5.

Table 3-5: Trade agreement table in non-ODBC trading partner database

Name	Type	Width	Description
CUSTNO	Character	35	Internal application identifier for the trading partner, used to link the trading partner (*customer.dbf*) table, trade agreement (*tradstat.dbf*) table, and the application data field that has the attribute "Trading Partner ID."
MAP_TRAN	Character	6	EDI transaction (message) identifier. Outbound: • Transaction Set Identifier Code – ST 01 • Message Type – UHH S009 0065
ST03	Character	35	Value to be used as the third element in the transaction on outbound X12 processing. Outbound – Implementation Convention Reference – ST 03
MAP_EXT	Character	8	Unique map identifier
DIR	Character	3	Direction/purpose of maps: • IN or OUT – Direction of Transaction Map • PRT – Print Map • CMP – Compliance Map
STAT	Character	1	Transaction mode: • T – Test • P – Production • D – Debug • I – Information • 1–9 • Null Outbound: • Indicator – ISA 15 • TEST INDICATOR – UNB S005 0035
VERS	Character	12	Version of EDI Standard used in the map. Outbound: • Version/Release/Industry Identifier Code – GS 08 • Message Version Number – UNG S008 0052 • Message Release Number – UNG S008 0054

Name	Type	Width	Description
TBCODE	Character	60	Name of the map.
MBOX_NAME	Character	35	Name of the trading partner mailbox - used only as a label on windows and reports. A value in this field overrides a value in the MBOX_NAME field of the trading partner table.
DEST	Character	100	Folder/ full-path directory name used to override the trading partner mailbox folder/ full-path directory name – if EDI Out is checked and only for this trade agreement. A value in this field overrides a value in the MAILBOX field of the trading partner table.
FILE	Character	30	File name of the trading partner mailbox that is used – if EDI Out is checked and only for this trade agreement
GS_NO	Character	14	Unique functional group control number used with tpkey in outbound maps to quickly retrieve trade agreement records once they have been found. Outbound: • Functional Group Header Control Number – GS 06 • GROUP REFERENCE NUMBER – UNG S004 0048
ISA_TYPE	Character	5	EDI standard used by this trading partner in this transaction/message. Outbound: • Interchange Control Version Number – ISA 12 • Syntax Identifier – UNB S001 0001 (char 1-4) • Syntax Version Number – UNB S001 0002 (char 5)
SERV_CODE	Character	6	Outbound – Service Code List Directory Version Number – UNB S001 0080
<filler>	Character	1	No longer used. (formerly DEL_CODE)
RCV_GSID	Character	35	Main code used to identify the group level override receiver on outbound maps. A value is this field overrides a value in the GSID field of the trading partner table. Outbound: • Application Receiver Code – GS 03 • Application Recipient ID – UNG S007 0044 • Receiving Application – MSH 00006 • File Receiving Application – FHS 00072 • Batch Receiving Application – BHS 00086

Name	Type	Width	Description
RCV_IDQUAL	Character	4	Qualifier that specifies the type of main code used to identify the interchange level override receiver on outbound maps. A value in this field overrides a value in the ID_QUAL field of the trading partner table. Outbound: • Interchange Receiver ID Qualifier – ISA 07 • Interchange Recipient ID Code Qualifier – UNB S003 0007
RCV_IDCODE	Character	35	Main code used to identify the interchange level override receiver on outbound maps. A value in this field overrides the value in the ID_CODE field of the trading partner table. Outbound: • Interchange Receiver ID Code – ISA 08 • Interchange Receiver ID – UNB S003 0010 • Receiving Facility – MSH 00005 • File Receiving Facility – FHS 00071 • Batch Receiving Facility – BHS 00085
ACK_RQSTD	Character	1	Flag that specifies whether an X12 interchange-level TA1 acknowledgement is expected on outbound EDI maps • 1 = TA1 acknowledgement expected • = TA1 acknowledgement not expected Outbound: • Acknowledgement Requested – ISA 14 • ACKNOWLEDGEMENT REQUEST – UNB S005 0031
ACK_RQSTD2	Character	1	Flag that specifies whether an X12 group-level 997 functional acknowledgement is expected on outbound EDI maps: Outbound: • 1 = 997 functional acknowledgement expected • = 997 functional acknowledgement not expected
EDI_OUT	Character	1	Flag that specifies if inbound EDI data is to be passed through to a mailbox, only on inbound maps: • 1 = map EDI data and pass it through • = map EDI data but do not pass it through
DAYS	Character	2	“Days” portion of the time period within which the trading partner expects to receive an interchange-level acknowledgement.

Name	Type	Width	Description
HOURS	Character	2	"Hours" portion of the time period within which the trading partner expects to receive an interchange-level acknowledgement.
MINUTES	Character	2	"Minutes" portion of the time period within which the trading partner expects to receive an interchange-level acknowledgement.
SECONDS	Character	2	"Seconds" portion of the time period within which the trading partner expects to receive an interchange-level acknowledgement.
APPL_REF	Character	14	Name of the application messages contained in the EDIFACT UNB envelope. Outbound – Application Reference – UNB S005 0026
ACK_MSG	Character	1	Flag that specifies whether a message-level CONTRL segment (UCM) is generated in response to inbound EDIFACT messages. • 1 = generate UCM • = do not generate UCM
ACK_INTCH	Character	1	Flag that specifies whether an interchange-level CONTRL segment (UCI) is generated in response to inbound EDIFACT messages. • 1 = generate UCI • = do not generate UCI Outbound – Acknowledgement Request – UNB S005 0031
RCVR_ROUTE	Character	14 (35 for Syntax 4)	Internal code used to identify the interchange level override sender on outbound maps. A value in this field overrides the value in the RCVR_ROUTE field of the trading partner table. Outbound – Interchange Receiver Internal ID – UNB S003 0014
RCVR_SUBID	Character	35	Internal sub-code used to identify the interchange level override receiver on outbound maps. A value in this field overrides the value in the RCVR_SUBID field of the trading partner table. (EDIFACT Syntax 4 only) Outbound – Interchange Receiver Internal Sub-ID – UNB S003 0046
PROC_PRIOR	Character	1	Outbound – Processing Priority Code – UNB S005 0029
COMM_AGM	Character	35	Outbound – Interchange Agreement Identifier – UNB S005 0032
APP_PSWD	Character	14	Outbound – Application Password – UNG S008 0058

Name	Type	Width	Description
ASSOC_CODE	Character	6	Outbound – Association Assigned Code: • UNG S008 0057 • UNH S009 0057
CNT_AG1	Character	3	Outbound – Controlling Agency, Coded: • UNG S004 0051 • UNH S009 0051
CLIST_VER	Character	6	Outbound – Code List Directory Version Number – UNH S009 0110 (EDIFACT Syntax 4 only)
MSG_TYPE	Character	6	Outbound – Message Type Sub-Function Identifier – UNH S009 0113 (EDIFACT Syntax 4 only)
MSG_SUBID	Character	14	Outbound – Message Subset ID – UNH S016 011 5(EDIFACT Syntax 4 only)
MSG_SUBVER	Character	3	Outbound – Message Subset Version Number – UNH S016 0116 (EDIFACT Syntax 4 only)
MSG_SUBREL	Character	3	Outbound – Message Subset Release Number – UNH S016 0118 (EDIFACT Syntax 4 only)
CNT_AG2	Character	3	Outbound – Controlling Agency, Coded – UNH S016 0051 (EDIFACT Syntax 4 only)
MSG_IMPID	Character	14	Outbound – Message Implementation Guideline ID – UNH S017 0121 (EDIFACT Syntax 4 only)
MSG_IMPVER	Character	3	Outbound – Message Implementation Guideline Version Number – UNH S017 0122 (EDIFACT Syntax 4 only)
MSG_IMPREL	Character	3	Outbound – Message Implementation Guideline Release Number – UNH S017 0124 (EDIFACT Syntax 4 only)
CNT_AG3	Character	3	Outbound – Controlling Agency, Coded – UNH S017 0051 (EDIFACT Syntax 4 only)
SCEN_ID	Character	14	Outbound – Scenario ID – UNH S018 0127 (EDIFACT Syntax 4 only)
SCEN_VER	Character	3	Outbound – Scenario Version Number – UNH S018 0128 (EDIFACT Syntax 4 only)
SCEN_REL	Character	3	Outbound – Scenario Release Number – UNH S018 0130 (EDIFACT Syntax 4 only)
CNT_AG4	Character	3	Outbound – Controlling Agency, Coded – UNH S018 0051 (EDIFACT Syntax 4 only)
STD_TYPE	Character	2	The type of standard used in the map: • X X12 Standard • H HL7 Standard • E3 EDIFACT Standard, Syntax 3 • E4 EDIFACT Standard, Syntax 4

Name	Type	Width	Description
APP_RCV_QL	Character	4	Qualifier that specifies the type of main code used to identify the group level override receiver on outbound maps. (A value in this field overrides the value in the app_rcv_ql field of the trading partner table.) Outbound – Application Receiver ID/ID Code Qualifier – UNG S007 0007
TRADKEY	Numeric	10	Unique auto-increment field used to: • Update the gs_no control number • Prevent simultaneous update of the trade agreement (tradstat) table by multiple users

Trade agreement table in ODBC trading partner database

The *tradstat* file contains trade agreement information for ODBC databases, as shown in Table 3-6.

Table 3-6: Trade agreement table in ODBC trading partner database

Name	Type	Precision	Description
CUSTNO	SQL_VARCHAR	35	Internal application identifier for the trading partner, used to link the trading partner (tp) table, trade agreement (tradstat) table, and the application data field that has the attribute “Trading Partner ID.”
MAP_TRAN	SQL_VARCHAR	6	EDI transaction (message) identifier. Outbound: • Transaction Set Identifier Code – ST 01 • Message Type – UHH S009 0065
ST03	SQL_VARCHAR	35	Value to be used as the third element in the transaction on outbound X12 processing. Outbound – Implementation Convention Reference – ST 03
MAP_EXT	CHARACTER	8	Unique map identifier
DIR	SQL_VARCHAR	3	Direction/purpose of maps: • IN or OUT – direction of Transaction Map • PRT – print Map • CMP – compliance Map

Name	Type	Precision	Description
STAT	SQL_VARCHAR	1	Transaction mode: • T – test • P – production • I – information • D – debug • 1–9 • Null Outbound: • Indicator – ISA 15 • TEST INDICATOR – UNB S005 0035
VERS	SQL_VARCHAR	12	Version of EDI Standard used in the map. Outbound: • Version/Release/Industry Identifier Code – GS 08 • Message Version Number – UNG S008 0052 • Message Release Number – UNG S008 0054
TBCODE	SQL_VARCHAR	60	Name of the map.
MBOX_NAME	SQL_VARCHAR	35	Name of the trading partner mailbox - used only as a label on windows and reports. A value in this field overrides a value in the MBOX_NAME field of the trading partner table.
DEST	SQL_VARCHAR	100	Folder/ full-path directory name used to override the trading partner mailbox folder/ full-path directory name – if EDI Out is checked and only for this trade agreement. A value in this field overrides a value in the MAILBOX field of the trading partner table.
FILE	SQL_VARCHAR	30	File name of the trading partner mailbox that is used – if EDI Out is checked and only for this trade agreement
GS_NO	SQL_VARCHAR	14	Unique functional group control number used with tpkey in outbound maps to quickly retrieve trade agreement records once they have been found. Outbound: • Functional Group Header Control Number – GS 06 • GROUP REFERENCE NUMBER – UNG S004 0048

Name	Type	Precision	Description
ISA_TYPE	SQL_VARCHAR	5	EDI standard used by this trading partner in this transaction/message. Outbound: • Interchange Control Version Number – ISA 12 • Syntax Identifier – UNB S001 0001 (Char 1–4) • Syntax Version Number–UNB S001 0002 (Char 5)
SERV_CODE	SQL_VARCHAR	6	Outbound – Service Code List Directory Version Number – UNB S001 0080 (EDIFACT Syntax 4 only)
<filler>	SQL_VARCHAR	1	No longer used. (formerly DEL_CODE)
RCV_GSID	SQL_VARCHAR	35	Main code used to identify the group level override receiver on outbound maps. Outbound: • Application Receiver Code – GS 03 • Application Recipient ID – UNG S007 0044 • Receiving Application – MSH 00006 • File Receiving Application – FHS 00072 • Batch Receiving Application – BHS 00086
RCV_IDQUAL	SQL_VARCHAR	4	Qualifier that specifies the type of main code used to identify the interchange level override receiver on outbound maps. Outbound: • Interchange Receiver ID Qualifier – ISA 07 • Interchange Recipient ID Code Qualifier – UNB S003 0007
RCV_IDCODE	SQL_VARCHAR	35	Main code used to identify the interchange level override receiver on outbound maps. Outbound: • Interchange Receiver ID Code – ISA 08 • Interchange receiver ID – UNB S003 0010 • Receiving Application – MSH 00005 • File Receiving Application – FHS 00071 • Batch Receiving Application – BHS 00085

Name	Type	Precision	Description
ACK_RQSTD	SQL_VARCHAR	1	Flag that specifies whether an X12 interchange-level TA1 acknowledgement is expected on outbound EDI maps: • 1 = TA1 acknowledgement expected • = TA1 acknowledgement not expected Outbound: • Acknowledgement Requested – ISA 14 • ACKNOWLEDGEMENT REQUEST – UNB S005 0031
ACK_RQSTD2	SQL_VARCHAR	1	Flag that specifies whether an X12 group-level 997 functional acknowledgement is expected on outbound EDI maps: • 1 = 997 functional acknowledgement expected • = 997 functional acknowledgement not expected Outbound: • Acknowledgement Requested – ISA 14 • ACKNOWLEDGEMENT REQUEST – UNB S005 0031
EDI_OUT	SQL_VARCHAR	1	Flag that specifies if inbound EDI data is to be passed through to a mailbox on inbound maps: • 1 = map EDI data and pass it through • = map EDI data but do not pass it through
DAYS	SQL_VARCHAR	2	“Days” portion of the time period within which the trading partner expects to receive an interchange-level acknowledgement, on outbound maps.
HOURS	SQL_VARCHAR	2	“Hours” portion of the time period within which the trading partner expects to receive an interchange-level acknowledgement, on outbound maps.
MINUTES	SQL_VARCHAR	2	“Minutes” portion of the time period within which the trading partner expects to receive an interchange-level acknowledgement, on outbound maps.
SECONDS	SQL_VARCHAR	2	“Seconds” portion of the time period within which the trading partner expects to receive an interchange-level acknowledgement, on outbound maps.
APPL_REF	SQL_VARCHAR	14	Name of the application messages contained in the EDIFACT UNB envelope. Outbound – Application Reference – UNB S005 0026

Name	Type	Precision	Description
ACK_MSG	SQL_VARCHAR	1	Flag that specifies whether a message-level CONTRL segment (UCM) is generated in response to inbound EDIFACT messages. • 1 = generate UCM • = do not generate UCM
ACK_INTCH	SQL_VARCHAR	1	Flag that specifies whether an interchange-level CONTRL segment (UCI) is generated in response to inbound Edifact messages. • 1 = generate UCI • = do not generate UCI Outbound – Acknowledgement Request – UNB S005 0031
RCVR_ROUTE	SQL_VARCHAR	14 (35 for Syntax 4)	Internal code used to identify the group level override receiver on outbound maps and override sender on inbound maps. (A value in this field overrides the value in the RCVR_ROUTE field of the trading partner table.) Outbound – Interchange Receiver Internal ID – UNB S003 0014
RCVR_SUBID	SQL_VARCHAR	35	Internal sub-code used to identify the group level override receiver on outbound maps and override sender on inbound maps. (A value in this field overrides the value in the RCVR_SUBID field of the trading partner table.) (EDIFACT Syntax 4 only) Outbound – Interchange Receiver Internal Sub-ID – UNB S003 0046
PROC_PRIOR	SQL_VARCHAR	1	Outbound – Processing Priority Code – UNB S005 0029
COMM_AGM	SQL_VARCHAR	35	Outbound – Interchange Agreement Identifier – UNB S005 0032
APP_PSWD	SQL_VARCHAR	14	Outbound – Application Password – UNG S008 0058
ASSOC_CODE	SQL_VARCHAR	6	Outbound – Association Assigned Code: • UNG S008 0057 • UNH S009 0057
CNT_AG1	SQL_VARCHAR	3	Outbound – Controlling Agency, Coded: • UNG S004 0051 • UNH S009 0051
CLIST_VER	SQL_VARCHAR	6	Outbound – Code List Directory Version Number – UNH S009 0110(EDIFACT Syntax 4 only)

Name	Type	Precision	Description
MSG_TYPE	SQL_VARCHAR	6	Outbound – Message Type Sub-Function Identifier – UNH S009 0113 (EDIFACT Syntax 4 only)
MSG_SUBID	SQL_VARCHAR	14	Outbound – Message Subset ID – UNH S016 0115 (EDIFACT Syntax 4 only)
MSG_SUBVER	SQL_VARCHAR	3	Outbound – Message Subset Version Number – UNH S016 0116 (EDIFACT Syntax 4 only)
MSG_SUBREL	SQL_VARCHAR	3	Outbound – Message Subset Release Number – UNH S016 0118 (EDIFACT Syntax 4 only)
CNT_AG2	SQL_VARCHAR	3	Outbound – Controlling Agency, Coded – UNH S016 0051 (EDIFACT Syntax 4 only)
MSG_IMPID	SQL_VARCHAR	14	Outbound – Message Implementation Guideline ID – UNH S017 0121 (EDIFACT Syntax 4 only)
MSG_IMPVER	SQL_VARCHAR	3	Outbound – Message Implementation Guideline Version Number – UNH S017 0122 (EDIFACT Syntax 4 only)
MSG_IMPREL	SQL_VARCHAR	3	Outbound – Message Implementation Guideline Release Number – UNH S017 0124 (EDIFACT Syntax 4 only)
CNT_AG3	SQL_VARCHAR	3	Outbound – Controlling Agency, Coded – UNH S017 0051 (EDIFACT Syntax 4 only)
SCEN_ID	SQL_VARCHAR	14	Outbound – Scenario ID – UNH S018 0127 (EDIFACT Syntax 4 only)
SCEN_VER	SQL_VARCHAR	3	Outbound – Scenario Version Number – UNH S018 0128 (EDIFACT Syntax 4 only)
SCEN_REL	SQL_VARCHAR	3	Outbound – Scenario Release Number – UNH S018 0130 (EDIFACT Syntax 4 only)
CNT_AG4	SQL_VARCHAR	3	Outbound – Controlling Agency, Coded – UNH S018 0051 (EDIFACT Syntax 4 only)
STD_TYPE	SQL_VARCHAR	2	The type of standard used in the map: • X – X12 Standard • H – L7 Standard • E3 EDIFACT Standard, Syntax 3 • E4 EDIFACT Standard, Syntax 4
APP_RCV_QL	SQL_VARCHAR	4	Qualifier that specifies the type of main code used to identify the group level override receiver on outbound maps. A value in this field overrides the value in the APP_RCV_QL field of the trading partner table. Outbound – Application Receiver ID/ID Code Qualifier – UNG S007 0007

Name	Type	Precision	Description
TRADKEY	SQL_INTEGER	10	Unique auto-increment field used to: • Update the gs_no control number • Prevent simultaneous update of the Trade Agreement database (tradstat table) by multiple users

Non-ODBC transaction log table in log database

The *translog.in* and *translog.out* files contain transaction logging for non-ODBC databases, as shown in Table 3-7.

Table 3-7: Non-ODBC transaction log table in log database

Name	Type	Width	Description
RUN_ID	Number	Single	Runtime ID – loaded from internal run ID number that was passed in as a -id parameter. This run ID is also loaded once into the SYS_RUNID system variable, but the value of SYS_RUNID is not used for trlog. The run ID always numeric from initial parameter value.
TYP	Character	1	Type flag: • H – Header (ST) • T – Trailer (SE) • D – Detail (messages between ST and SE) • U – User Write Log command
RUN_DATE	Date	14	Runtime date – loaded from the SYS_DATE and SYS_HHMMSS system variables (SYS_DATE and SYS_HHMMSS are loaded from the system time.)
ACKBY_DATE	Date	14	Date by which an acknowledgement must be made • Inbound – Before each log write, if TYP is H, date is loaded from the INT_HEAD_DATE and INT_HEAD_TIME system variables, which are loaded from envelope. If TYP is not H, then null date. • Outbound – Before each log write, if TYP is H and an acknowledgment was requested, date is calculated from today's date plus the value in the DAYS, HOURS, and MINUTES fields in the trade agreement database. If TYP is not H, then null date.

Name	Type	Width	Description
TRANS_CODE	Character	2	Transaction code – loaded from the SYS_TRCODE system variable. • Inbound – SYS_TRCODE loaded from envelope. • Outbound – SYS_TRCODE loaded from parameter.
TRANS_NAME	Character	6	Transaction name – loaded from the SYS_TRANS system variable, which is loaded from internal transaction code before each log write. Message Type – MSH 00009
TPTNER_ID	Character	35	Code used to identify the trading partner Loaded from the SYS_TRADNO system variable. • Inbound – TPTNER_ID is looked up in trading partner database based on lookup criteria and data in EDI envelope. • Outbound – TPTNER_ID is loaded from an application file.

VERSION	Character	12	Version of EDI standard used in the map
			Loaded from the X12_VERSION system variable.
			• Inbound – X12_version loaded from envelope
			• Outbound – Loaded from trade agreement table
			  • Version/Release/Industry Identifier Code – GS 08
			  • Message Version Number – UNH S009 0052
			  • Message Release Number – UNH S009 0054

ISA_TYPE	Character	5	EDI standard used by this trading partner in this transaction
			• Inbound – Loaded from the INT_VERSION system variable, which is loaded from the EDI envelope.
			• Outbound – Loaded from the ISA_TYPE system variable, which is loaded from trade agreement table
			  • Interchange Control Version Number – ISA 12
			  • Syntax Identifier – UNB S001 0001
			  • Syntax Version Number – UNB S001 0002

INTERCHANG	Character	35	Interchange code – loaded from the INT_HEAD_NUM system variable
			• Inbound – Value taken from the EDI envelope.
			• Outbound – Loaded from the ISA control number taken from trading partner table and incremented.
			  • Interchange Control Number – ISA 13
			  • Interchange Control Count – UNB S004 0020

Name	Type	Width	Description

GROUP_NO	Character	35	Group number – loaded from the FUNC_GP_NUM system variable
			• Inbound – Value taken from EDI envelope.
			• Outbound – Loaded from GS control number taken from the trade agreement table and incremented
			  • Functional Group Header Control Number – GS 06
			  • Batch Control ID – BHS 00091

TRANS_NO	Character	35	Transaction number – loaded from the FUNC_ST_NUM system variable • Inbound – Value taken from EDI envelope. • Outbound – Loaded from 1000 * (GS control number taken) + transaction count.

APP_RCV_CD	Character	35	Code used to identify the group level receiver on outbound maps and sender on inbound maps. Loaded from the APP_RECV_CODE system variable.
			• Inbound – Loaded from EDI envelope.
			• Outbound – Loaded from Tradepartner.
			  • Application Receiver Code – GS 03
			  • Interchange Receiver Internal Sub-ID – UNB S003 0046
			  • Receiving Facility – MSH 00006
			  • File Receiving Facility – FHS 00072
			  • Batch Receiving Facility – BHS 00086

APP_SND_CD	Character	35	Code used to identify the group level sender on outbound maps and receiver on inbound maps. Loaded from the APP_SEND_CODE system variable.
			• Inbound – Loaded from EDI envelope
			• Outbound – Loaded from tradepartner.
			  • Application Sender Code – GS 02
			  • Interchange Sender Internal Sub-ID – UNB S002 0042
			  • Sending Facility – MSH 00004
			  • File Sending Facility – FHS 00070
			  • Batch Sending Facility – BHS 00084

Name	Type	Width	Description

RECV_CODE	Character	35	Code used to identify the interchange level receiver on outbound maps and sender on inbound maps. Loaded from the RECV_CODE system variable.
			• Inbound – Loaded from EDI envelope
			• Outbound – Loaded from tradepartner.
			  • Interchange Receiver ID Code – ISA 08
			  • Interchange Receiver ID – UNB S003 0010
			  • Receiving Application – MSH 00005
			  • File Receiving Application – FHS 00071
			  • Batch Receiving Application – BHS 00085

SEND_CODE	Character	35	Code used to identify the interchange level sender on outbound maps and receiver on inbound maps. Loaded from the SEND_CODE system variable.
			• Inbound – Loaded from EDI envelope
			• Outbound – Loaded from tradepartner.
			  • Interchange Sender ID Code – ISA 06
			  • Interchange Sender ID – UNB S002 0004
			  • Sending Application – MSH 00003
			  • File Sending Application – FHS 00069
			  • Batch Sending Application – BHS 00083

RECV_QUAL	Character	4	Qualifier that specifies the type of code used to identify the interchange level receiver on outbound maps and sender on inbound maps. Loaded from the RECV_QUAL system variable.
			• Inbound – Loaded from EDI envelope
			• Outbound – Loaded from tradepartner.
			  • Interchange Receiver ID Qualifier – ISA 07
			  • Interchange Receiver ID Code Qualifier –UNB S003 0007

SEND_QUAL	Character	4	Qualifier that specifies the type of code used to identify the interchange level sender on outbound maps and receiver on inbound maps. Loaded from the SEND_QUAL system variable.
			• Inbound – Loaded from EDI envelope
			• Outbound – Loaded from tradepartner.
			  • Interchange Sender ID Qualifier – ISA 06
			  • Interchange Sender ID Code Qualifier – UNB S002 0007

Name	Type	Width	Description
ERROR	Number	Single	Total errors – loaded from internal system count of errors between ST and SE. The LOG_ERRS and TOT_ERRS system variables are loaded at same time. LOG_ERRS is the number of errors between ST and SE. TOT_ERRS is the total number of errors for the run.
STAT	Character	1	Status – loaded from internal count of total # errors. The LOG_STATUS system variable is also loaded at time of write log. • W – SEG_ST, SEG_SE, User Write • T – Bad Tradepartner • S – Bad ISA, GS or ST • U – Stop Run • A – Abort Trans • F – Fatal Error • E – Other Error Message
BYTE-COUNT	Number	Single	Count of the number of bytes between ST and SE – is zero for every ST and increment until SE is written. The LOG_SIZE system variable is loaded with the byte count at time of write log.
DIR	Character	3	Direction of map • Outbound – OUT • Inbound – IN, CMP, PRT There is no system variable
FLOW_LEVEL	Character	5	Level of segment in flow The LOG_LEVEL system variable is loaded at time of write log.

Name	Type	Width	Description
RECORD_NAM	Character	10	Record name – the LOG_RECNAME system variable is loaded at time of write log. Can be assigned by user in write log. If: • Mapping is in progress and the field being mapped to or from is a record field, then this record name value is used. • Error occurs during a rule, then code attempts to identify the record name involved. • The field involved is a memory variable or a string variable, then the memory variable or string variable is written to the log field RECORD_NAM. • No record is associated with the error, then this field will be blank. This field cannot be written to by the user assigning a value to the system variable.
RECORD_NO	Character	6	Record number – the LOG_READ_CNT system variable can be assigned by user in write log command. Otherwise the system variable and log are loaded and written at the same time. • Inbound – The line number being processed in the incoming EDI file. • Outbound – Contains the count that this record type has been read if the error message involves a record. If not, it will be zero.
FIELD_NAME	Character	15	Field Name – the LOG_FIELDNAME system variable can be assigned by user in write log command. Otherwise the LOG_FIELDNAME and FIELD_NAME log field are loaded at the same time of write log. If: • Mapping is in progress and the field being mapped to or from is a record field, then this field name value is used. • Error occurs during a rule, then code attempts to identify the field name involved. This field cannot be written to by the user directly assigning a value to the system variable.
SEGMENT	Character	3	Segment – can be assigned by user in write log command. • Inbound – Current segment name LOG_SEG • Outbound – Will have a value on write logs for SEG_ST and SEG_SE types and for errors that occur during direct mapping to EDI file.

Name	Type	Width	Description
SEG_COUNT	Number	Long Integer	Segment count – can be assigned by user in write log command. Count of segments between ST and SE inclusive. The SEGMENT_COUNT system variable loaded every time segment is written/read from EDI File.
ELEMENT	Character	2	Element – can be assigned by user in write log command. The LOG_ELEM system variable
SUBELEM	Character	2	Sub-element – can be assigned by user in write log command. The LOG_SUBELEM system variable
SEV_CODE	Character	2	Severity code – can be assigned by user in write log command. For all system error messages this code is a 1. For other system non-error messages this code is 0.
MSG_NO	Character	5	Message number – can be assigned by user in write log command. The LOG_MSG_NO system variable
MSG_TEXT	Character	100	Message text – can be assigned by user in write log command. The LOG_MSG system variable
FILENAME	Character	160	For ST segments: • Outbound – FILENAME contains the current EDI outbound file name (can change based on tradstat and tp mailbox entries). Filename is not available as the system variable. • Inbound – The inbound EDI file name (always the same). Filename is not available as the system variable. For SE segments and inbound transactions: FILENAME • Contains the current file name of any EDI OUT file names (can change based on tradstat, tp mailboxes and tradstat EDI OUT field) • Consists of both complete path and Filename. • Is available as the EDI_OUT_FILENAME system variable.
FIELDVAL	Character	30	Field value – the LOG_VALUE system variable. Can be assigned by user in write log command. Otherwise the system variable and log field value are loaded at time of write to the log. If this is an error message, the value of the field code attempts to load the value of the record field, memory variable, or string constant in error.

Name	Type	Width	Description
USER_IDENT	Character	35	User-defined field – the SYS_USER_FIELD system variable. Log field is loaded from system variable. RTP does not assign values to SYS_USER_FIELD.
ACK_EXPECT	Character	1	Flag that specifies whether a TA1 interchange-level acknowledgement is expected, only on outbound maps: • 1 – TA1 acknowledgement requested • 0 – TA1 acknowledgement not requested the ACK_REQSTD system variable. Inbound – Loaded from EDI Envelope (ISA element 14).
TR_ACK_TYP	Character	1	Flag that specifies whether a group-level functional acknowledgement is expected, only on outbound maps: • 1 – functional acknowledgement requested • 0 – functional acknowledgement not requested The corresponding system variable is TR_ACK. Outbound – Loaded from tradstat table
T_P_IND	Character	1	Test/Production Indicator • T – T Test • P – Production • I – Information • D – Debug • 1–9 • Null The TEST_IND system variable Inbound – Loaded from EDI envelope Outbound – Loaded from Tradstat.
TRANS_CNT	Number	Long Integer	Transaction count No system variable. Outbound – Log value is loaded from an internal count of the number of ST–SE transactions read or written between SEG_GS and SEG_GE.
FILEOFFSET	Number	Single	Number of Bytes File Offset No system variable. Written from internal count of number of bytes read (inbound) or number of bytes written to the EDI file.
RCOUNT	Number	Integer	Field for Record Manipulation Always set equal to 1 before log write.

Name	Type	Width	Description
SNDR_ROUTE	Character	14 (35 for Syntax 4)	Internal code used to identify the interchange level sender on outbound maps and receiver on inbound maps. Interchange Sender Internal ID – UNB S002 0008
SNDR_SUBID	Character	35	Internal sub-code used to identify the interchange level sender on outbound maps and receiver on inbound maps. Interchange Sender Internal Sub-ID – UNB S002 0042 (EDIFACT Syntax 4 only)
RCVR_ROUTE	Character	14 (35 for Syntax 4)	Internal code used to identify the interchange level receiver on outbound maps and sender on inbound maps. Interchange Receiver Internal ID – UNB S003 0014
RCVR_SUBID	Character	35	Internal sub-code used to identify the interchange level receiver on outbound maps and sender on inbound maps. Interchange Receiver Internal Sub-ID – UNB S003 0046 (EDIFACT Syntax 4 only)
APPL_REF	Character	14	Name of the application messages contained in the EDIFACT UNB envelope. Application Reference – UNB S005 0026
PROC_PRIOR	Character	1	Processing Priority Code – UNB S005 0029
COMM_AGM	Character	35	Interchange Agreement Identifier – UNB S005 0032
APP_SND_QL	Character	4	Qualifier that specifies the type of code used to identify the trading partner at the group level – as the sender on outbound maps and as the receiver on inbound maps. Application Sender ID/ID Code Qualifier – UNG S006 0007
APP_RCV_QL	Character	4	Qualifier that specifies the type of code used to identify the trading partner at the group level – as the receiver on outbound maps and as the sender on inbound maps. Application Receiver ID/ID Code Qualifier – UNG S007 0007
ASSOC_CODE	Character	6	Association Assigned Code – • UNG S008 0057 • UNH S009 0057
APP_PSWD	Character	14	Application Password – UNG S008 0058
CLIST_VER	Character	6	Code List Directory Version Number – UNH S009 0110 (EDIFACT Syntax 4 only)
MSG_TYPE	Character	6	Message Type Sub-Function Identifier – UNH S009 0113 (EDIFACT Syntax 4 only)

ODBC transaction log table in log database

The *trlog* file contains transaction logging information for ODBC databases, as shown in Table 3-8.

Table 3-8: ODBC transaction log table in log database – expanded log

Name	Type	Precision	Description
AFLD	SQL_INTEGER	10	Auto increment field (AutoNumber)
RUN_ID	SQL_BIGINT	9	Runtime ID – loaded from internal run id number that was passed in as a –id parameter. This run id is also loaded once into the SYS_RUNID system variable, but the value of SYS_RUNID is not used for TRLOG. The run id always numeric from initial parameter value.
TYP	SQL_VARCHAR	1	Type flag: • H – header (ST) • T – trailer (SE) • D – detail (messages between ST and SE) • U – user write log command
RUN_DATE	SQL_TIMESTAMP	14	Runtime date – loaded from the SYS_DATE and SYS_HHMMSS system variables (SYS_DATE and SYS_HHMMSS are loaded from the system time.)
ACKBY_DATE	SQL_TIMESTAMP	14	Date by which an acknowledgement must be made • Inbound – Before each log write, if TYP is H, date is loaded from the INT_HEAD_DATE and INT_HEAD_TIME system variables, which are loaded from envelope. If TYP is not H, then null date. • Outbound – Before each log write, if TYP is H and an acknowledgment was requested, date is calculated from today's date plus the value in the DAYS, HOURS, and MINUTES fields in the tradstat database. If TYP is not H, then null date.
TRANS_CODE	SQL_VARCHAR	2	Transaction code – loaded from the SYS_TRCODE system variable. • Inbound – SYS_TRCODE loaded from envelope. • Outbound – SYS_TRCODE loaded from parameter.

Name	Type	Precision	Description
TRANS_NAME	SQL_VARCHAR	6	Transaction name – loaded from the SYS_TRANS system variable, which is loaded from internal transaction code before each log write. Message Type – MSH 00009
TPTNER_ID	SQL_VARCHAR	35	Code used to identify the trading partner Loaded from the SYS_TRADNO system variable. • Inbound – TPTNER_ID is looked up in trading partner database based on lookup criteria and data in EDI envelope. • Outbound – TPTNER_ID is loaded from an application file.

VERSION	SQL_VARCHAR	12	Version of EDI standard used in the map
			Loaded from the X12_VERSION system variable.
			• Inbound – X12 version loaded from envelope
			• Outbound – Loaded from tradstat table
			  • Version/Release/Industry Identifier Code – GS 08
			  • Message Version Number – UNH S009 0052
			  • Message Release Number – UNH S009 0054

ISA_TYPE	SQL_VARCHAR	5	EDI standard used by this trading partner in this transaction
			• Inbound – Loaded from the INT_VERSION system variable, which is loaded from the EDI envelope.
			• Outbound – Loaded from the ISA_TYPE system variable, which is loaded from tradstat table.
			  • Interchange Control Version Number – ISA 12
			  • Syntax Identifier – UNB S001 0001
			  • Syntax Version Number – UNB S001 0002

INTERCHANG	SQL_VARCHAR	35	Interchange code – loaded from the INT_HEAD_NUM system variable
			• Inbound – Value taken from the EDI envelope.
			• Outbound – Loaded from the ISA control number taken from trading partner table and incremented.
			  • Interchange Control Number – ISA 13
			  • Interchange Control Count – UNB S004 0020

Name	Type	Precision	Description

GROUP_NO	SQL_VARCHAR	35	Group number – loaded from the FUNC_GP_NUM system variable
			• Inbound – Value taken from EDI envelope.
			• Outbound – Loaded from GS control number taken from the tradstat table and incremented
			  • Functional Group Header Control Number – GS 06
			  • Batch Control ID – BHS 00091

TRANS_NO	SQL_VARCHAR	35	Transaction number – loaded from the FUNC_ST_NUM system variable • Inbound – Value taken from EDI envelope. • Outbound – Loaded from 1000 * (GS control number taken) + transaction count.

APP_RCV_CD	SQL_VARCHAR	35	Code used to identify the group level receiver on outbound maps and sender on inbound maps. Loaded from the APP_RECV_CODE system variable.
			• Inbound – Loaded from EDI envelope.
			• Outbound – Loaded from Tradepartner.
			  • Application Receiver Code – GS 03
			  • Interchange Receiver Internal Sub-ID – UNB 0003 0046
			  • Receiving Facility – MSH 00006
			  • File Receiving Facility – FHS 00072
			  • Batch Receiving Facility – BHS 00086

APP_SND_CD	SQL_VARCHAR	35	Code used to identify the group level sender on outbound maps and receiver on inbound maps. Loaded from the APP_SEND_CODE system variable.
			• Inbound – Loaded from EDI envelope
			• Outbound – Loaded from tradepartner.
			  • Application Sender Code – GS 02
			  • Interchange Sender Internal Sub-ID – UNB S002 0042
			  • Sending Facility – MSH 00004
			  • File Sending Facility – FHS 00070
			  • Batch Sending Facility – BHS 00084

Name	Type	Precision	Description

RECV_CODE	SQL_VARCHAR	35	Code used to identify the interchange level receiver on outbound maps and sender on inbound maps. Loaded from the RECV_CODE system variable.
			• Inbound – Loaded from EDI envelope
			• Outbound – Loaded from tradepartner.
			  • Interchange Receiver ID Code – ISA 08
			  • Interchange Receiver ID – UNB S003 0010
			  • Receiving Application – MSH 00005
			  • File Receiving Application – FHS 00071
			  • Batch Receiving Application – BHS 00085

SEND_CODE	SQL_VARCHAR	35	Code used to identify the interchange level sender on outbound maps and receiver on inbound maps.
			Loaded from the SEND_CODE system variable.
			• Inbound – Loaded from EDI envelope
			• Outbound – Loaded from tradepartner.
			  • Interchange Sender ID Code – ISA 06
			  • Interchange Sender ID – UNB S002 0004
			  • Sending Application – MSH 00003
			  • File Sending Application – FHS 00069
			  • Batch Sending Application – BHS 00083

RECV_QUAL	SQL_VARCHAR	4	Qualifier that specifies the type of code used to identify the interchange level receiver on outbound maps and sender on inbound maps.
			Loaded from the RECV_QUAL system variable.
			• Inbound – Loaded from EDI envelope
			• Outbound – Loaded from tradepartner.
			  • Interchange Receiver ID Qualifier – ISA 07
			  • Interchange Receiver ID Code Qualifier – UNB S003 0007

Name	Type	Precision	Description

SEND_QUAL	SQL_VARCHAR	4	Qualifier that specifies the type of code used to identify the interchange level sender on outbound maps and receiver on inbound maps.
			Loaded from the SEND_QUAL system variable.
			• Inbound – Loaded from EDI envelope
			• Outbound – Loaded from tradepartner.
			  • Interchange Sender ID Qualifier – ISA 06
			  • Interchange Sender ID Code Qualifier – UNB S002 0007

ERRORS	SQL_BIGINT	10	Total errors – loaded from internal system count of errors between ST and SE. The LOG_ERRS and TOT_ERRS system variables are loaded at same time. LOG_ERRS is the number of errors between ST and SE. TOT_ERRS is the total number of errors for the run.
STAT	SQL_VARCHAR	1	Status – loaded from internal count of total # errors. The LOG_STATUS system variable is also loaded at time of write log. • W – SEG_ST, SEG_SE, User Write • T – Bad Tradepartner • S – Bad ISA, GS or ST • U – Stop Run • A – Abort Trans • F – Fatal Error • E – Other Error Message
BYTE-COUNT	SQL_BIGINT	10	Count of the number of bytes between ST and SE – will be zero for every ST and increment until SE is written. The LOG_SIZE system variable is loaded with the byte count at time of write log.
DIR	SQL_VARCHAR	3	Direction of map • Outbound – OUT • Inbound – IN, CMP, PRT There is no system variable
FLOW_LEVEL	SQL_VARCHAR	5	Level of segment in flow – the LOG_LEVEL system variable is loaded at time of write log.

Name	Type	Precision	Description
RECORD_NAM	SQL_VARCHAR	10	Record name – the LOG_RECNAME system variable is loaded at time of write log. Can be assigned by user in write log if the following conditions are present: • Mapping is in progress and the field being mapped to or from is a record field, then this record name value is used. • Error occurs during a rule, then code attempts to identify the record name involved. • The field involved is a memory variable or a string variable, then the memory variable or string variable is written to the RECORD_NAM log field. • No record is associated with the error, then this field is blank. This field cannot be written to by the user assigning a value to the system variable.
RECORD_NO	SQL_VARCHAR	6	Record number – the LOG_READ_CNT system variable can be assigned by user in write log command. Otherwise the system variable and log are loaded and written at the same time. • Inbound – This is the line number being processed in the incoming EDI file. • Outbound – This field will contain the count that this record type has been read if the error message involves a record. Otherwise it will be zero.
FIELD_NAME	SQL_VARCHAR	15	Field name – the LOG_FIELDNAME system variable can be assigned by user in write log command. Otherwise LOG_FIELDNAME and FIELD_NAME log field are loaded at the same time of write log. If mapping is in progress and the field being mapped to or from is a record field, then this field name value is used. If error occurs during a rule, then code attempts to identify the field name involved. This field cannot be written to by the user directly assigning a value to the system variable.

Name	Type	Precision	Description
SEGMENT	SQL_VARCHAR	3	Segment – can be assigned by user in write log command. • Inbound – Current segment name LOG_SEG • Outbound – Will have a value on write logs for SEG_ST and SEG_SE types and for errors that occur during direct mapping to EDI file.
SEG_COUNT	SQL_INTEGER	10	Segment count – can be assigned by user in write log command. Count of segments between ST and SE inclusive. The SEGMENT_COUNT system variable loaded every time segment is written/read from EDI File.
ELEMENT	SQL_VARCHAR	2	Element – can be assigned by user in write log command. The LOG_ELEM system variable
SUBELEM	SQL_VARCHAR	2	Sub-element – can be assigned by user in write log command. The LOG_SUBELEM system variable
SEV_CODE	SQL_VARCHAR	2	Severity code – can be assigned by user in write log command. For all system error messages this code is a 1. For other system non-error messages this code is 0.
MSG_NO	SQL_VARCHAR	5	Message number – can be assigned by user in write log command. The LOG_MSG_NO system variable
MSG_TEXT	SQL_VARCHAR	100	Message text – can be assigned by user in write log command. The LOG_MSG system variable

Name	Type	Precision	Description
FILENAME	SQL_VARCHAR	160	For ST segments: • Outbound – FILENAME contains the current EDI outbound file name (can change based on tradstat and tp mailbox entries). Filename is not available as the system variable. • Inbound – The inbound EDI file name (always the same).Filename is not available as the system variable. For SE segments and inbound transactions: FILENAME • Contains the current file name of any EDI OUT file names (can change based on tradstat, tp mailboxes and tradstat EDI_OUT field) • Consists of both complete path and file name. • Is available as the EDI_OUT_FILENAME system variable.
FIELDVAL	SQL_VARCHAR	30	Field value – the LOG_VALUE system variable. Can be assigned by user in write log command. Otherwise the system variable and log field value are loaded at time of write to the log. If this is an error message, the value of the field code attempts to load the value of the record field, memory variable, or string constant in error.
USER_IDENT	SQL_VARCHAR	35	User-defined field – the SYS_USER_FIELD system variable. Log field is loaded from system variable. RTP does not assign values to SYS_USER_FIELD.
ACK_EXPECT	SQL_VARCHAR	1	Flag that specifies whether a TA1 interchange-level acknowledgement is expected, only on outbound maps: • 1 = TA1 acknowledgement requested • 0 = TA1 acknowledgement not requested The ACK_REQSTD system variable. Inbound – Loaded from EDI Envelope (ISA element 14).

Name	Type	Precision	Description
TR_ACK_TYP	SQL_VARCHAR	1	Flag that specifies whether a group-level functional acknowledgement is expected, only on outbound maps: • 1 = functional acknowledgement requested • 0 = functional acknowledgement not requested The corresponding system variable is TR_ACK. Outbound – Loaded from tradstat table
T_P_IND	SQL_VARCHAR	1	Test/Production Indicator • T – T Test • P – Production • I – Information • D – Debug • 1–9 • Null The TEST_IND system variable Inbound – Loaded from EDI envelope Outbound – Loaded from Tradstat.
TRANS_CNT	SQL_INTEGER	10	Transaction count – no system Variable. Inbound/Outbound – This log value is loaded from an internal count of the number of ST-SE transactions read or written between SEG_GS and SEG_GE.
FILEOFFSET	SQL_BIGINT	10	Number of Bytes File Offset – no system variable. Written from internal count of number of bytes read (inbound) or number of bytes written to the EDI file.
RCOUNT	SQL_SMALLINT	1	Field for Record Manipulation – always set equal to 1 before log write.
SNDR_ROUTE	SQL_VARCHAR	14 (35 for Syntax 4)	Internal code used to identify the interchange level sender on outbound maps and receiver on inbound maps. Interchange Sender Internal ID – UNB S002 0008
SNDR_SUBID	SQL_VARCHAR	35	Internal sub-code used to identify the interchange level sender on outbound maps and receiver on inbound maps. (EDIFACT Syntax 4 only) Interchange Sender Internal Sub-ID – UNB S002 0042

Name	Type	Precision	Description
RCVR_ROUTE	SQL_VARCHAR	14 (35 for Syntax 4)	Internal code used to identify the interchange level receiver on outbound maps and sender on inbound maps. Interchange Receiver Internal ID – UNB S003 0014
RCVR_SUBID	SQL_VARCHAR	35	Internal sub-code used to identify the interchange level receiver on outbound maps and sender on inbound maps. (EDIFACT Syntax 4 only) Interchange Receiver Internal Sub-ID-UNB S003 0046
APPL_REF	SQL_VARCHAR	14	Name of the application messages contained in the EDIFACT UNB envelope. Application Reference – UNB S005 0026
PROC_PRIOR	SQL_VARCHAR	1	Processing Priority Code – UNB S005 0029
COMM_AGM	SQL_VARCHAR	35	Interchange Agreement Identifier – UNB S005 0032
APP_SND_QL	SQL_VARCHAR	4	Qualifier that specifies the type of code used to identify the trading partner at the group level – as the sender on outbound maps and as the receiver on inbound maps. Application Sender ID/ID Code Qualifier – UNG S006 0007
APP_RCV_QL	SQL_VARCHAR	4	Qualifier that specifies the type of code used to identify the trading partner at the group level – as the receiver on outbound maps and as the sender on inbound maps. Application Receiver ID/ID Code Qualifier – UNG S007 0007
ASSOC_CODE	SQL_VARCHAR	6	Association Assigned Code – • UNG S008 0057 • UNH S009 0057
APP_PSWD	SQL_VARCHAR	14	Application Password – UNG S008 0058
CLIST_VER	SQL_VARCHAR	6	(EDIFACT Syntax 4 only) Code List Directory Version Number – UNH S009 0110
MSG_TYPE	SQL_VARCHAR	6	(EDIFACT Syntax 4 only) Message Type Sub-Function Identifier – UNH S009 0113

Name	Type	Precision	Description
RPT_NO	SQL_VARCHAR	2	(X12 version 40020 and later) Repeat Number – this field holds the number of a repeating element
ST03	SQL_VARCHAR	35	Implementation Convention Reference – ST 03 Value to be used as the third element in the transaction on outbound X12 processing.

CHAPTER 4

EDI Envelopes

About this chapter

This chapter describes EDI envelopes.

Topics

This chapter contains the following topics:

Overview

For EDI messages to be delivered and recognized, they need addressing information. In ECMap, EDI messages are enclosed in both an outer and an inner envelope that each contains addressing information. The information on these envelopes identifies the intended recipient of the message, allowing the data to be correctly sent and delivered. It also identifies the sender of the message. The outer envelope is referred to as the interchange level envelope, and the inner envelope is referred to as the group level envelope. One outer envelope can contain multiple inner envelopes. One inner envelope can contain multiple messages.

Different standards use different methods to provide the addressing information required for delivery of the message. Sometimes a message is sent in both an inner and an outer envelope. Sometimes a message is sent in only one envelope. And sometimes a message is sent with no envelope at all; instead, the addressing information is included in the header and trailer of the message itself.

Envelope types

In X12, addressing information is included in both an inner and an outer envelope. The outer envelope begins with an ISA segment and ends with an ISE segment. The inner envelope begins with a GS segment and ends with a GE segment. The message, or transaction, begins with an ST segment and ends with an SE segment.

In EDIFACT, there is always an outer envelope, but an inner envelope is not necessarily used. The outer envelope begins with a UNB segment and ends with a UNZ segment. The inner envelope begins with a UNG segment and ends with a UNE segment. The message begins with a UNH segment and ends with a UNT segment. There is an optional UNA segment that is used only to change the default delimiters.

In HL7, while there are outer and inner envelopes, they are used infrequently. The outer envelope begins with a FHS segment and ends with a FTS segment. The inner envelope begins with a BHS segment and ends with a BTS segment. Instead, addressing information is included in the header and trailer of the message. A message begins with the MSH segment and has no ending segment. (In ECMap, we artificially create an EOT end-of-message segment so that the map flow works correctly.)

The following charts show how the components of each EDI envelope are used by, stored in, or created by ECMap. The following information is shown for each element:

- The window and text box in which the value for each element is entered.
- The table in the trading partner database in which the information entered in the text boxes on the window is stored and from which it is retrieved.
- The system variable into which the information in the databases is loaded and from which the information is obtained.

The envelope structures are presented in the order shown below:

X12 envelope

Table 4-1: X12 envelopes

Envelope type	Description
ISA/IEA	Outer Envelope
GS/GE	Inner Envelope
ST/SE	Transaction

EDIFACT envelope

Table 4-2: EDIFACT envelopes

Envelope type	Description
UNB/UNZ	Outer Envelope
UNG/UNE	Inner Envelope
UNH/UNT	Message
FHS/FTS	Outer Envelope
BHS/BTS	Inner Envelope
MSH	Message (often used without an envelope)

HL7 envelopes

For outbound envelopes, the values used to create the envelopes can come from several sources. Values to the left always override values to the right. For example, if there is value in the trade agreement table, it overrides a value in the trading partner table. If a value exists in the trading partner table, it overrides a value in the trade agreement table.

For inbound envelopes, the values in the envelope are placed in the system variables to the far right.

For some system variables, values may be entered into the system variables via a first-level Before Rule. Values entered in this way override any values previously loaded from the trading partner database or the I/O Rule.

CHAPTER 5

ASCII Character Chart

About this chapter

This chapter describes ASCII characters and provides ASCII charts.

Topics

This chapter contains the following topics:

About ASCII characters

In ECMap, ASCII characters are used as delimiters—special characters that separate objects. For example, they are used to separate objects such as segments, elements, subelements, fields, components, and subcomponents. ASCII characters are also used as escape, or release, characters and as repeat, or repetition, characters.

The following pages contain the following information for each of the ASCII characters that can be used in ECMap:

- Decimal representation
- Binary representation
- Hexadecimal representation
- Description
- Abbreviation
- Printable character
- Non-printable character

ASCII Set 1

Dec	Bin	Hex	Description	Abbr	Non-Printable	Printable
0	0000 0000	0	NULL	NUL		
1	0000 0001	1	START OF HEADING	SOH		
2	0000 0010	2	START OF TEXT	STX		
3	0000 0011	3	END OF TEXT	ETX		
4	0000 0100	4	END OF TRANSMISSION	EOT		
5	0000 0101	5	ENQUIRY	ENQ		
6	0000 0110	6	ACKNOWLEDGE	ACK		

Dec	Bin	Hex	Description	Abbr	Non-Printable	Printable
7	0000 0111	7	BELL	BEL		
8	0000 1000	8	BACKSPACE	BS		
9	0000 1001	9	CHARACTER TABULATION	HT		
10	0000 1010	A	LINE FEED	LF		
11	0000 1011	B	LINE TABULATION	VT		
12	0000 1100	C	FORM FEED	FF		
13	0000 1101	D	CARRIAGE RETURN	CR		
14	0000 1110	E	SHIFT OUT	SO		
15	0000 1111	F	SHIFT IN	SI		
16	0001 0000	10	DATALINK ESCAPE	DLE		
17	0001 0001	11	DEVICE CONTROL ONE	DC1		
18	0001 0010	12	DEVICE CONTROL TWO	DC2		
19	0001 0011	13	DEVICE CONTROL THREE	DC3		
20	0001 0100	14	DEVICE CONTROL FOUR	DC4		
21	0001 0101	15	NEGATIVE ACKNOWLEDGE	NAK		
22	0001 0110	16	SYNCHRONOUS IDLE	SYN		
23	0001 0111	17	END OF TRANSMISSION BLOCK	ETB		
24	0001 1000	18	CANCEL	CAN		
25	0001 1001	19	END OF MEDIUM	EM		

Dec	Bin	Hex	Description	Abbr	Non-Printable	Printable
26	0001 1010	1A	SUBSTITUTE	SUB		
27	0001 1011	1B	ESCAPE	ESC		
28	0001 1100	1C	FILE SEPARATOR	IS4		
29	0001 1101	1D	GROUP SEPARATOR	IS3		
30	0001 1110	1E	RECORD SEPARATOR	IS2		
31	0001 1111	1F	UNIT SEPARATOR	IS1		
32	0010 0000	20	SPACE			
33	0010 0001	21	EXCLAMATION MARK			!
34	0010 0010	22	QUOTATION MARK			"
35	0010 0011	23	NUMBER SIGN			#
36	0010 0100	24	DOLLAR SIGN			$
37	0010 0101	25	PERCENT SIGN			%
38	0010 0110	26	AMPERSAND			&
39	0010 0111	27	APOSTROPHE			'
40	0010 1000	28	LEFT PARENTHESIS			(
41	0010 1001	29	RIGHT PARENTHESIS			)
42	0010 1010	2A	ASTERISK			*
43	0010 1011	2B	PLUS SIGN			+
44	0010 1100	2C	COMMA			,

Dec	Bin	Hex	Description	Abbr	Non-Printable	Printable
45	0010 1101	2D	HYPHEN-MINUS			-
46	0010 1110	2E	FULL STOP			.
47	0010 1111	2F	SOLIDUS			/
48	0011 0000	30	DIGIT ZERO			0
49	0011 0001	31	DIGIT ONE			1
50	0011 0010	32	DIGIT TWO			2
51	0011 0011	33	DIGIT THREE			3
52	0011 0100	34	DIGIT FOUR			4
53	0011 0101	35	DIGIT FIVE			5
54	0011 0110	36	DIGIT SIX			6
55	0011 0111	37	DIGIT SEVEN			7
56	0011 1000	38	DIGIT EIGHT			8
57	0011 1001	39	DIGIT NINE			9
58	0011 1010	3A	COLON			
59	0011 1011	3B	SEMICOLON			;
60	0011 1100	3C	LESS-THAN SIGN			<
61	0011 1101	3D	EQUALS SIGN			=
62	0011 1110	3E	GREATER-THAN SIGN			>
63	0011 1111	3F	QUESTION MARK			?

Dec	Bin	Hex	Description	Abbr	Non-Printable	Printable
64	0100 0000	40	COMMERCIAL AT			@
65	0100 0001	41	LATIN CAPITAL LETTER A			A
66	0100 0010	42	LATIN CAPITAL LETTER B			B
67	0100 0011	43	LATIN CAPITAL LETTER C			C
68	0100 0100	44	LATIN CAPITAL LETTER D			D
69	0100 0101	45	LATIN CAPITAL LETTER E			E
70	0100 0110	46	LATIN CAPITAL LETTER F			F
71	0100 0111	47	LATIN CAPITAL LETTER G			G
72	0100 1000	48	LATIN CAPITAL LETTER H			H
73	0100 1001	49	LATIN CAPITAL LETTER I			I
74	0100 1010	4A	LATIN CAPITAL LETTER J			J
75	0100 1011	4B	LATIN CAPITAL LETTER K			K
76	0100 1100	4C	LATIN CAPITAL LETTER L			L
77	0100 1101	4D	LATIN CAPITAL LETTER M			M
78	0100 1110	4E	LATIN CAPITAL LETTER N			N
79	0100 1111	4F	LATIN CAPITAL LETTER O			O
80	0101 0000	50	LATIN CAPITAL LETTER P			P
81	0101 0001	51	LATIN CAPITAL LETTER Q			Q
82	0101 0010	52	LATIN CAPITAL LETTER R			R

Dec	Bin	Hex	Description	Abbr	Non-Printable	Printable
83	0101 0011	53	LATIN CAPITAL LETTER S			S
84	0101 0100	54	LATIN CAPITAL LETTER T			T
85	0101 0101	55	LATIN CAPITAL LETTER U			U
86	0101 0110	56	LATIN CAPITAL LETTER V			V
87	0101 0111	57	LATIN CAPITAL LETTER W			W
88	0101 1000	58	LATIN CAPITAL LETTER X			X
89	0101 1001	59	LATIN CAPITAL LETTER Y			Y
90	0101 1010	5A	LATIN CAPITAL LETTER Z			Z
91	0101 1011	5B	LEFT SQUARE BRACKET			[
92	0101 1100	5C	REVERSE SOLIDUS			\
93	0101 1101	5D	RIGHT SQUARE BRACKET			]
94	0101 1110	5E	CIRCUMFLEX ACCENT			^
95	0101 1111	5F	LOW LINE			_
96	0110 0000	60	GRAVE ACCENT			`
97	0110 0001	61	LATIN SMALL LETTER A			a
98	0110 0010	62	LATIN SMALL LETTER B			b
99	0110 0011	63	LATIN SMALL LETTER C			c
100	0110 0100	64	LATIN SMALL LETTER D			d
101	0110 0101	65	LATIN SMALL LETTER E			e

Dec	Bin	Hex	Description	Abbr	Non-Printable	Printable
102	0110 0110	66	LATIN SMALL LETTER F			f
103	0110 0111	67	LATIN SMALL LETTER G			g
104	0110 1000	68	LATIN SMALL LETTER H			h
105	0110 1001	69	LATIN SMALL LETTER I			i
106	0110 1010	6A	LATIN SMALL LETTER J			j
107	0110 1011	6B	LATIN SMALL LETTER K			k
108	0110 1100	6C	LATIN SMALL LETTER L			l
109	0110 1101	6D	LATIN SMALL LETTER M			m
110	0110 1110	6E	LATIN SMALL LETTER N			n
111	0110 1111	6F	LATIN SMALL LETTER O			o
112	0111 0000	70	LATIN SMALL LETTER P			p
113	0111 0001	71	LATIN SMALL LETTER Q			q
114	0111 0010	72	LATIN SMALL LETTER R			r
115	0111 0011	73	LATIN SMALL LETTER S			s
116	0111 0100	74	LATIN SMALL LETTER T			t
117	0111 0101	75	LATIN SMALL LETTER U			u
118	0111 0110	76	LATIN SMALL LETTER V			v
119	0111 0111	77	LATIN SMALL LETTER W			w
120	0111 1000	78	LATIN SMALL LETTER X			x

Dec	Bin	Hex	Description	Abbr	Non-Printable	Printable
121	0111 1001	79	LATIN SMALL LETTER Y			y
122	0111 1010	7A	LATIN SMALL LETTER Z			z
123	0111 1011	7B	LEFT CURLY BRACKET			{
124	0111 1100	7C	VERTICAL LINE			!!
125	0111 1101	7D	RIGHT CURLY BRACKET			}
126	0111 1110	7E	TILDE			~
127	0111 1111	7F	DELETE (DEL)	DEL		

ASCII Set 2

Dec	Bin	Hex	Description	Abbr	Non-Printable	Printable
0	0000 0000	0	NULL	NUL		
1	0000 0001	1	START OF HEADING	SOH		
2	0000 0010	2	START OF TEXT	STX		
3	0000 0011	3	END OF TEXT	ETX		
4	0000 0100	4	END OF TRANSMISSION	EOT		
5	0000 0101	5	ENQUIRY	ENQ		
6	0000 0110	6	ACKNOWLEDGE	ACK		
7	0000 0111	7	BELL	BEL		

Dec	Bin	Hex	Description	Abbr	Non-Printable	Printable
8	0000 1000	8	BACKSPACE	BS		
9	0000 1001	9	CHARACTER TABULATION	HT		
10	0000 1010	A	LINE FEED	LF		
11	0000 1011	B	LINE TABULATION	VT		
12	0000 1100	C	FORM FEED	FF		
13	0000 1101	D	CARRIAGE RETURN	CR		
14	0000 1110	E	SHIFT OUT	SO		
15	0000 1111	F	SHIFT IN	SI		
16	0001 0000	10	DATALINK ESCAPE	DLE		
17	0001 0001	11	DEVICE CONTROL ONE	DC1		
18	0001 0010	12	DEVICE CONTROL TWO	DC2		
19	0001 0011	13	DEVICE CONTROL THREE	DC3		
20	0001 0100	14	DEVICE CONTROL FOUR	DC4		
21	0001 0101	15	NEGATIVE ACKNOWLEDGE	NAK		
22	0001 0110	16	SYNCHRONOUS IDLE	SYN		
23	0001 0111	17	END OF TRANSMISSION BLOCK	ETB		
24	0001 1000	18	CANCEL	CAN		
25	0001 1001	19	END OF MEDIUM	EM		
26	0001 1010	1A	SUBSTITUTE	SUB		

Dec	Bin	Hex	Description	Abbr	Non-Printable	Printable
27	0001 1011	1B	ESCAPE	ESC		
28	0001 1100	1C	FILE SEPARATOR	IS4		
29	0001 1101	1D	GROUP SEPARATOR	IS3		
30	0001 1110	1E	RECORD SEPARATOR	IS2		
31	0001 1111	1F	UNIT SEPARATOR	IS1		
32	0010 0000	20	SPACE			
33	0010 0001	21	EXCLAMATION MARK			!
34	0010 0010	22	QUOTATION MARK			"
35	0010 0011	23	NUMBER SIGN			#
36	0010 0100	24	DOLLAR SIGN			$
37	0010 0101	25	PERCENT SIGN			%
38	0010 0110	26	AMPERSAND			&
39	0010 0111	27	APOSTROPHE			'
40	0010 1000	28	LEFT PARENTHESIS			(
41	0010 1001	29	RIGHT PARENTHESIS			)
42	0010 1010	2A	ASTERISK			*
43	0010 1011	2B	PLUS SIGN			+
44	0010 1100	2C	COMMA			,
45	0010 1101	2D	HYPHEN-MINUS			-

Dec	Bin	Hex	Description	Abbr	Non-Printable	Printable
46	0010 1110	2E	FULL STOP			.
47	0010 1111	2F	SOLIDUS			/
48	0011 0000	30	DIGIT ZERO			0
49	0011 0001	31	DIGIT ONE			1
50	0011 0010	32	DIGIT TWO			2
51	0011 0011	33	DIGIT THREE			3
52	0011 0100	34	DIGIT FOUR			4
53	0011 0101	35	DIGIT FIVE			5
54	0011 0110	36	DIGIT SIX			6
55	0011 0111	37	DIGIT SEVEN			7
56	0011 1000	38	DIGIT EIGHT			8
57	0011 1001	39	DIGIT NINE			9
58	0011 1010	3A	COLON			:
59	0011 1011	3B	SEMICOLON			;
60	0011 1100	3C	LESS-THAN SIGN			<
61	0011 1101	3D	EQUALS SIGN			=
62	0011 1110	3E	GREATER-THAN SIGN			>
63	0011 1111	3F	QUESTION MARK			?
64	0100 0000	40	COMMERCIAL AT			@

Dec	Bin	Hex	Description	Abbr	Non-Printable	Printable
65	0100 0001	41	LATIN CAPITAL LETTER A			A
66	0100 0010	42	LATIN CAPITAL LETTER B			B
67	0100 0011	43	LATIN CAPITAL LETTER C			C
68	0100 0100	44	LATIN CAPITAL LETTER D			D
69	0100 0101	45	LATIN CAPITAL LETTER E			E
70	0100 0110	46	LATIN CAPITAL LETTER F			F
71	0100 0111	47	LATIN CAPITAL LETTER G			G
72	0100 1000	48	LATIN CAPITAL LETTER H			H
73	0100 1001	49	LATIN CAPITAL LETTER I			I
74	0100 1010	4A	LATIN CAPITAL LETTER J			J
75	0100 1011	4B	LATIN CAPITAL LETTER K			K
76	0100 1100	4C	LATIN CAPITAL LETTER L			L
77	0100 1101	4D	LATIN CAPITAL LETTER M			M
78	0100 1110	4E	LATIN CAPITAL LETTER N			N
79	0100 1111	4F	LATIN CAPITAL LETTER O			O
80	0101 0000	50	LATIN CAPITAL LETTER P			P
81	0101 0001	51	LATIN CAPITAL LETTER Q			Q
82	0101 0010	52	LATIN CAPITAL LETTER R			R
83	0101 0011	53	LATIN CAPITAL LETTER S			S

Dec	Bin	Hex	Description	Abbr	Non-Printable	Printable
84	0101 0100	54	LATIN CAPITAL LETTER T			T
85	0101 0101	55	LATIN CAPITAL LETTER U			U
86	0101 0110	56	LATIN CAPITAL LETTER V			V
87	0101 0111	57	LATIN CAPITAL LETTER W			W
88	0101 1000	58	LATIN CAPITAL LETTER X			X
89	0101 1001	59	LATIN CAPITAL LETTER Y			Y
90	0101 1010	5A	LATIN CAPITAL LETTER Z			Z
91	0101 1011	5B	LEFT SQUARE BRACKET			[
92	0101 1100	5C	REVERSE SOLIDUS			\
93	0101 1101	5D	RIGHT SQUARE BRACKET			]
94	0101 1110	5E	CIRCUMFLEX ACCENT			^
95	0101 1111	5F	LOW LINE			_
96	0110 0000	60	GRAVE ACCENT			`
97	0110 0001	61	LATIN SMALL LETTER A			a
98	0110 0010	62	LATIN SMALL LETTER B			b
99	0110 0011	63	LATIN SMALL LETTER C			c
100	0110 0100	64	LATIN SMALL LETTER D			d
101	0110 0101	65	LATIN SMALL LETTER E			e
102	0110 0110	66	LATIN SMALL LETTER F			f

Dec	Bin	Hex	Description	Abbr	Non-Printable	Printable
103	0110 0111	67	LATIN SMALL LETTER G			g
104	0110 1000	68	LATIN SMALL LETTER H			h
105	0110 1001	69	LATIN SMALL LETTER I			i
106	0110 1010	6A	LATIN SMALL LETTER J			j
107	0110 1011	6B	LATIN SMALL LETTER K			k
108	0110 1100	6C	LATIN SMALL LETTER L			l
109	0110 1101	6D	LATIN SMALL LETTER M			m
110	0110 1110	6E	LATIN SMALL LETTER N			n
111	0110 1111	6F	LATIN SMALL LETTER O			o
112	0111 0000	70	LATIN SMALL LETTER P			p
113	0111 0001	71	LATIN SMALL LETTER Q			q
114	0111 0010	72	LATIN SMALL LETTER R			r
115	0111 0011	73	LATIN SMALL LETTER S			s
116	0111 0100	74	LATIN SMALL LETTER T			t
117	0111 0101	75	LATIN SMALL LETTER U			u
118	0111 0110	76	LATIN SMALL LETTER V			v
119	0111 0111	77	LATIN SMALL LETTER W			w
120	0111 1000	78	LATIN SMALL LETTER X			x
121	0111 1001	79	LATIN SMALL LETTER Y			y

Dec	Bin	Hex	Description	Abbr	Non-Printable	Printable
122	0111 1010	7A	LATIN SMALL LETTER Z			z
123	0111 1011	7B	LEFT CURLY BRACKET			{
124	0111 1100	7C	VERTICAL LINE			!!
125	0111 1101	7D	RIGHT CURLY BRACKET			}
126	0111 1110	7E	TILDE			~
127	0111 1111	7F	DELETE (DEL)	DEL		

Index

A

C

D

E

F

H

I

M

N

O

R

S

T

U

W

X