

IBM Sterling Gentran:Server for Windows



Application Integration User Guide

Version 5.3.1

IBM Sterling Gentran:Server for Windows



Application Integration User Guide

Version 5.3.1

Note

Before using this information and the product it supports, read the information in “Notices” on page 201.

This edition applies to the 5.3.1 version of IBM Sterling Gentran:Server for Microsoft Windows and to all subsequent releases and modifications until otherwise indicated in new editions.

© Copyright IBM Corporation 1996, 2024.

US Government Users Restricted Rights – Use, duplication or disclosure restricted by GSA ADP Schedule Contract with IBM Corp.

Contents

Chapter 1. Application Integration

Basics 1

About the Application Integration Subsystem	1
Inbound Translation Process	1
Outbound Translation Process	3
Application File Format.	5
EDI File Format	6
About the Application Integration User Interface	7
Global Display Settings	7
Customizing Global Display Options	8
Customizing Global Colors	8
Customizing Global Fonts	9
Customizing Global Display of Links.	10
Customizing the Auto-increment Map Version.	10
Making the Two Sides of the Map Equal.	11
Setting the Default Date/Time Format	11
Customizing Global Confirmation Options	14
Map Building Process	14
Map Types	16

Chapter 2. Map Design 17

Preparation and Analysis	17
Analysis of Your Application File Format	17
Analysis of the Customer EDI File.	17
Reconcile Your Application File and the Customer EDI File	18
Creating a Map	18
Translation Object Details Dialog Box.	20
Defining Translation Object Details	24
Loading a File Definition	24
Saving a File Definition	25
Activating EDI Map Components	25
Using Auto Trim	26
Promoting Groups and Repeating Segments	26
Splitting Groups and Repeating Segments	27
Using Copy, Cut, and Paste	27
About Fixed-format Files	28
Changing Record Delimiters.	29
Changing Decimal Points.	29
Creating the First Record	29
Creating Subsequent Records	30
Temporary Records.	31
Creating Temporary Records--Example	32
Creating a Group	33
Creating Fields	34
About EDI Files	36
Verifying EDI Delimiters	36
Modifying Group Properties.	37
Modifying Segment Properties	37
About Loop Start and Loop End Segments	39
Defining an LS Segment Inbound	39
Defining an LE Segment Inbound	40
Defining an LS Segment Outbound	40
Defining an LE Segment Outbound	41
Modifying Composite Properties	41

Modifying Element Properties	41
Defining and Modifying Relational Conditions	43
About CII Files	44
Modifying CII File Properties	44
Creating a Group	45
Creating a TFD	45
About Data Formatting	47
String Type Fields and Syntax Tokens	47
Creating and Editing Syntax Tokens for Western European Languages	49
Creating and Editing Syntax Tokens for East Asian Languages	50
Deleting Syntax Tokens	51
Deleting a Character Range	51
Using Syntax Tokens	52
Using the Number Type	52
Using the Date/Time Type	54
Completing a Map	54
Creating Simple Links.	54
Binary Segments.	55
Setting up the Input Side	56
Setting up the Output Side	56
Setting the Auto-register Option	57
Compiling a Map	57
Printing a Mapping Report	58

Chapter 3. Standard Rules 61

About Standard Rules	61
Standard Rule Tab - select Function	61
Table access examples	63
Using Information from the Partner Definition.	64
Using Information from a Cross-reference Table	65
Using Information from Location Tables	67
Using Information from Lookup Tables	67
Standard Rule Tab - update Function.	68
Document Name and Reference Data.	69
Setting up the Document Name	69
Setting up the Reference Data	70
Standard Rule Tab - Use System Variable Function	70
Using the System Date and Time	70
Standard Rule Tab - Use Constant Function.	71
Using a Constant in a Map	72
Defining a Qualifying Relationship	72
Defining and Editing Literal Constants	73
Deleting Literal Constants	73
Mapping Literal Constants	74
Generating Qualifiers	74
Standard Rule Tab - Loop Count Function	75
Using the Loop Count Function	75
Standard Rule Tab - Use Accumulator Function	75
Counting Line Items	78
Calculating Hash Totals	80
Multiplying Quantity Invoiced by Unit Price	81
Generating a Running Total of Extended Price.	82
Loading a Running Total of Extended Price.	83

Standard Rule Tab - Use Code Function	84
Defining and Modifying a Code List	86
Deleting a Code List	86
Deleting a Code List Entry	86
Importing a Code List	87
Exporting a Code List	87
Loading a Code List Table from the Standard	88
Copying and Pasting Code Lists	88
Validating Data Against Code List Tables	89
Mapping Code Item Descriptions	90

Chapter 4. The Import Process 91

About the Import Process	91
Creating a System Import Map	93
How to Define the Six-Field Key	94
Defining the Partner Key	94
Defining the Standard Field	95
Defining the Version Field	95
Defining the Transaction Field	95
Defining the Release Field	96
Defining the Test/Production Field	96
How to Define the Alternate Key	97
Defining the Partner Key	97
Defining the Application ID Field	98
Defining the Application Alias Value Field	98
Compiling the System Import Translation Object	99

Chapter 5. The Export Process 101

About the Export Process	101
Inbound Process Before Exporting Data	101
Setting up the Export Process	102
Using Supplementary Envelope Information	102

Chapter 6. Extended Rules 105

About Extended Rules	105
Declarations and Initialization	105
Statements	106
When Extended Rules are Processed	107
How to Define Extended Rules	109
Defining a Session Rule	109
Defining a Map Component Rule	110
Extended Rule Syntax	111
Keywords and Commands	111
Operators	112
Symbols	113
Extended Rule Functions	115
About the Extended Rule Functions	115
atoi	120
aton	120
auditlog	121
begin ... end	122
break	122
cerrror	123
concat	126
continue	127
count	127
createobject	128
date	128
delete	130
deleteobject	130

empty	130
exec	131
exist	132
fseek	133
ftell	133
get	134
getiid	134
if ... then ... else	135
index	136
insert	137
left	138
len	138
messagebox	139
mid	140
ntoa	141
param	142
queryobject	143
readblock	143
readbytes	144
right	145
select	146
set	146
strdate	147
strstr	148
unreadblock	149
update	150
while ... do	152
winexec	152
writeblock	154
writebytes	155
select and update Options	156

Chapter 7. User Exits 163

ActiveX and User Exit Functions	163
About ActiveX Technology	163
About User Exits	165
Examples of User Exits	167
Examples of Automation Servers	170
Creating a User Exit	172

Chapter 8. GentranEx.DLL 175

About GentranEx.DLL	175
About Database Access	175
About Debugging	176
About Rules Extensions	178

Chapter 9. Translator Command Line Interface 181

Command Line Syntax	181
-------------------------------	-----

Chapter 10. Error Messages 183

About Error Messages	183
Compile Error Messages	183
Sterling Gentran:Server Error Messages	188
About Translator Report Error Messages	191
Translator Error Messages	192
User Level Error Messages	195
Core Translator Error Messages	198
Core Translator Error Messages: Sending Batch Data Using the NCPDP Data Record Field	199

Notices	201	Index	205
--------------------------	------------	------------------------	------------

Chapter 1. Application Integration Basics

About the Application Integration Subsystem

The IBM® Sterling Gentran:Server® for Microsoft Windows Application Integration subsystem enables you to generate import, export, and document turnaround translation objects. It enables you to translate your application files to EDI standard formats for documents you send to your trading partners (outbound mapping) and to translate EDI standard formats to your application format for documents that you receive from your trading partners (inbound mapping).

Once you generate a translation object using the Sterling Gentran:Server Application Integration subsystem (or the Sterling Gentran:Server Forms Integration subsystem), you must register the translation object with Sterling Gentran:Server.

In addition, you must establish correct trading relationships for all trading partners from which you receive or to which you send data. During the process of establishing a trading relationship, you need to specify the name of the appropriate translation object that the translator uses to translate the data.

Inbound Translation Process

The following diagram illustrates the inbound translation process:

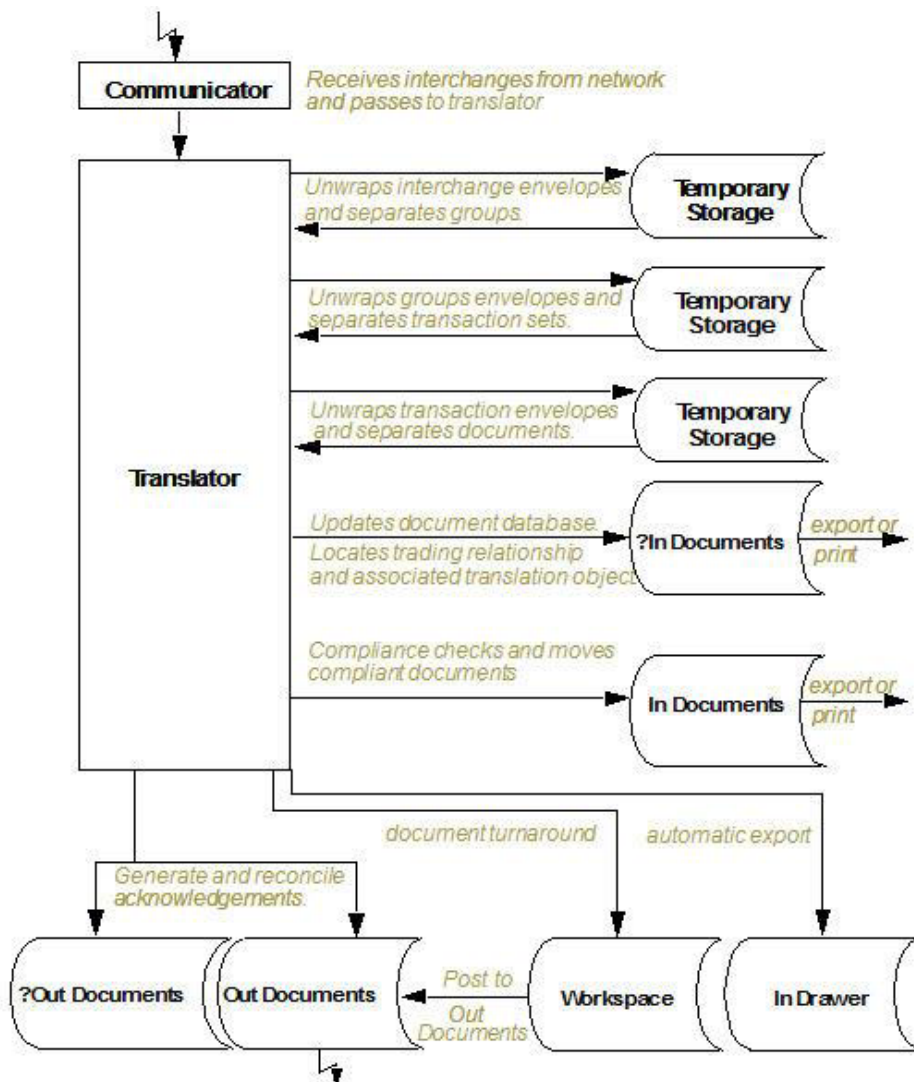


Table 1. Inbound Translation Process

Stage	Description
1	The Communicator receives interchanges from your trading partners through a network.
2	The Communicator passes the interchanges to the translator.
3	The translator uses a system interchange break translation object to unwrap the interchange envelopes and separate each group into temporary storage.
4	The translator uses a system group break translation object to unwrap the group envelopes and separate each transaction set into temporary storage.
5	<p>The translator uses a system transaction break translation object to do the following:</p> <ul style="list-style-type: none"> • Unwrap the transaction envelopes. • Separate each document into a separate file on the system data store. • Write a record to the database with reference information about the document.

Table 1. Inbound Translation Process (continued)

Stage	Description
6	<p>Does the translator locate a trading relationship for each document?</p> <ul style="list-style-type: none"> • If yes (a trading relationship is located), the translator attempts to identify the export, document turnaround, or print translation object associated with that relationship. If the translator locates a trading relationship and translation object, it uses that translation object to compliance check the document. • If no (the translator does not locate the trading relationship or translation object), the document is marked as not compliant and is moved to ?In Documents.
7	<p>Is the document compliant with the EDI standard?</p> <ul style="list-style-type: none"> • If yes, the translator changes the document status to compliant and moves the document to In Documents. • If no, the document remains in ?In Documents. The translator writes a detailed error report to help you determine the problem that was encountered.
8	<p>In the trading relationship, if you specify that the system needs to generate a functional acknowledgement for a document, the translator uses the system acknowledgement translation object to generate the acknowledgement.</p> <ul style="list-style-type: none"> • Compliant acknowledgements are moved to Out Documents to be sent. • Non-compliant acknowledgements are moved to ?Out Documents. If an error occurred with the acknowledgement translation object, the acknowledgement is also moved to ?Out Documents. <p>The translator also reconciles acknowledgements if you receive an acknowledgement-type transaction (such as 997 or CONTRL).</p>
9	<p>If you specified either automatic export or automatic turnaround in the trading relationship, the translator uses the specified export or document turnaround translation object to either export or generate the appropriate response document.</p>

Outbound Translation Process

The following diagram illustrates the outbound translation process:

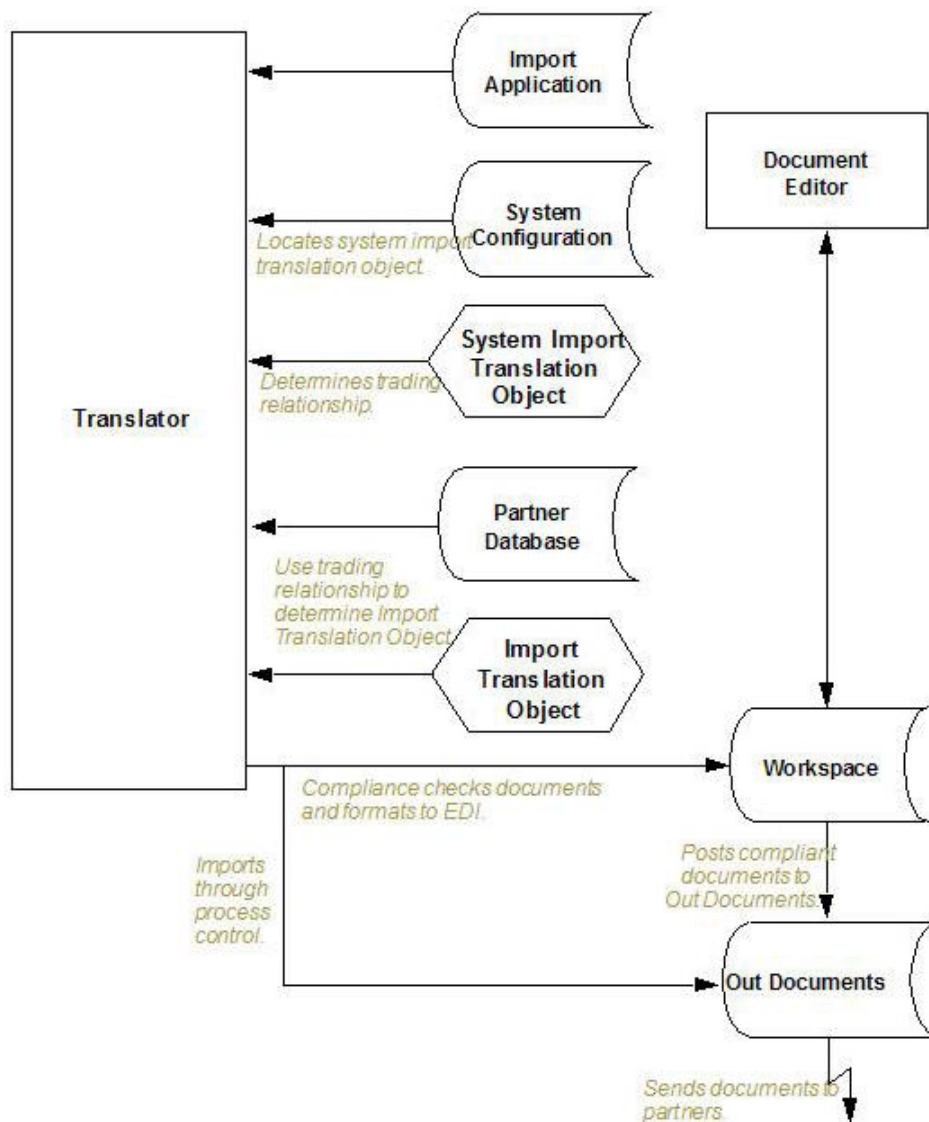


Table 2. Outbound Translation Process

Stage	Description
1	<p>Use one of the following three processes to initiate an outbound translation:</p> <ul style="list-style-type: none"> Import a file through the process control system using a timed or polled session. This writes all valid documents to the database with a compliant status and moves the documents to Out Documents. Invalid documents are marked with a non-compliant status and are moved to ?Out Documents. Import an application file. Documents that you import manually are located in the Workspace. Use the Document Editor to enter documents (if there is an appropriate data entry translation object registered with Sterling Gentran:Server). These documents are located in the Workspace.
2	<p>If you import a file, the translator checks the import definitions from the system configuration to match the file name with a system import translation object.</p>




Table 2. Outbound Translation Process (continued)

Stage	Description
3	The translator uses the system import translation object to determine which trading relationship (established in Partner Editor) corresponds to each document in the application file, so the system knows which import map to use to process the document.
4	The translator ascertains which import translation object is specified in the trading relationship.
5	The translator uses the import translation object to compliance check the document. If the document is compliant (valid), it is marked OK. If the document is not compliant (invalid), it is marked NotOK.
6	If there is another document remaining in the import file, the translator repeats steps 3 - 5 until all documents are processed.
7	If you manually import a file through the EDI Manager or use the Document Editor, you need to post the compliant document to Out Documents. Once documents are located in Out Documents, they can be sent using the process control system or by using the EDI Manager transmit option.

Application File Format

If you are creating an import or export map, you must define your application to the Application Integration subsystem. In Sterling Gentran:Server terminology, your application file is also referred to as a fixed-format file or a positional file. Your application file must contain all the information that you need to either extract from your partner's document (if the map is inbound) or send to your partner (if the map is outbound).

This table describes the map components that you use to define your application file.

Component	Icon	Description
Group		This is a looping structure that contains related records and/or groups that repeat in sequence until either the group data ends or the maximum number of times that the loop is allowed to repeat is exhausted. If you create a group that is subordinate to another group (a subgroup), this corresponds to a nested looping structure (a loop within a loop). The application (positional) file is a group and therefore, it is visually represented the same way as other groups and subgroups in the Application Integration subsystem.
Record		Contains a group of related fields. A record can occur once or can repeat multiple times.
Field		This is the smallest piece of information defined in the application file. A field is the application map component that is mapped (linked) to a corresponding EDI element. When an element contains a standard rule a black asterisk appears to the right of the element icon.

Notes:

- When a field has a mapping operation performed against it, a red checkmark is displayed over the field icon.
- When a field contains a standard or extended rule a black asterisk appears to the right of the element icon.
- When a group contains an extended rule, a yellow asterisk appears to the right of the group icon.

Before you define your application file format, you should obtain a layout of the necessary records, fields, and groups. Each map component is arranged sequentially in the order that it is most logical for the system to process. Therefore, each level of your application file must be created sequentially. For example, your application file contains records and groups. The records contain fields and the groups contain records and/or subgroups. This means that you must create records and groups before you create the subordinate fields.

EDI File Format

The EDI file must contain all the information that you expect to receive from your partner (if the map is inbound) or need to send to your partner (if the map is outbound).

The Application Integration subsystem generates an EDI file for you, based on the standard (agency), version, transaction set, and release (for TRADACOMS only) that you selected. The system includes all the groups, segments, composites, and elements that are defined by the standards agency for the version of the document you selected. If you are creating an import or export map, you typically need to customize the system-generated EDI file by modifying the properties of the map components and using specialized Sterling Gentran:Server functions to manipulate the EDI file structure.


The specific EDI map components that you use depends on the type of map you are creating. This includes the standard, version, and transaction set (document) selected, and which groups, segments, composites, and elements your company requires. We recommend that you determine which map components you are using before generating or defining an EDI file.




The system activates all of the groups, segments, composites, and elements that are defined as mandatory by the standard. The system does not enable you to deactivate the mandatory groups, segments, composites, and elements. By default, Sterling Gentran:Server displays active map components with a black font, and inactive map components with a grey (dimmed) font.

When translating data, the system does not process groups, segments, composites, and elements (or records and fields) that are not activated. Therefore, you must activate the groups, segments, composites, and elements that are not defined as mandatory by the standard, but that you have determined that you need to use in mapping.

If you want to use a specialized version of an EDI standard that is not available in the Sterling Gentran:Server standards database, it may be appropriate for you to define the EDI file yourself.

This table describes the map components Sterling Gentran:Server uses to define the EDI file.

Component	Icon	Description
Group		This is a looping structure that contains related segments and/or groups that repeat in sequence until either the group data ends or the maximum number of times that the loop is allowed to repeat is exhausted. Groups are defined by the EDI standards. A group that is subordinate to another group is a subgroup (this corresponds to a nested looping structure – a loop within a loop). The EDI file is a group and is visually represented the same way as other groups and subgroups in the Application Integration subsystem.

Component	Icon	Description
Segment		Contains a group of related elements or composite data elements that combine to communicate useful data. Segments are defined by the EDI standards. A segment can occur once or can repeat multiple times.
Composite		This is a data element that contains two or more component data elements or subelements. Composites are defined by the EDI standards that use them (EDIFACT, TRADACOMS, and certain ANSI X12 standards).
Element		This is the smallest piece of information defined by the EDI standards. An element is the EDI map component that is mapped (linked) to a corresponding application field to move data to and from the EDI file.

Notes:

- When an element has a mapping operation performed against it, a red checkmark is displayed over the element icon.
- When a field contains a standard or extended rule a black asterisk appears to the right of the element icon.
- When a group contains an extended rule, a yellow asterisk appears to the right of the group icon.

About the Application Integration User Interface

The Sterling Gentran:Server Application Integration Window allows you to access its functionality in four different ways:

- Select the menu option from the Main Menu Bar.
- Click the button on the Main Toolbar.
- Click the appropriate part of the map.
- Right-click a map component to access a menu of functions available for that map component. The content of these menus varies, depending on the type and level of the selected map component.

When you start Sterling Gentran:Server Application Integration, the Main Menu Bar contains a subset of menu items. The full set of menu items is displayed after you create a new map or open (load) an existing map.

This table lists the parts of the Application Integration Window.

Table 3. Application Integration window parts and functions

Part	Function
Main menu bar	Contains drop-down menus. Unavailable items are dimmed.
Main Toolbar	Enables you to access some of the most common functions in the Application Integration subsystem. Unavailable items are dimmed. Note: The Main Toolbar is dockable, so you can affix it to any edge of the client window.
Status bar	Displays status information about a selection, command, or process, defines menu items as you highlight each item in the menu, and indicates any current keyboard-initiated modes for typing.

Global Display Settings

The Preferences dialog box enables you to set global defaults for Sterling Gentran:Server. The map display options can be set or changed at any time.

The Sterling Gentran:Server Application Integration subsystem allows you to customize the display of maps in several different ways:

- Customizing global display options
- Customizing colors (global option)
- Customizing fonts (global option)
- Displaying links (global option)
- Customizing the auto-increment map version
- Making the two sides of a map equal
- Setting the default date/time format
- Customizing confirmation options

Customizing Global Display Options

You can display descriptions for map components, such as groups, records/segments, and fields/elements.

About this task

Use this procedure to customize the global display options.

Procedure

1. Select **Options > Preferences**.

The system displays the Preferences dialog box. The Preferences dialog box enables you to set global defaults for Sterling Gentran:Server.

2. To turn on the default display of group, record (segment), and field (element) descriptions, select the appropriate options.

Note: Typically, you want to have all the descriptions displayed for reference. However, depending on the size of your monitor, it may be easier to view the entire map if the descriptions are not displayed. You may also want to experiment with shrinking the size of the font for the map.

3. Click **OK** to save changes and exit the Preferences dialog box.

Customizing Global Colors

The Colors feature enables you to select foreground and background colors to visually define the various map or form components. The use of color is optional.

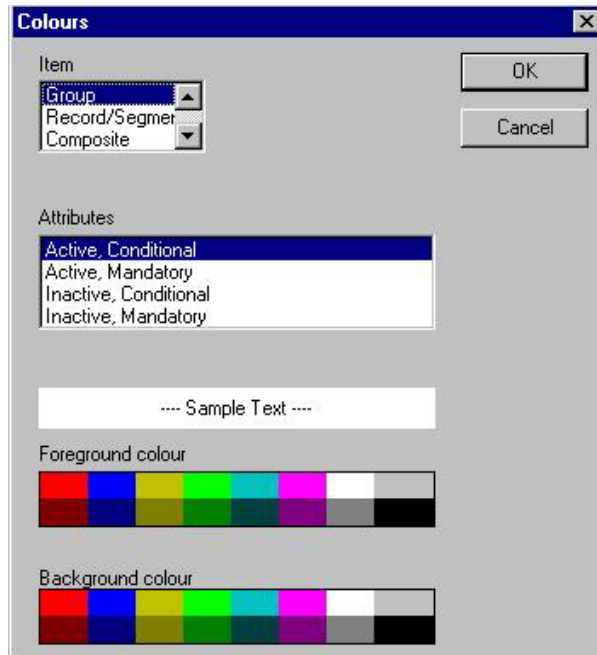
About this task

Use this procedure to customize colors for all maps or forms.

Procedure

1. Select **Options > Preferences**.
2. Click **Colours**.

The system displays the Colours dialog box, as shown below.



3. From the Item list, select the type of component (Group, Record/Segment, Composite, Field/Element).
4. From the Attributes list, further define the type of component by selecting whether it is active or inactive and conditional or mandatory.
5. Select the foreground and background colors.
6. Click **OK** to globally define the selected colors for all maps or forms.

Customizing Global Fonts

The Font feature enables you to globally change the font type, style, and point size that are used in the display of all maps or forms. This gives you the flexibility to shrink the font if you need to view more of the map on your monitor, enlarge the font, or change the type and style to be more easily readable.

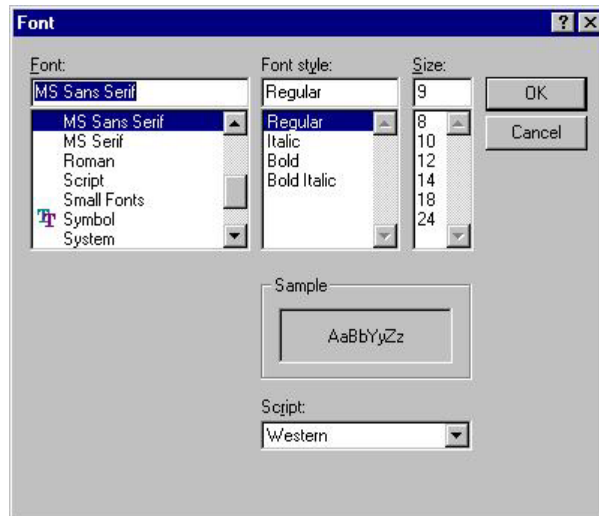
About this task

Use this procedure to customize the display font for all maps.

Procedure

1. Select **Options > Preferences**.
2. Click **Font**.

The system displays the Font dialog box, as shown below.



3. From the Font box, select the type of font. The default is MS Sans Serif.
4. From the Font Style box, select the style. The default is Regular.
5. From the Size box, select the point size. The default is 9.
6. Click **OK** to make the global font change to all maps and forms and exit the Font dialog box.

Customizing Global Display of Links

Mapping Links are the visual lines that connect each field/element/TFD on the Input side of the map to a field/element/TFD on the Output side of the map.

About this task

Use this procedure to customize the global display of mapping links.

Procedure

1. Select **Options > Preferences**.
2. Select the **Links** tab.
3. Select the linking option that you want to set as the default for all maps.
Valid options are:
 - **Show no links:** Do not visually display mapping links.
 - **Show links to or from the currently selected element:** Display only the mapping links for the currently selected field. This option enables you to concentrate on the selected field and removes the confusion of viewing many links at once.
 - **Show links to or from all visible elements:** Display all the mapping links.
4. Click **OK** to save changes and exit the Preferences dialog box.

Customizing the Auto-increment Map Version

Sterling Gentran:Server supports automatically incrementing the version number of a map based on a global option you can set.

About this task

You can set a global option on the Application Integration Preferences dialog box to specify when map version numbers are incremented. The following are the options for automatically incrementing the map version number:

- Never auto-increment.
- Ask whether to increment when saving a map.
- Ask whether to auto-increment when compiling a map.
- Always auto-increment when saving a map.
- Always auto-increment when compiling a map.

Note: The auto-increment function updates the minor version number of the map on the Translation Object Details dialog box, up to 255. If the minor version number exceeds 255, the system updates the minor version number to zero and increases the major version number. For example, version number 1.255 will be auto-incremented to 2.0.

Use this procedure to set the auto-increment map version function.

Procedure

1. Select **Options > Preferences**.
2. Select the **Version** tab.
3. Select the appropriate option and click **OK**.

The system sets the global auto-increment option you selected.

Making the Two Sides of the Map Equal

The Equalize function enables you to reinstate the two sides of the map with focus in equal dimensions. The use of the Equalize function is optional.

About this task

When you open a map, the Input and Output sides of a map are displayed in equal dimensions. After manipulating a map and moving the center bar that divides the display of the Input and Output sides, you may wish to again make the two sides equal.

Use this procedure to make the two sides of a map equal.

Procedure

Select **View > Equalize**.

This moves the center dividing bar between the two sides of the map with focus, so that the two sides of the map are equally proportioned.

Setting the Default Date/Time Format

The Date Formats global option changes the default date format for all maps or forms. However, the format of the existing date fields do not change; the default is only used for new maps or forms.

About this task

You typically establish the default date format for all date fields one time only. This default is used when EDI documents are initially loaded from the standard. This default can be overridden on the Field Properties dialog box.

Use this procedure to establish the default date format.

Procedure

1. Select **Options > Preferences**.
2. Select the **Standard Formats** tab.
3. From the Six-character dates and Eight character dates lists, select the appropriate six-character and eight-character default date formats.
4. Click **OK** to accept the default date formats and exit the Preferences dialog box.

Tip: To change the order in which the date formats appear in the Six-character dates and Eight character dates lists or to add a new date format to the lists, select **Options > Date Formats**.

Date/Time Formats

The Date Formats global option changes the default date format for all maps or forms. However, the format of the existing date fields do not change; the default is only used for new maps or forms.

This table lists the valid date and time formats.

Format	Description
YYMMDD	Two-digit year, two-digit month, two-digit day
MMDDYY	Two-digit month, two-digit day, last two digits of year (example: 121599)
YYYYMMDD	Four-digit year, two-digit month, two-digit day (example: 19991215)
DDMMYYYY	Two-digit day, two-digit month, four-digit year (example: 15121999)
MMDDYYYY	Two-digit month, two-digit day, four-digit year (example: 12151999)
DDMMYY	Two-digit day, two-digit month, last two digits of year (example: 151299)
YYMMMDD	Last two digits of year, three-letter abbreviation of the month, two-digit day (example: 99JAN02)
DDMMYY	Two-digit day, three-letter abbreviation of the month, last two digits of year (example: 02JAN99)
MMMDDYY	Three-letter abbreviation of the month, two-digit day, last two digits of year (example: JAN0299)
YYYYMMMDD	Four-digit year, three-letter abbreviation of the month, two-digit day (example: 2003JUL04)
DDMMYYYY	Two-digit day, three-letter abbreviation of the month, four-digit year (example: 04JUL2003)
MMMDDYYYY	Three-letter abbreviation of the month, two-digit day, four-digit year (example: JUL042003)
YYDDD	Last two digits of year, three-digit Julian day (example: 99349 for the 349th day of 1999)

Format	Description
DDDDYY	Three-digit Julian day, last two digits of year (example: 34999)
YYYYDDDD	Four-digit year, three-digit Julian day (example: 1999349)
DDDDYYYY	Three-digit Julian day, four-digit year (example: 3491999)
YY/MM/DD	Last two digits of year, separator, two-digit month, separator, twodigit day (example: 99/12/05)
DD/MM/YY	Two-digit day, separator, two-digit month, separator, last two digits of year (example: 05/12/99)
MM/DD/YY	Two-digit month, separator, two-digit day, separator, last two digits of year (example: 12/15/99)
YYYY/MM/DD	Four-digit year, separator, two-digit month, separator, two-digit day (example: 1999/12/15)
DD/MM/YYYY	Two-digit day, separator, two-digit month, separator, four-digit year (example: 15/12/1999)
MM/DD/YYYY	Two-digit month, separator, two-digit day, separator, four-digit year (example: 12/15/1999)
YY/MMM/DD	YY/MMM/DD Two-digit year, separator, three-letter abbreviation of the month, separator, two-digit day (example: 99/JUL/20)
DD/MMM/YY	Two-digit day, separator, three-letter abbreviation of the month, separator, two-digit year (example: 20/JUL/99)
MMM/DD/YY	Three-letter abbreviation of the month, separator, two-digit day, separator, two-digit year (example: JUL/20/99)
YYYY/MMM/DD	Four-digit year, separator, three-letter abbreviation of the month, separator, two-digit day (example: 2003/JUL/25)
DD/MMM/YYYY	Two-digit day, separator, three-letter abbreviation of the month, separator, four-digit year (example: 25/JUL/2003)
MMM/DD/YYYY	Three-letter abbreviation of the month, separator, two-digit day, separator, four-digit year (example: JUL/25/2003)
YY/DDDD	Last two digits of year, separator, three-digit Julian day (example: 99/349)
DDDD/YY	Three-digit Julian day, separator, last two digits of year (example: 349/99)
YYYY/DDDD	Four-digit year, separator, three-digit Julian day (example: 1999/349)
DDDD/YYYY	Three-digit Julian day, separator, four-digit year (example: 349/1999)
MONTH	Month (example: December)
DAY	Day of the week (example: Friday)
HHMM	Two-digit hour, two-digit minutes (example: 0330 for 30 minutes past 3 o'clock)
HHMMSS	Two-digit hour, two-digit minutes, two-digit seconds (example: 033045 for 30 minutes and 45 seconds past 3 o'clock)
HH:MM	Two-digit hour, separator, two-digit minutes (example: 03:30)
HH:MM:SS	Two-digit hour, separator, two-digit minutes, separator, two-digit seconds (example: 03:30:45)

Format	Description
YYYYMMDDTHH MMSS.mmmZ	ISO-8601 format: Four-digit year, two-digit month, two-digit day, T (time) indicator, two-digit hour, two-digit minutes, two-digit seconds in Universal Time (also called Zulu Time or Greenwich Mean Time), Z (Zulu time) indicator (example: 20031209T123000.000Z)
YYYYMMDDZ	ISO-8601 date format: Four-digit year, two-digit month, two-digit day, Z (Zulu time) indicator (example: 20031209Z)
MM/DD/YY HH:MM:SS	Two-digit month, separator, two-digit day, separator, last two digits of year, two-digit hour, separator, two-digit minutes, separator, two-digit seconds (example: 12/15/99 03:30:45)
YYMMDD HHMMSS	Last two digits of year, two-digit month, two-digit day, two-digit hour, two-digit minutes, two-digit seconds (example: 991025 033045)
YYYY-MMDDTHH: MM:SS	Four-digit year, separator, two-digit month, separator, two-digit day, T represents a blank separator, two-digit hour, separator, twodigit minutes, separator, two-digit seconds (example: 2002-02-02 03:30:45)
YYYY-MM-DD	Four-digit year, separator, two-digit month, separator, two-digit day (example: 2002-02-02)
YYYY-MM	Four-digit year, separator, two-digit month (example: 2002-02)
YYYY	Four-digit year (example: 2002)
--MM-DD	Two dashes, two-digit month, separator, two-digit day (example: - -12-02)
---DD	Three dashes, two-digit day (example: ---02)

Customizing Global Confirmation Options

The Confirmations tab on the Preferences dialog allows you specify when you want confirmation messages displayed.

About this task

Use this procedure to set the confirmation options.

Procedure

1. Select **Options > Preferences**.
2. Select the **Confirmation** tab.
3. Set the global confirmation options by either selecting **Confirm everything** (displays all confirmation messages) or by selecting individual confirmation messages by action performed.
4. Click **OK** to save the confirmation options.

The system sets the confirmation options you selected.

Map Building Process

This topic provides an overview of the map creation process.

This table defines how to proceed with building a map using the Sterling Gentran:Server Application Integration subsystem.

Stage	Description
1	<p>Prepare and analyze.</p> <ul style="list-style-type: none"> Obtain a layout of your application file and determine how it corresponds with the EDI standard you are using. Determine how you move data to or from each application field.
2	<p>Set global defaults (one time only).</p> <p>The first time you use the Sterling Gentran:Server Application Integration subsystem, you should establish the default date format that the system uses. See <i>Setting the Default Date Format</i> for more information.</p>
3	<p>Create a new map.</p> <p>See <i>Creating a Map</i> for more information.</p>
4	<p>Activate the appropriate EDI groups, segments, and elements or auto trim based on a sample EDI file.△</p> <p>Alternatively, define your CII EDI file.</p> <p>For more information, see the following topics:</p> <ul style="list-style-type: none"> Activating EDI Map Components Using Auto Trim About CII Files
5	<p>Define your application.</p> <p>See <i>About Fixed-format Files</i> for more information.</p>
6	<p>Map the appropriate data for each application field.</p> <p>For more information, see the following topics:</p> <ul style="list-style-type: none"> Creating Simple Links the Standard Rules topics the Extended Rules topics
7	<p>Compile the translation object.</p> <p>See <i>Compiling a Map</i> for more information about compiling the translation object and the translation object naming conventions.</p>
8	<p>Print the mapping report. Validate and review the map. Make modifications as needed.</p> <p>See <i>Printing a Mapping Report</i> for more information.</p>
9	<p>Register the translation object with Sterling Gentran:Server.</p> <p>See the <i>IBM Sterling Gentran:Server for Microsoft Windows User Guide</i> for more information.</p>
10	<p>Establish relationships in Sterling Gentran:Server with your trading partners.</p> <p>See the <i>IBM Sterling Gentran:Server for Microsoft Windows User Guide</i> for more information about creating trading relationships.</p>
11	<p>Test the translation object. Obtain test data from your partners and process the data. Verify acknowledgement processing (if applicable). Verify communications with your network.</p>

Map Types

There are several types of maps available in Sterling Gentran:Server.

This table provides brief descriptions of the map types.

Map Type	Description
Import	Used for outbound maps.
Export	Used for inbound maps.
Interchange break	Used in advanced mapping to separate interchanges.
Functional Acknowledgement Inbound	Used in advanced mapping to reconcile functional acknowledgements.
Functional Acknowledgement Outbound	Used in advanced mapping to generate functional acknowledgements.
System Import Header	Used to determine which trading relationship (established in Partner Editor) corresponds to each document in the application file, so the system knows which import translation object to use to process the document.
Turnaround	Used for EDI to EDI maps.
Transaction build	Used in advanced mapping to build transaction envelopes.
Transaction break	Used in advanced mapping to separate documents.
Functional group build	Used in advanced mapping to build functional group envelopes.
Functional group break	Used in advanced mapping to separate functional groups.
Interchange build	Used in advanced mapping to build interchange envelopes.
Direct Map	Do not use this map type. Any existing maps that are configured with the Direct Map type should be changed to another type.

Chapter 2. Map Design

Preparation and Analysis

The first step in creating a map is the analysis of the mapping requirements. This is the most important step in creating a successful map. If the analysis you perform is complete, you have all the information you need to create the map in an efficient and logical manner. If you skip this step and start directly creating your map, it may take you longer and result in an invalid map due to oversights and omissions.

This table lists the stages of mapping analysis. These apply whether you are translating data inbound or outbound.

Stage	Description
1	Analyzing your application file format
2	Analyzing your partners EDI file format
3	Correlating your application and the EDI file formats

After you complete these steps, you can begin creating the map.

Analysis of Your Application File Format

The first step of mapping analysis is analyzing your application file format, because this is probably the component that is the most familiar to you.

Your application file format contains all the information that you need to extract from the purchase order your partner sends you, so that your system can correctly process the purchase order, and your company can fill the order correctly.

You must define your application file format to Sterling Gentran:Server. If your company has an existing application file format, you should obtain the record layout from the appropriate person. If you did not have an existing application file format, you need to create one by determining which fields are necessary to process the data correctly, and then grouping the fields logically under records.

Analysis of the Customer EDI File

To analyze the EDI file, you must first determine what your trading partners are sending you. You and your partners need to agree on which standard, version, and transaction set you are using. It is important to know the information that your partners are sending you and what data is contained in each element used.

It is very helpful to review the EDI standards that you are using before analyzing the EDI file. Obtain an EDI standards manual for each standard and version that you are using. Standards manuals are available from EDI standard agencies.

After discussing with your partners what they are sending, determining which segments and elements your company requires, and reviewing the EDI standard, you can list the map components that you need to make available for use (activate).

Reconcile Your Application File and the Customer EDI File

To reconcile your application file format with the EDI file, you must identify each application file with its corresponding element in the EDI file and select a method for mapping it. To map information to a field, you use linking, standard rules, extended rules, or a combination of all three.

Linking (simple mapping) enables you to map a field or element from the input side of the map to a field or element on the output side of the map. The link between two map components (fields) is visually represented with a line connecting the two fields.

Standard rules give you access to mapping operation functions that are more complex than simple linking, but less involved than extended rules.

Extended rules enable you to use a Sterling Gentran:Server proprietary programming language to perform virtually any mapping operation you require.

Additionally, you may determine that you need to establish temporary storage (work) areas for the map to handle items such as Ship To and Bill To name and address information, which is extracted using extended rules from a group in the EDI data.

Note: Determine which mapping operations are required on a field-by-field basis for your application file.

Creating a Map

You can create four different types of maps: System Import (containing the header information for an Import map), Import (for Outbound Maps), Export (for Inbound Maps), or Turnaround (for EDI to EDI maps). The initial procedures for creating a new map of each of the four types is the same.

About this task

Use this procedure to create a map.

Procedure

1. Select **File > New**.

The system displays the New Map Wizard.

2. Enter the following information and click **Next**:

- Select the type of map
- Type the unique name of the map. The system adds the .MAP extension.
- Type your name if it differs from the user name prompted by the system.

The system displays the New Map Wizard - Input Format dialog box.

Note: You need to complete the format for the Input side of the map (Steps 3 - 7). This is the format of the data that is translated by the Sterling Gentran:Server system.

3. For the input side of the map, do one of the following:
 - Create a data format using a syntax that you define - go to step 4.
 - Load the data format from a saved definition - go to step 7.
4. Select one of the following input format options:

- **Delimited EDI** (Electronic Data Interchange file)
- **Positional** (VDA, GENCOD, application files)
- **CII** (Japanese standard)
- **CII Positional** (for CII Build/Break maps)

Notes:

- For Inbound maps (Export), the Input Format Type is CII or EDI.
 - For Outbound maps (System Import or Import), the Input Format Type is Positional.
 - For Turnaround maps, the Input Format Type is EDI or CII.
5. If you selected **Delimited EDI** or **CII** and want to customize the format, click **Customize** and continue with the next step. Otherwise, click **Next** and continue with step 9.

The system displays the New Delimited EDI Wizard or New CII Wizard dialog box. The Delimited EDI and CII wizards enable you to create your format from the standards database.

6. Follow the steps for the appropriate dialog box:
- If the dialog is **New Delimited EDI Wizard**, do the following:
 - a. Click **Next**.
 - b. Select the ODBC data source that contains the standards database.
 - c. Select the standards agency, version, transaction set, and (for TRADACOMS) release number and click **Next**.
 - d. Click **Finish**.
 - If the dialog is **New CII Wizard**, do the following:
 - a. Click **Next**.
 - b. Select the ODBC data source that contains the standards database.
 - c. Specify if you want to use Japanese descriptions and click **Next**.
 - d. Select the standards agency, version, transaction set, and (for TRADACOMS) release number and click **Next**.
 - e. Select the appropriate Multi-detail header and click **Next**.
 - f. Click **Finish**.

The system displays the New Map Wizard - Output Format dialog box. Continue with step 8.

7. To load the data format from a saved definition, select **Load the data format from a saved definition** and either enter the path and filename of the saved definition or browse to navigate and select the file.

Note: You can select either a .DDF or .IFD file. If the DDF is invalid, the system displays a message box explaining the problem and terminates the import.

8. For the output side of the map, do one of the following:
- Load the data format from a saved definition - go to step 12.
 - Create a data format using a syntax that you define - go to step 9.
9. Select one of the following input format options:
- **Delimited EDI** (Electronic Data Interchange file)
 - **Positional** (VDA, GENCOD, application files)
 - **CII** (Japanese standard)
 - **CII Positional** (for CII Build/Break maps)

Notes:

- For Inbound maps (Export), the Output Format Type is CII or EDI.
 - For Outbound maps (System Import or Import), the Output Format Type is Positional.
 - For Turnaround maps, the Output Format Type is EDI or CII.
10. If you selected **Delimited EDI** or **CII** and want to customize the format, click **Customize** and continue with the next step. Otherwise, go to step 13.
- The system displays the New Delimited EDI Wizard or New CII Wizard dialog box. The Delimited EDI and CII wizards enable you to create your format from the standards database.
11. Follow the steps for the appropriate dialog box:
- If the dialog is **New Delimited EDI Wizard**, do the following:
 - a. Click **Next**.
 - b. Select the ODBC data source that contains the standards database.
 - c. Select the standards agency, version, transaction set, and (for TRADACOMS) release number. Click **Next**.
 - d. Click **Finish**.
 - If the dialog is **New CII Wizard**, do the following:
 - a. Click **Next**.
 - b. Select the ODBC data source that contains the standards database.
 - c. Specify if you want to use Japanese descriptions and click **Next**.
 - d. Select the standards agency, version, transaction set, and (for TRADACOMS) release number. Click **Next**.
 - e. Select the appropriate Multi-detail header and click **Next**.
 - f. Click **Finish**.

The system displays the New Map Wizard dialog box. Continue with step 13.

12. To load the data format from a saved definition, select the Load the data format from a saved definition option and either enter the path and filename of the saved definition or browse to navigate and select the file.

Note: You can select either a .DDF or .IFD file. If the DDF is invalid, the system displays a message box explaining the problem and terminates the import.

13. Click **Finish** to load the standards information you selected and create the new map (this may take a few seconds).

The system displays the new map in the Application Integration Window.

What to do next

After you finish creating and saving a new map, you need to define the Input and Output sides of the map. The steps you take are different, depending on whether the map is an Import, System Import, Export, or Turnaround map.

Translation Object Details Dialog Box

The Translation Object Details dialog box enables you to edit the details of the translation object, including the description and version information.

This dialog box enables you to instruct the translator to use the pad character and alignment settings of each string field when reading a positional or CII positional file, to determine how to trim pad characters from the string data.

Translation Object Details

Summary

Author: Sterling Description: PET X 3030 850 Export Translation Object Function: Export

Flags

☐ System ☐ Use Configurable Trimming ☐ Gentran:Server for Windows 2.x Compatible Rule Execution

Version Control

Major version: 1 Minor version: 0 Compiled on: 02-Mar-2000

EDI Associations

	Input	Output
Agency	X12	
Version	003030 ANSI X12 VERSION	
Transaction	850 PURCHASE ORDE	
Release	0	
F Group	PO	

Table 4. Translation Object Details dialog box parts and functions

Part	Function
Author	Identifies the person or department that created the map. Mandatory.
Description	<p>Contains a unique description of the map. Mandatory.</p> <p>This description is used by the system to identify the map. We strongly recommend that you use the following identifying characteristics of the map: the partner that this map is used for, the standard, the version, the type of transaction this map uses, and the direction of the map (for example, Import, Export).</p> <p>For example, MWT X 3030 850 Export is the description of a map used with partner MWT, for an ANSI X.12 version 003030 Purchase Order (850), that is sent Inbound (Export).</p>
Translation Object Function	Verify the type of map from the list. Mandatory.
System	<p>Designates this map as a system translation object (one that will be used internally by the system).</p> <p>You should only select this check box if you are certain that you want to define this translation object as a system translation object. Once a system translation object is registered with Sterling Gentran:Server, it cannot be deleted.</p>

Table 4. Translation Object Details dialog box parts and functions (continued)

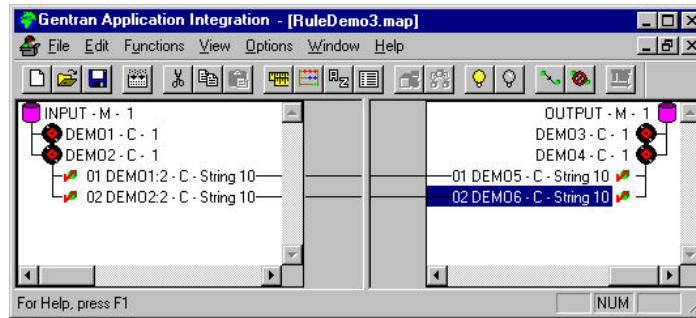
Part	Function
Use Configurable Trimming	<p>Instructs the translator to use the pad character and alignment settings of each string field when reading a positional or CII positional file, to determine how to trim pad characters from the string data.</p> <p>For example, if the alignment setting indicates that the data is aligned in the left of the field, then pad characters are trimmed from the right of the field. If the alignment setting indicates that the data is aligned at the right of the field, then pad characters are trimmed from the left of the field. Finally, if the alignment setting indicates that the field data is aligned in the center of the field, then pad characters are trimmed from both the left and right of the field.</p> <p>If this option is cleared (which is the default) pad characters are always trimmed from both the left and right of each positional string field, which is the current system behavior.</p> <p>Use configurable trimming to preserve either trailing or leading spaces in your application data.</p>
Gentran:Server for Windows 2.x Compatible Rule Execution	Enables you to specify how the translator executes standard and extended rules on the output side of a map.
Major version/Minor version	<p>Enables you to designate different versions of a translation object.</p> <p>The valid values for each box is 0 - 255. These two boxes are available for your use only. They are not used by the system.</p> <p>For example, if you enter 5 in the Major box and 45 in the Minor box, the Translation Object Version Number is 5.45.</p>
Compiled on	Displays the date when the translation object was compiled. This will be blank if the translation object has not been compiled yet.
EDI Associations section	<p>If you selected Delimited EDI for the input and/or output sides of the map, this section contains the EDI information.</p> <p>You should change these boxes only if you want to change the EDI agency, version, transaction, release, or functional group of an existing map. This enables you to copy and alter an existing map.</p>

Rule execution prior to version 3.0

The Translation Object Details dialog box also enables you to specify whether you want the translator to execute standard and extended rules as it did for Sterling Gentran:Server prior to version 3.0. The standard behavior for Sterling Gentran:Server prior to version 3.0 is as follows: if an output record/segment is linked to an input field in a record/segment that does not contain data but is at the same hierarchical level as an input record/segment that does contain data, the standard and extended rules on the output field will be executed even though there is no data present.

In the example diagram below, the extended rule on the DEMO5 field on the output side of the map is executed even though there is no data for the field to

which it is linked (DEMO1:2 on the input side of the map). The rule is executed because there is data present for a field in the DEMO1 record on the input side of the map, and that record is at the same level as DEMO2.



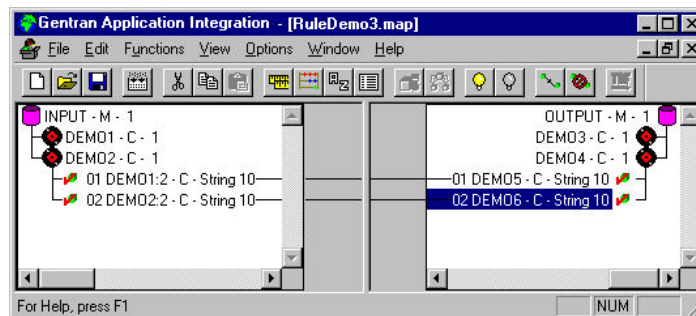
For any map that was compiled with Sterling Gentran:Server prior to version 3.0, the system emulates the previous behavior by automatically selecting **Gentran:Server for Windows 2.x Compatible Rule Execution** on the Translation Object Details dialog box. You can modify this default if you wish, but you would then need to re-map any instances where you want standard or extended rules to run on an output record/segment that does not contain fields that will be populated with data.

Rule execution in version 3.0 and later

In version 3.0, the standard rule execution behavior is as follows: standard and extended rules for a record/segment on the output side of the map are only executed if any of its links populate the output map component with data.

For any map created with Sterling Gentran:Server version 3.x or later, the system default is to clear the **Gentran:Server for Windows 2.x Compatible Rule Execution** option on the Translation Object Details dialog box. We recommend that you do not select this check box for version 3.x and later, as this differs from the prescribed behavior.

In the example diagram below, the extended rule on the DEMO5 field on the output side of the map is not executed because there is no data for the field to which it is linked (DEMO1:2 on the input side of the map). The rule is *not* executed even though there is data present for a field in the DEMO1 record on the input side of the map, and that record is at the same level as DEMO2.



Defining Translation Object Details

Details about the translation object are created when you create the map. However, you may want to change some of those details, such as the description or the version number.

About this task

Use this procedure to specify translation object details.

Procedure

1. Select **Edit > Details**.
The system displays the Translation Object Details dialog box.
2. To change the map description, type the new description in the Description box.
3. To change the compatible rule execution specification, do one of the following:
 - For versions prior to 3.0, select **Gentran:Server for Windows 2.x Compatible Rule Execution**.
 - For versions 3.0 and later, make sure the **Gentran:Server for Windows 2.x Compatible Rule Execution** option is clear.
4. To change the map version, type the appropriate version numbers in the Major and Minor boxes and click **OK**.
The system saves your changes and exits the Translation Object Details dialog box.

Loading a File Definition

Sterling Gentran:Server enables you to load an individual file format definition that you previously saved. This feature provides you with a quick way to build either side of your map.

Before you begin

Important: Loading a file definition replaces the selected side of the map. Please be certain that is your intent before performing this task.

About this task

Use this procedure to load a file format definition.

Procedure

1. Right-click the **File Format** icon (either the input or output side of the map) and select **Open File Definition**.
If you already used Sterling Gentran:Server to create that side of the map, you are prompted with a message that warns you that the existing file format will be replaced.
2. Click **Yes** to continue.
The system displays the Open File Definition dialog box.
3. Navigate to the location where Sterling Gentran:Server is installed and either select the file definition or enter the filename.
4. Click **Open**.
The system loads the selected file format definition.

Saving a File Definition

Sterling Gentran:Server enables you to save an individual file format definition so that you can use it as a guide in future maps. This provides you with a quick way to build either side of your map.

About this task

Use this procedure to save an individual file format definition.

Procedure

1. Right-click the **File Format** icon (either the input or output side of the map) and select **Save File Definition**.

The system displays the Save File Definition dialog box.

2. Navigate to the location where Sterling Gentran:Server is installed and either select the file definition or enter the filename.
3. Click **Save**.

The system saves the file format definition.

Activating EDI Map Components

When Sterling Gentran:Server generates the EDI sides of the map, the system includes all the groups, segments, composites, and elements that are defined by the standard agency for the version of the document you selected. The system activates all the groups, segments, composites, and elements that are defined as mandatory by the standard. The system does not enable you to deactivate the mandatory groups, segments, composites, and elements.

About this task

When translating data, the system does not process groups, segments, composites, and elements (or records and fields) that are not activated. Therefore, you must activate the groups, segments, composites, and elements that are not defined as mandatory by the standard, but that you have determined that you need to use in mapping.

Tip: As an alternative to activation, you can use the Auto Trim feature if you have a sample EDI file defined. See Using Auto Trim for more information.

Use this procedure to activate groups, segments, composites, and elements.

Procedure

1. Select **Functions > Activate**.
2. Double-click each group that contains segments or groups that need to be activated.

Note: As an alternative to opening each map component, you can select **View > Expand All** to open every map component. You can also select a map component and select **View > Expand Branch** to open that map component.

3. Select each inactive group that you need to use. This activates the groups.

Note: If you accidentally click a group, segment, composite, or element that you did not mean to activate, right-click the map component and select **Deactivate**.

4. Select each inactive segment that you need to use.

5. Open each segment that contains composites or elements that need to be activated.
6. Select each inactive composite that you need to use.
7. Open each composite that contains elements that need to be activated.
8. Select each inactive element that you need to use.
9. Once you have activated all the necessary groups, segments, and elements, select **Functions > Activate** to turn activation mode off.

Using Auto Trim

Instead of activating map components manually, you can use the Auto Trim feature to modify the EDI side of the map according to a sample EDI file that you select (you must have previously created this EDI file).

About this task

The Auto Trim feature examines the EDI file that you specify and then activates and deactivates map components on the EDI side of the map, so that the two files match. The sample EDI file must be the same standard, version, and transaction set (message) as the map for Auto Trim to match map components.

To use Auto Trim, you must define the Segment and Element Delimiters. If you do not supply the Tag Delimiter, the system substitutes the Segment Delimiter. If you want to use composite elements, you must specify the Sub Element Delimiter. If you want to use release characters, you must specify the Release Character. See *Verifying EDI Delimiters* for more information.

The sample EDI file must be compliant and must not contain envelope segments.

Use the following procedure to use Auto Trim:

Procedure

1. Right-click the **EDI File** icon on the EDI side of the map and select **Auto Trim**.
The system displays the Auto-trim Sample EDI File dialog box.
2. Navigate to the location where the EDI file is stored (default is GENSRVNT\BIN) and select the sample EDI file that you want to load into the system.
3. Click **Open** to begin the auto trim process.
When auto trim is complete, the system displays a message stating whether or not auto trim was successful.
4. Click **OK** to acknowledge the message.

Promoting Groups and Repeating Segments

The Promote function extracts one iteration (instance) of a group or repeating segment. For maps, this enables you to map unique data from your application file, and/or enter a specialized definition. For forms, it also sets the promote flag and places the active elements in the parent frame.

About this task

For maps, Sterling Gentran:Server specifies that only one-to-one (no loops) or many-to-many (loop) mapping relationships are valid. For more information on this action, see the *Application Integration* tutorials.

Note: The Promote function is only available if a group or repeating segment is selected.

Tip: You can use the Copy and Paste functions (and change the number in the maximum usage box) to accomplish the same task. However, Promote is a specialized function that guarantees the integrity of your EDI structure. Depending on the circumstances, you may want to use the Split function or Copy/Cut and Paste functions instead.

Use this procedure to promote a group or repeating segment.

Procedure

1. Select the group or repeating segment from which you want to extract one iteration.
2. Click **Promote** on the Main Toolbar.
This extracts one iteration (instance) of the looping structure.

Splitting Groups and Repeating Segments

The Split function enables you to split (break) a group or repeating segment into two loops. You typically use this function when you need more than one instance of the same component that occurs multiple times. For more information on this action, see the tutorial.

About this task

Note: The Split function is only available if a group or repeating segment is selected.

Tip: You can use the Copy and Paste functions (and change the number in the maximum usage box) to accomplish the same task. However, Split is a specialized function that guarantees the integrity of your EDI structure.

Use this procedure to split a group or repeating segment.

Procedure

1. Highlight the group or repeating segment from which you want to extract one iteration.
2. Click **Split** on the Main Toolbar.
The system displays the Split dialog box.
3. In the First Loop Entries box, type the number that indicates the sequential number of iterations where you want the group or repeating segment split.
For example, if the X loop repeats a maximum number of 5 times, and you type 2 in the First Loop Entries box, the resulting split generates one X loop that repeats a maximum of 2 times and a second X loop that repeats a maximum of 3 times.
4. Click **OK**.

Using Copy, Cut, and Paste

The Copy, Cut, and Paste functions are typically used to move EDI information in the map or form. You may need to use these functions if you are using an EDI standard differently, or if you need to create nested looping structures.

About this task

You can cut or copy a single component (loop, segment, composite, element, record, or field) and paste it in another location in the map. Copied components retain all the information of the original component. If the copied component contains subordinate components (like a segment contains subordinate elements), the subordinate components are also copied.

You can also cut, copy, and paste a component from one map or form to another.

Use this procedure to cut, copy, and paste a component.

Procedure

1. Select the component that you want to cut or copy.
2. Select **Edit > Copy**.

Note: If you are pasting the component in another map or form, open that map or form, if it is not already open.

3. Select the component that you want the cut or copied selection pasted after.

Note: For maps, if you cut or copied an EDI component, you can only paste that component into the EDI side of a map. Conversely, if you cut or copied a positional component, you can only paste that component into the positional side of a map.

4. Click **Paste** on the Main Toolbar to paste the contents of the Clipboard.

If the component that you selected is a group, Sterling Gentran:Server prompts you to specify whether you want the contents of the Clipboard pasted as a child (subordinate) of the group or pasted at the same level as the group. Select the appropriate option and click **OK**.

About Fixed-format Files

If a side of your map is positional (fixed-format), you must either define your application to Sterling Gentran:Server or load a previously-saved file definition. Your application file must contain all the information that you need to extract from your partner's document (if the map is inbound) or send to your partner (if the map is outbound).

Before you define your application, you should obtain a layout of the necessary records, fields, and groups. The records contain related fields and the groups contain related records. For example, your application contains records and groups. The records contain fields and the groups contain records, subgroups, or both. This means that you must create records and groups before you create the subordinate fields.

Each map component is arranged sequentially in the order that it is most logical for the system to process. Therefore, each level of your application must be created sequentially.

You will use two different Sterling Gentran:Server functions to create the necessary group and records—Create Sub and Insert.

Table 5. When to use the Create Sub and Insert functions

To create a group or record	Right-click the map component above and select
at the same level (equal)	<ul style="list-style-type: none"> • Insert > Group • Insert > Record
that is subordinate to the selected map component	<ul style="list-style-type: none"> • Create Sub > Group • Create Sub > Record

Changing Record Delimiters

About this task

Use this procedure to set the record delimiters.

Procedure

1. In Application Integration, open the positional file.
2. Right-click the file format icon and select **Properties**.
3. In the **Record** tab, change the **Record Delimiters 1** and **Record Delimiters 2** as needed.
4. Click **OK**. You must click OK, even if the delimiter fields are empty (to reset the delimiters).
5. Save and compile the map.

Changing Decimal Points

To specify that each decimal point in your data is defined and generated as a something other than the default period, change the options on the Decimal Point tab.

About this task

Use this procedure to change the default decimal point setting:

Procedure

1. In Application Integration, open the positional file.
2. Right-click the file format icon and select **Properties**.
3. Click the **Decimal Point** tab.
4. Select **Define Decimal Point**.
5. Type the value you want to use as a decimal point (for example, a comma). This resets the decimal point default to use what you specified instead of a period.
6. Click **OK**.
7. Save and compile the map.

Creating the First Record

The first record in your application is generally the header record. The first record is subordinate to the icon in the application side of the map. This shows how the records (and groups) are contained within it.

About this task

Tip: You should include a header record in each application file. The header record is mandatory and does not repeat.

Use this procedure to create the first application record.

Procedure

1. Select the Positional File icon (output for an inbound map and input for an outbound map).
2. Select **Edit > Create Sub > Record**.
The system displays the Positional Record Properties dialog box.
3. On the **Name** tab, type the record name and a description of the record.

Note: The description allows you to differentiate similar records.

4. Select the **Tag** tab.
5. Type the record identification code <TAG>. The record tag for each record enables the system to recognize that record and then determine the mapping requirements.

For example, a record is recognized by the system as:

<TAG>[Field_1][Field_2]

Define the record tag on the Positional Record Properties dialog box, instead of defining fields with the purpose of explicitly containing the record tag.

6. In the Position box, type the starting column position of the tag in the data record.
7. Select the **Special** tab only if you want to specify floating or wildcard options. Otherwise, accept the defaults.
 - Floating - The record does not have a fixed position in the file.
 - Wildcard - Generally only used with build or break maps.
8. Select the **Looping** tab only if the record is a looping structure. Otherwise, accept the defaults and continue with step 10.
9. If this record is a looping structure, on the **Looping** tab, enter the minimum and maximum number of times the record can repeat. Type **1** to make the header record mandatory.

Note: If the Min Usage box contains **0**, the record is conditional. If the Min Usage box contains **1** or greater, the record is mandatory.

10. Click **OK** to create the record.

Creating Subsequent Records

After you create the first record, you can define subsequent records at the same level. You can also define groups, if your application includes looping structures.

About this task

You may need to define a loop if your application requires a group of related records (and/or subgroups) that repeat in sequence. See [Creating a Group](#) for more information.

Use this procedure to create additional application records.

Procedure

1. Select the map component that precedes the record you are creating and select **Edit > Insert > Record** or **Edit > Create Sub > Record**, depending on the level. The system displays the Positional Record Properties dialog box.
 2. On the **Name** tab, type the record name and a description of the record.
 3. Select the **Tag** tab.
 4. In the Tag box, type the record identification code <TAG>. The record tag for each record enables the system to recognize that record and then determine the mapping requirements.
For example, a record is recognized by the system as:
`<TAG>[Field_1][Field_2]`
Define the record tag on the Positional Record Properties dialog box, instead of defining fields with the purpose of explicitly containing the record tag.
 5. In the Position box, type the starting column position of the tag in the data record.
 6. Select the **Special** tab only if you want to specify floating or wildcard options. Otherwise, accept the defaults.
 - Floating - The record does not have a fixed position in the file.
 - Wildcard - Generally only used with build or break maps.
 7. Select the **Looping** tab only if the record is a looping structure. Otherwise, accept the defaults and continue with step 9.
 8. If this record is a looping structure, on the **Looping** tab, enter the minimum and maximum number of times the record can repeat. Type **1** to make the header record mandatory.
- Note:** If the Min Usage box contains **0**, the record is conditional. If the Min Usage box contains **1** or greater, the record is mandatory.
9. Click **OK** to create the record.

Temporary Records

You may need to use temporary records and fields when you cannot use a simple link or if you need to extract only specific occurrences of a record from your data file. A simple link allows you to join data from the Input and Output sides of the map in either a one-to-one relationship (map components that both do not repeat) or a many-to-many relationship (map components that repeat the same number of times).

When to use temporary records and fields

If link a single iteration of a map component on the Input side of a map to a map component on the Output side that repeats multiple times, it causes an infinite loop when the map is translated. And, if you link a map component on the Input side that repeats multiple times to a map component on the Output side that does not repeat, the data from the Input side never populates the Output side of the map. In either of these instances, you should use temporary records and fields.

These are other common reasons why you would want to incorporate temporary records and field into your map:

- The Input hierarchical level in a map does not match the Output hierarchical level.
- You only want to populate an Output field with data if a specified qualifier is used or if specific criteria is met.

You may need to use extended rules in addition to creating temporary records and fields. See the Extended Rules topics for more information.

Where to use temporary records and fields

You can add temporary records and fields at any hierarchical level in a map. However, when you use temporary records and fields, you must locate them immediately after the map component that contains the necessary data.

The following caveats apply to temporary records:

- You must use a record tag that you would never receive in your Input file, and the recommended default is \$\$\$.
- If you are creating a temporary record for an XML file, use the tag XXX, because \$\$\$ is not permitted as a tag in XML.
- The system does not run standard or extended rules on temporary records.

Creating Temporary Records--Example

To map the shipping information from the Input side of your map (from the N1 Group that repeats a maximum of 200 times) to the ShipTo record in your application file format (that does not repeat), you need to create a temporary storage record and fields on the EDI side of the map that do not repeat. Then, you use an extended rule to extract the shipping and billing information from the N1 group and move it to the appropriate temporary storage elements. Finally, you map the shipping and billing information directly from the temporary storage elements to your application fields.

About this task

First, you need to create a temporary storage record and fields on the EDI side of the map that do not repeat. The temporary record is located outside the N1 Group at the same hierarchical level as it, and has a maximum repeat number of 1 (it does not repeat). Then you will create the appropriate temporary fields. After that, create an ON_END extended rule for the N1 Group that assigns the necessary information to the temporary fields.

Use this procedure to create the ShipToDet temporary storage record.

Procedure

1. Highlight the N1 group. The two temporary storage segments are located after the N1 group, at the same level.
2. Select **Edit > Insert > Segment**. (You select Segment instead of Record because you are creating the temporary record on the EDI side of the map).
The EDI Segment Properties dialog box is displayed.
3. In the Name box, type ShipToDet.
4. In the Description box, type Ship To Details.
5. Select the **Tag** tab.
6. In the Tag box, type \$\$\$.

Note: The system does not read a segment with a value of \$\$\$ in the Tag box. Therefore, it does not flag this temporary storage segment as an error during compliance checking.

7. Click **OK** to create the ShipToDet temporary storage segment.

8. Create the appropriate temporary fields, for example:
 - SHIPTONAME
 - SHIPTOADDR1
 - SHIPTOADDR2
 - SHIPTOCITY
 - SHIPTOSTATE
 - SHIPTOPCODE
9. Right-click the **N1** group and select **Extended Rules** to display the Group Properties dialog box.
10. Select **On End**. This specifies that the rule is executed when the loop terminates. The system loads an occurrence of the N1 group (containing the billing or shipping information), and then executes this rule.
11. In the Editor list, type the following:

```

IF #0098 = "BT" THEN
BEGIN
  $850.#BILLTNAME = #0093;
  $850.#BILLTOADDR1 = #0166;
  $850.#BILLTOADDR2 = #0166:2;
  $850.#BILLTOCITY = #0019;
  $850.#BILLTOSTATE = #0156;
  $850.#BILLTOPCODE = #0116;
END
IF #0098 = "ST" THEN
BEGIN
  $850.#SHIPTONAME = #0093;
  $850.#SHIPTOADDR1 = #0166;
  $850.#SHIPTOADDR2 = #0166:2;
  $850.#SHIPTOCITY = #0019;
  $850.#SHIPTOSTATE = #0156;
  $850.#SHIPTOPCODE = #0116;
END

```

Note: If a segment/record or element/field occurs more than once in a map, it is identified by its name <ID>. The second and subsequent occurrences are identified by <ID>:n, where n is the number of occurrences in the map.

12. Click **Compile** to validate the syntax of the extended rule.
 Every rule in the map is compiled when you compile the translation object, after you complete the map. However, the system allows you to compile each rule individually, so that you can verify the accuracy of the rule after you create it.
 This compiles the rule interactively, and allows you to correct any errors that are generated. Any errors or warnings generated in the compilation process are displayed in the Errors list.
13. Click **OK** to add the extended rule to the N1 group.
14. Link the temporary fields on the Input side of the map with the corresponding Output fields.

Creating a Group

A group contains related records, groups, or both that repeat in sequence until either the data ends or the maximum number of times that the loop is allowed to repeat is exhausted. If you create a group that is subordinate to another group, this corresponds to a nested looping structure (a loop within a loop).

About this task

Use this procedure to create a group.

Procedure

1. Select the map component that precedes the record you are creating and select **Edit > Insert > Group** or **Edit > Create Sub > Group**, depending on the level.
The system displays the Group Properties dialog box.
2. On the Name tab, type the group name and a description of the loop.
Do not use spaces or dashes (-) in the group name. You can use the underscore (_) to separate words.
3. On the Looping tab, enter the minimum and maximum number of times the record can repeat. Type **1** to make the header record mandatory.

Note: If the Min Usage box contains **0**, the record is conditional. If the Min Usage box contains **1** or greater, the record is mandatory.

4. If this is a single iteration group, select **Promote records to parent** if you want to specify that the subordinate records and groups should be extracted and located in the parent group when the group is compiled.
5. To specify an extended rule for this group, select the **Loop Extended Rules** tab.
See How to Define Extended Rules for more information.
6. Click **OK** to create the group.

Creating Fields

Each record you create contains a group of logically-related application fields. These fields define the structure and content of the data to the system.

About this task

The easiest way to add application fields to a record is to use the Positional Field Editor. Generally, you create the fields for the first record in the application file, and then proceed with each sequential record.

Note: Do not define fields with the purpose of using the field to explicitly contain the record tag. This is because the system takes the tag that you define in the record into account when the automatic sequencing (Auto Position) feature is used. We recommend that you define the record tag on the Positional Record Properties dialog box.

See Creating Subsequent Records for more information about the Positional Record Properties dialog box.

Use this procedure to create the application fields for a record.

Procedure

1. Right-click the application record and select **Edit Fields**.
The system displays the Positional Field Editor dialog box.
2. If this is the first field in the record, click **New**. Otherwise, highlight the field that precedes the field you are creating and click **New**.
The system displays a highlight bar in the Fields section where the new field is positioned.
3. In the Name box, type the field name.

Notes:

- Each application field must have a unique name. It is useful to tag the end of the fields that occur in multiple records with a suffix that identifies the record that contains it.
 - Do not use spaces or dashes (-) in the field name. You can use the underscore (_) to separate words.
4. If you want to designate the field as mandatory, select **Mandatory**.
 5. In the Description box, type a description of the field.
 6. From the Data Type list, select the type of the field:
 - **String** - alpha element
 - **Number** - numeric or real element
 - **Date/Time** - date or time element
 7. From the Format list, select how the field is formatted.

Notes:

- The choices for this field depend on the type of field you selected from the Data Type list. If you choose Number or Date/Time in the Data Type box, you can select the data format from the Format list. If you selected String from the Data Type box, you should type a syntax token to denote that this field must be formatted as the specified syntax token dictates (the default syntax token is X).
- When you installed Sterling Gentran:Server, you assigned a default format to the string fields. This format serves as the basis for character validation. Most U.S. users use a default format that corresponds to ASCII characters (the X syntax token). Most users of Asian or European languages and encoded character sets should use the Free Format (0x01-0xFF).

See Using Syntax Tokens for more information.

8. If you want to indicate the exact position of the field in the record, type the starting position of the field in the Start Pos box.

Note: You want to specify field start positions if, for example, you are only using a few fields but you want them positioned exactly in the record. The alternative to specifying the start position of each field, is to add the fields sequentially in the record and then use the Auto Position function.

9. Enter the minimum and maximum number of characters for this field.
10. Click **New**.

The system adds the field and creates a new field with blank values ready for you to identify, which is positioned below the field.
11. Create the rest of the fields according to your record layout.
12. Click **Delete** to stop adding fields.
13. After adding the last field, if you want the system to automatically position the fields in the record, click **Auto Position**.
14. Repeat steps 2 through 13 to create as many fields as you need. When you are done, click **Close**.

For inbound maps, you may want to use a literal constant to move a value into an application field that has no corresponding EDI data (you cannot link the field directly to an EDI element).

See Mapping Literal Constants for more information about mapping a constant to a field.

About EDI Files

Sterling Gentran:Server generates an EDI file for you, based on the standard (agency), version, transaction set, and release you select. The system includes all the groups, segments, composites, and elements that are defined by the standards agency for the version of the document you select.

See *Creating a Map* for more information about generating an EDI file when you create a new map.

You can modify the system-generated EDI file by modifying the properties of the map components and using the promote, split, copy, cut, and paste functions.

However, if you want to use a specialized version of an EDI standard that is not available in the Sterling Gentran:Server standards database, it may be appropriate for you to either load an EDI file definition or define the EDI file yourself. See *About CII Files* for more information about defining a CII file.

Regardless of whether the system generates the EDI file or you load or define it, the specific EDI map components that you use depends on the type of map you are creating. This includes the standard, version, and transaction set (document) selected, and which groups, segments, composites, and elements your company requires. We recommend that you determine which map components you are using before generating or defining an EDI file.

Verifying EDI Delimiters

If you are using an EDI standard that contains composite elements or sub-elements, you must verify that Sterling Gentran:Server is specifying the correct delimiters. Delimiters are flags that you define to the system as separating specific EDI components.

About this task

Delimiters are necessary for all variable field-length standards, because the data is compressed (and the leading zeroes and trailing blanks are removed). Since the fields vary in length, the system needs a flag to determine where one element ends and another begins. For example, an element delimiter marks the beginning of a new element.

Note: Although verifying EDI delimiters in Sterling Gentran:Server is mandatory only if you are using a standard with composite elements or sub-elements, we recommend that you perform this task regardless of which standard you use.

Use this procedure to verify EDI delimiters.

Procedure

1. Right-click the EDI file icon and select **Properties**.
The system displays the File Properties dialog box.
2. Select the **Delimiters** tab to access delimiter options.
3. Verify that **Specify defaults** is selected.
4. Verify the required delimiters for the EDI standard you are using.

Note: If the delimiters differ from the defaults specified, type either the character or the hexadecimal value in the correct box.

5. If you are using an X12, EDIFACT, or TRADACOMS standard on the output side of the map:
 - To suppress leading zeros for the EDI elements that use an R-format numeric (including decimal values less than one), select **Suppress leading zero on Numeric R* format values (for example, 0.25 becomes .25)**.
 - To pad numeric values (regardless of format) for the EDI elements with leading zeros, select the **Pad with leading zero on Numeric values (for example, 25 becomes 000025)** option.

Note: When you select this option, Numeric values (regardless of format) are padded with leading zeros out to the maximum length of the field.
6. Click **OK** to exit the File Properties dialog box.

Modifying Group Properties

A group contains related segments and/or groups that repeat in sequence until either the group data ends or the maximum number of times that the loop is allowed to repeat is exhausted. If you create a group that is subordinate to another group, this corresponds to a nested looping structure (a loop within a loop).

About this task

Use this procedure to modify the properties of a group.

Procedure

1. Right-click the group you want to modify and select **Properties**.

The system displays the Group Properties dialog box (Name tab displayed by default).
2. If you want to modify the group name, in the first box on the Name tab, type the new name.
3. If you want to modify the group description, in the second box on the Name tab, type the new description.
4. On the Looping tab, enter the minimum and maximum number of times the record can repeat. Type **1** to make the header record mandatory.

Note: If the Min Usage box contains **0**, the record is conditional. If the Min Usage box contains **1** or greater, the record is mandatory.

5. If this is a single iteration group, select **Promote records to parent** if you want to specify that the subordinate records and groups should be extracted and located in the parent group when the group is compiled.
6. If you want to specify an extended rule for this group, select the **Loop Extended Rules** tab.

See *How to Define Extended Rules* for more information.
7. Click **OK** to save changes to the group.

Modifying Segment Properties

You can modify the properties of any segment, including the minimum and maximum number of times the segment can repeat, whether the segment is mandatory or conditional, and whether it is a loop start or loop end segment.

About this task

See About Loop Start and Loop End Segments for more information.

Use this procedure to modify the properties of a segment.

Procedure

1. Right-click the segment you want to modify and select **Properties**.
The system displays the EDI Segment Properties dialog box.
2. If you need to modify the name of the segment, type the segment name in the first box on the Name tab.

Note: If a segment occurs more than once in a map it is identified by its name <ID>. The second and subsequent occurrences are identified by <ID>:n, where n is the number of the occurrence in the map.

3. If you want to modify the description of the segment, type a new description in the second box on the Name tab.

This box is used to provide a brief explanation of the segment that allows you to differentiate it from similar segments.

4. Select the **Tag** tab only if you want to access tag information.
5. If you want to modify the segment tag, type the segment identification code <TAG> in the Tag box.

The segment tag for each segment enables the system to recognize that segment. For example, a record is recognized by the system as:

```
<TAG>[Delimiter]<DATA>[Delimiter]  
<DATA>.....<DATA>[Segment Terminator]
```

6. Select the **Key Field** tab only if you want to use a key field.
The key field function enables you to specify a second qualification in selecting a segment (the segment name is the first qualification). Data must be provided in the order designated through the use of key fields.
7. Select the **Special** tab only if you want to specify floating (the segment does not have a fixed position in the file) or wildcard (generally only used with build or break maps) options, or if you want to flag this segment as containing binary data. Otherwise, accept the defaults.
 - Floating - The record does not have a fixed position in the file.
 - Wildcard - Generally only used with build or break maps.

Notes:

If you select **Binary**, you must define an element of data type Bin Length and another element of data type Bin Data. The Bin Length element must precede the Bin Data element.

See Modifying Element Properties for more information about defining an element data type.

8. If this segment is a looping structure, on the Looping tab, enter the minimum and maximum number of times the record can repeat. Type 1 to make the header record mandatory.
9. To indicate where this segment is in the loop, select one of the following options:
 - **Loop Start** - beginning of the loop
 - **Loop End** - end of the loop

- **Normal** - in the loop but not the beginning or ending segment
10. Click **OK** to save changes to the segment.

About Loop Start and Loop End Segments

Certain EDI standards use Loop Start (LS) and Loop End (LE) segments. LS and LE segments differentiate between two or more loops of the same type. If the transaction contains LS and LE segments and depending on whether your map is inbound or outbound, you need to define the LS and LE segments for the loops you are using in the map in one of two different ways.

You need to communicate with your trading partner to determine which loop identification code your partner will send to you (for an inbound map and/or expects to receive from you (for an outbound map).

For an inbound map, you need to do the following:

- Define an LS segment inbound.
- Define an LE segment inbound.

For an outbound map, you need to do the following:

- Define an LS segment outbound.
- Define an LE segment outbound.

Defining an LS Segment Inbound

About this task

Use this procedure to define an LS segment for an inbound map.

Procedure

1. Right-click the LS segment and select **Properties**.
The system displays the EDI Segment Properties dialog box.
2. On the **Looping** tab, verify that **Loop Start** is selected.
3. On the **Key Field** tab from the Field list, select **Loop Identifier Code**.
This list contains all the elements that are contained in the segment and gives you the ability to define the Loop Start segment definition by specifying that the loop identifier code must have the value you specify in the Matching rules section.
4. Select **Use constant**.
The Matching rules section enables you to access all the literal constants and code lists currently defined for this map.
5. Click **Edit**.
The system displays the Map Constants dialog box.
6. Click **New**.
The system displays the Edit Constant dialog box.
7. In the ID box, type the literal constant identifier.
8. From the Type list, select **String**. This is the category of this literal constant.
9. In the Value box, type the value of the literal constant.
For an inbound map, this is the loop identifier code that you expect to receive from your trading partner.
10. Click **OK** to add the constant to the system.

11. Select the constant you created from the list. The system matches the selected constant against the loop identifier code.
12. Click **OK** to exit the EDI Segment Properties dialog box.

Defining an LE Segment Inbound

About this task

Use this procedure to define an LE segment for an inbound map.

Procedure

1. Right-click the LE segment and select **Properties**.
The system displays the EDI Segment Properties dialog box.
2. On the **Looping** tab, verify that **Loop End** is selected.
3. On the **Key Field** tab from the Field list, select **Loop Identifier Code**.
This list contains all the elements that are contained in the segment and gives you the ability to define the Loop End segment definition, by specifying that the loop identifier code must have the value you specify in the Matching rules section.
4. Select the constant you created for the LS segment from the list.
The system matches the selected constant against the loop identifier code. The Matching rules section enables you to access all the literal constants and code lists currently defined for this map.
5. Click **OK** to exit the EDI Segment Properties dialog box.

Defining an LS Segment Outbound

About this task

Use this procedure to define an LS segment for an outbound map.

Procedure

1. Double-click the Loop Identifier Code element in the LS segment.
The system displays the Element Properties dialog box.
2. Select the **Standard Rule** tab.
3. From the standard rule list, select **Use Constant**.
The system displays the constant options.
4. Click **Edit**.
The system displays the Map Constants dialog box.
5. Click **New**.
The system displays the Edit Constant dialog box.
6. In the ID box, type the literal constant identifier.
7. From the Type list, select **String**. This is the category of this literal constant.
8. In the Value box, type the value of the literal constant. For an outbound map, this is the loop identifier code that your partner is expecting to receive from you.
9. Click **OK** to add the constant to the system.
10. Click **Close** to exit the Map Constants dialog box.
11. From the constant list, select the constant that you created.
The system matches the selected constant against the loop identifier code.

12. Click **OK** to exit the Element Properties dialog box.

Defining an LE Segment Outbound

About this task

Use this procedure to define an LE segment for an outbound map.

Procedure

1. Double-click the Loop Identifier Code element in the LE segment.
The system displays the Element Properties dialog box.
2. Select the **Standard Rule** tab to access the standard rule options.
3. From the standard rule list, select **Use Constant**.
The system displays the constant options.
4. From the constant list, select the constant that you created.
The system matches the selected constant against the loop identifier code.
5. Click **OK** to exit the Element Properties dialog box.

Modifying Composite Properties

A composite is a data element that contains two or more component data elements or subelements. You can modify the composite name and description, and whether or not it is mandatory.

About this task

Note: Composite elements are only used for EDI maps.

Use this procedure to modify the properties of a composite.

Procedure

1. Right-click the composite you want to modify and select **Properties**.
The system displays the Composite Properties dialog box.
2. If you want to change the composite name, type the data element identification number in the first box on the **Name** tab.

Note: If a composite occurs more than once in a map or form, it is identified by its name <ID>. The second and subsequent occurrences are identified by <ID>:n, where n is the number of the occurrence in the map or form.

3. If you want to change the composite description, type the new description in the second box on the **Name** tab.
4. If you want to specify that the composite is mandatory, select the check box on the **Validation** tab.
5. Click **OK** to save the changes to the composite.

Modifying Element Properties

Each segment or composite contains a group of logically-related elements. These elements define the structure of the EDI data that your system needs to process the document.

About this task

Use this procedure to modify the properties of an element.

Procedure

1. Double-click the element you want to modify.
The system displays the Element Properties dialog box.
2. To change the name of the element, type the new name in the first box on the Name tab. The name is typically the element sequence number.

Note: If an element occurs more than once in a map it is identified by its name <ID>. The second and subsequent occurrences are identified by <ID>:n, where n is the number of the occurrence in the map.
3. To change the description of the element, type the new description in the second box on the Name tab.
The description is used to provide a brief explanation of the element that allows you to differentiate it from similar elements.
4. If you want to specify that the composite is mandatory, select the check box on the **Validation** tab.
5. To change the minimum length of the element, type the minimum length in the Minimum box.
If the data is less than the minimum length, a compliance error is generated when translation occurs.
6. To change the maximum length of the element, type the maximum length in the Maximum box.
7. From the Data Type list, select the type of the element:
 - **String** - alpha element
 - **Number** - numeric or real element
 - **Date/Time** - date or time element
 - **Bin Data** - binary data (only available if you select "Binary" on the Special tab of the EDI Segment Properties dialog box)
 - **Bin Length** - length of binary data (only available if you select "Binary" on the Special tab of the EDI Segment Properties dialog box)See Modifying Segment Properties for more information about selecting "Binary" on the Special tab.
8. From the Format list, select how the data element is formatted.

Notes:

- The choices for this field depend on the type of field you selected from the Data Type list. If you choose Number or Date/Time in the Data Type box, you can select the data format from the Format list. If you selected String from the Data Type box, you should type a syntax token to denote that this field must be formatted as the specified syntax token dictates.
- When you installed Sterling Gentran:Server, you assigned a default format to the string fields. This format serves as the basis for character validation. Most U.S. users use a default format that corresponds to ASCII characters (the X syntax token). Most users of Asian or European languages and encoded character sets should use the Free Format (0x01-0xFF).

See Using Syntax Tokens for more information.

9. Click **OK** to save the changes to the element.

Defining and Modifying Relational Conditions

You can use relational conditions to connect fields together for syntax or compliance reasons. For example, Field A is invalid unless Field B is present. Therefore, if you set up a condition that pairs Fields A and B, the system generates a compliance error if one of those fields is not present. You can also view the conditional relationships between elements, as provided by the standard.

About this task

Important: The system enables you to edit EDI conditional relationships, but if you do, you generate a compliance error. We recommend that you view EDI conditions only.

Use this procedure to define, modify, and view field and element relational conditions.

Procedure

1. Double-click the field or element you want to modify.
The system displays the Field (or Element) Properties dialog box.
2. Select the **Conditions** tab.
The system displays the relational condition options.

The screenshot shows the 'Element Properties' dialog box with the 'Conditions' tab selected. The dialog has five tabs: Name, Validation, Extended Rule, Standard Rule, and Conditions. Under 'Please choose the type of relationship to use:', 'Conditional' is selected in a dropdown. Under 'Please select the condition field:', '0235 PRODUCT/SERVICE ID QUALIFIER' is selected in a dropdown. Below this, there are two lists: 'Available fields:' and 'Fields used in relationship:'. The 'Available fields' list contains: 0350 ASSIGNED IDENTIFIC, 0330 QUANTITY ORDERED, 0355 UNIT OR BASIS FOR, 0212 UNIT PRICE, 0639 BASIS OF UNIT PRICE, 0235:2 PRODUCT/SERVICE, and 0234:2 PRODUCT/SERVICE. The 'Fields used in relationship' list contains: 0234 PRODUCT/SERVICE ID. Between the two lists are 'Add-->' and '<-- Remove' buttons. At the bottom of the dialog are 'OK', 'Cancel', 'Apply', and 'Help' buttons.

3. Select the field connection condition from the type of relationship list:
 - **Paired/Multiple** - if any of the specified fields/elements are present then all fields/elements must be present
 - **Required** - at least one of the specified fields/elements must be present
 - **Exclusion** - no more than one of the specified fields/elements may be present
 - **Conditional** - if the first Condition field/element is present, the rest of the fields/elements must also be present
 - **List Conditional** - if the first Condition field/element is present, at least one of the specified fields/elements must also be present
4. Select the first field from the condition field list.

Notes:

- This is the field/element on which the conditional relationship hinges if you chose Conditional or List Conditional from the type of relationship list.
 - The condition field list is only active if you chose either Conditional or List Conditional from the type of relationship list.
5. From the Available fields list, select the field(s)/element(s) and click **Add**.
The system moves the fields to the Fields used in relationship list, to include the fields as a part of the conditional relationship.
- Notes:**
- The Available fields list contains all the fields in the translation object that are valid to be used in a condition at this point.
 - The Fields used in relationship list contains the fields that you selected to be a part of the conditional relationship.
 - To remove the fields from the conditional relationship, select a field or fields from the Fields used in relationship list and click **Remove** to move the fields back to the Available fields list.
6. Click **OK** to add the conditional relationship to the field.

About CII Files

If a side of your map is CII, you must either define it to Sterling Gentran:Server or load a previously-saved file definition.

A CII message consists of one message header, one TFD area, and one message trailer.

Before you define the CII file, you should determine which groups and TFDs you will need to use. The groups contain related groups and TFDs. This means that you must create groups before you create the subordinate subgroups and TFDs.

Each map component is arranged sequentially in the order that it is most logical for the system to process. Therefore, each level of your application must be created sequentially.

Note: Also see the *IBM Sterling Gentran:Server for Microsoft Windows CII User Guide*.

Modifying CII File Properties

The CII File Properties dialog box enables you to modify setting for the CII or CII-Positional sides of a map. This dialog box enables you toggle the use of dividing mode (a method of formatting CII data into fixed-length 251-byte records), non-transparent mode (encodes characters in the CII data so that they do not cause problems with some communications file-transfer protocols). This dialog box also enables you to specify single- or double-byte character sets to use in data translation.

About this task

Use this procedure to modify CII File Properties.

Procedure

1. Right-click the CII file icon and select **Properties**.
The system displays the CII File Properties dialog box.
2. Select the **Mode** tab.

3. If you will be communicating with a transmission system that is unable to process variable-length records, select the dividing mode **On** option.

Note: A dividing mode is a method of formatting CII data into the plural number of 251-byte fixed records.

4. If you want to use non-transparent mode, select **Non-Transparent Mode**. Otherwise, the default is cleared, which means you are using transparent mode.

Note: Use non-transparent mode to encode characters in the CII data so they do not cause problems with some communications file-transfer protocol communications control characters.

5. Select the **Character Sets** tab.
6. If you want to use single- or double-byte character translation, select the appropriate option from the 8-bit character set and 16-bit character set lists.
7. If want to specify an extended rule for this group, click the **Loop Extended Rules** tab.

See *How to Define Extended Rules* for more information.

8. Click **OK**.

The system saves your modifications and exits the CII File Properties dialog box.

Creating a Group

A group contains related TFDs, groups, or both that repeat in sequence until either the data ends or the maximum number of times that the loop is allowed to repeat is exhausted. If you create a group that is subordinate to another group, this corresponds to a nested looping structure (a loop within a loop).

About this task

Use this procedure to create a group.

Procedure

1. Select the map component that precedes the group you are creating and select **Edit > Insert > Group** or **Edit > Create Sub > Group**, depending on the level. The system displays the Group Properties dialog box.
2. On the Name tab, type the group name and the loop description.
3. On the Looping tab, enter the minimum and maximum number of times the record can repeat. Type **1** to make the header record mandatory.

Note: If the Min Usage box contains **0**, the record is conditional. If the Min Usage box contains **1** or greater, the record is mandatory.

4. If this is a single iteration group, select **Promote records to parent** if you want to specify that the subordinate records and groups should be extracted and located in the parent group when the group is compiled.
5. If you want to specify an extended rule for this group, select the **Loop Extended Rules** tab. See *How to Define Extended Rules* for more information.
6. Click **OK** to create the group.

Creating a TFD

The first TFD in your CII file is subordinate to the icon on the CII of the map. This graphic representation shows how the TFDs (and groups) are contained within the CII file.

About this task

After you create the first TFD, you can define subsequent TFDs at the same level. You can also define groups, if your application includes looping structures. You may need to define a group if your application requires a group of related TFDs (and/or subgroups) that repeat in sequence. Please see *Creating a Group* for more information.

When you create a TFD, the default data type is String, so the system displays the Character Set tab. If you change the TFD data type to Number or Date/Type, the system removes the Character Set tab.

Procedure

Use this procedure to create a TFD.

Procedure

1. If this is the first TFD in your file, select the **CII File** icon (INPUT for an inbound map and OUTPUT for an outbound map) and select **Edit Create Sub > TFD**.
2. If you are creating a subsequent TFD, select the TFD that precedes the TFD you are creating and select **Edit > Insert > TFD**. The system displays the CII TFD Properties dialog box.
3. On the **Name** tab, type the TFD name and a description.
4. Select the **Tag** tab.
5. Select either the Hex or Decimal option, depending on how you want to enter the data tag. (The default is Hex.)
6. If you want to use an extended range of values for the tag, select **Extended Mode**.

Notes:

- If you select Extended Mode, you can enter up to hex 7FFF (decimal 524287) in the Tag box. In compressed mode, you can only use up to hex EF (decimal 239).
 - The default is cleared, which means that the data tag will be expressed in compressed mode.
 - In Extended Mode, the range that you can use for the TFD tag is restricted to decimal 61440 - 65535 and hex F000 - FFFF.
7. Type the identification tag. You must enter a tag in this box.
 8. Select the **Looping** tab.
 9. Accept the default of Normal, unless you want to indicate that the segment is the beginning of a loop (Loop Start), the end of a loop (Loop End), or that the loop repeats (Loop Repeat).

Note: If you select Loop Start, the system displays the Key Field tab.

10. Select the **Validation** tab.
11. If you want to designate the TFD as mandatory, select the first check box.
12. To change the minimum length of the TFD, type the minimum length in the Minimum box.

Note: If the data is less than the minimum length, a compliance error is generated when translation occurs.

13. To change the maximum length of the TFD, type the maximum length in the Maximum box.
14. From the Data Type list, select the type of the TFD:
 - **String** - alphabetic
 - **Number** - numeric or real
 - **Date/Time** - date or time
15. From the Format list, select how the TFD is formatted.

Notes:

- The choices for this field depend on the type of field you selected from the Data Type list. If you choose Number or Date/Time in the Data Type box, you can select the data format from the Format list. If you selected String from the Data Type box, you should type a syntax token to denote that this field must be formatted as the specified syntax token dictates (the default syntax token is "X").
- When you installed Sterling Gentran:Server, you assigned a default format to the string fields. This format serves as the basis for character validation. Most users in the U.S. use a default format that corresponds to ASCII characters (the X syntax token). Most users of Asian or European languages and encoded character sets should use the Free Format (0x01-0xFF).

See Using Syntax Tokens for more information.

16. If the TFD data-type is Number or String and you want the data read/written in binary format:
 - a. Select the raw bytes check box.
 - b. From the Width list (Number format only), select how many bytes should be used to represent the number.
 - c. Select either the little-endian or big-endian option to specify in which order the bytes should be read.
17. If the **Character Set** tab is available, it enables you to set the single-double-byte character translation for this string. The default character set is 8-bit. Select the appropriate option from the 8-bit Character Set and 16-bit Character Set lists.
18. Click **OK** to create the TFD.

About Data Formatting

When you define or modify a field, element, or TFD, you must specify the type and format. The options that are available for the format of the field, element, or TFD depend on which type you select (string, number, or Date/Time).

Data formatting includes the following:

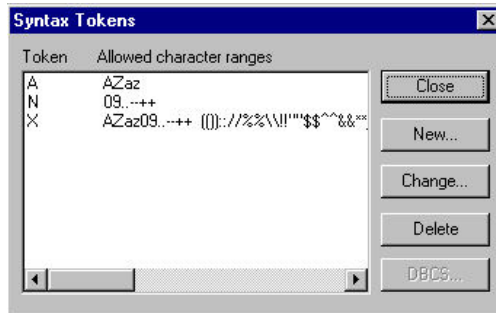
- Creating and editing syntax tokens
- Defining number formats (for number type fields)
- Defining date/time formats

String Type Fields and Syntax Tokens

A string-type field or element contains one or more printable characters. If you specify that a field or element is a string type, you must designate the format by specifying a syntax token.

Syntax tokens enable you to designate a token that defines ranges of characters and numbers that are allowed to be used for a string-type element or field. You can then use the syntax tokens in the Format field of the Field Properties dialog box. This enables you to define the type of characters to be used while compliance checking each element/field (for example, alphanumeric within a certain range or numeric within a certain range).

The following shows the Syntax Tokens dialog box.



Note: When you set up a token, it applies only to that map or form. You may need to set one up for each map or form that you create.

Sterling Gentran:Server uses the ANSI Character Set when determining the start-end range.

Western European Languages

For Western European languages, consult the ANSI character chart (1252 Windows Latin 1). This chart also displays ranges so you can enter appropriate ranges for the characters in your language. If no chart is available, use the following guidelines:

- To include all the accented characters in the major languages of Western Europe, add the following ranges:

Start	End
0xC0	0xD6
0xD8	0xF6
0xF8	0xFC

- Scandinavian users should also add the following in order to include \mathfrak{C} and \mathfrak{c} .

Start	End
0x8C	0x8C
0x9C	0x9C

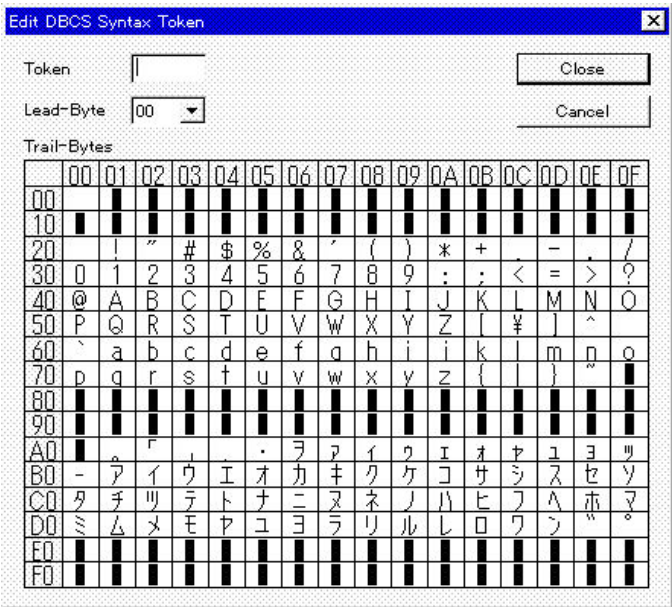
- To include the euro symbol, use the following.

Start	End
0x80	0x80

East Asian Languages

The DBCS syntax token function enables you to create a map that accepts double-byte characters. If you are running Sterling Gentran:Server on a Chinese, Japanese, or Korean version of Microsoft Windows, the DBCS button on the Syntax Tokens dialog box is available.

When you click the DBCS button, the system displays the Edit DCBS Syntax Token dialog box.



Creating and Editing Syntax Tokens for Western European Languages

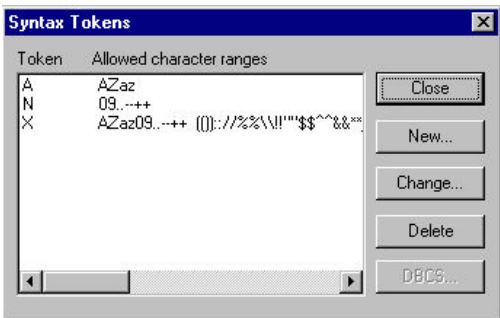
About this task

Use this procedure to create a syntax token for Western European Languages.

Procedure

1. Select **Edit > Syntax Tokens**.

The system displays the Syntax Tokens dialog box.



Notes:

- The Token column contains the value designated as the syntax token for each existing syntax token. The Token contains a range of characters that, when applied to an element, dictate the way that element must be formatted. If the element is not formatted as specified, the system generates a compliance error.
 - The Allowed character ranges column contains the ranges of characters that are permitted for each existing syntax token. Each range consists of a pair of characters that define the start and end characters.
 - The DBCS button is only available if you are executing a double-byte version of Microsoft Windows.
2. If you are creating a new syntax token, click **New**. Otherwise, select the token you want to edit and click **Edit**.
The system displays the Edit Syntax Token dialog box.
 3. Type the unique one-character alphanumeric value that the system recognizes as containing the allowed range of characters you designate.

Notes:

- The Token can only be one unique character, upper- or lowercase alphabetic or numeric.
 - The Character Ranges list contains the character range or ranges that you define for this token. You can define more than one character range for each token. For example, you can define the token "A" as allowing both uppercase and lowercase alphabetic characters.
4. If you want to create a new character range, click **New**.
The system displays the Edit Character Range dialog box.
 5. Type the characters that begin and end the allowed token range.
For example, if the character range you want to define is "B" through "D," type "B" in the Start character box and "D" in the End character box.
 - If you type a character, such as é, that is not accepted, you need to enter it in hex code. To enter hex characters, "0(zero)x" or "0X," followed by the hex code. For example, the hex equivalent of é is 0xE9.
 - The Start character and End character can only be one (1) character, upper- or lowercase alphabetic or numeric "1" - "9."
 6. Click **OK** to return to the Edit Syntax Tokens dialog box.
 7. To add additional character ranges to the syntax token, repeat steps 4 - 6 as many times as necessary.
 8. Click **OK** to save the syntax token and return to the Syntax Tokens dialog box.
 9. Click **Close** to exit the Syntax Tokens dialog box.

Creating and Editing Syntax Tokens for East Asian Languages

The DBCS syntax token function enables you to create a map or form that accepts double-byte characters. If you are running Sterling Gentran:Server on a Chinese, Japanese, or Korean version of Microsoft Windows, the DBCS button on the Syntax Tokens dialog box is available.

About this task

DBCS tokens are displayed only in the DBCS Syntax Tokens dialog box, not in the list in the Syntax Tokens dialog box.

Note: When you set up a DBCS token, it applies only to that map or form. You may need to set one up for each map or form that you create.

Use this procedure to create a DBCS syntax token.

Procedure

1. Select **Edit > Syntax Tokens**.
The system displays the Syntax Tokens dialog box.
2. Click **DCBS**.
The system displays the DCBS Syntax Token dialog box.
3. Click **New**.
The system displays the Edit DCBS Syntax Token dialog box.

Note: Double-byte characters are composed of a lead byte and a trail byte. In the above example, the character A is code point 0041.

4. Type the unique one-character alphanumeric value that the system recognizes as containing the allowed range of characters you designate.
5. If you do not want to include all characters in the Trail-Bytes section, do the following:
 - Use the Lead-Byte list to view other characters in the code page on your system.
 - To exclude individual characters or groups of characters from the token, select them (grey them out).
6. Click **Close** to save the syntax token and return to the Syntax Tokens dialog box.
7. Click **Close** to exit the Syntax Tokens dialog box.

Deleting Syntax Tokens

You can delete a syntax token from the system. You can also delete a specific character range from a syntax token.

About this task

Use this procedure to delete a syntax token.

Procedure

1. Select **Edit > Syntax Tokens**.
The system displays the Syntax Tokens dialog box.
2. Select the token that you want to delete and click **Delete**.
The selected entry is deleted without a warning message.

Deleting a Character Range

You can delete a syntax token from the system. You can also delete a specific character range from a syntax token.

About this task

Use this procedure to delete a character range from a syntax token.

Procedure

1. Select **Edit > Syntax Tokens**.
The system displays the Syntax Tokens dialog box.
2. Select a syntax token and click **Change**.
The system displays the Edit Syntax Tokens dialog box.
3. Select the character range that you want to delete and click **Delete**.

Using Syntax Tokens

The use of a syntax token enables you to define what characters should be used while compliance checking the field, element, or TFD (alphanumeric within a certain range, numeric within a certain range).

About this task

You can use syntax tokens in the Format box of the Field (or Element or CII Record) Properties dialog box. This enables you to designate a token that defines ranges of characters and/or numbers that are allowed to be used for a string-type field, element, or TFD.

Use this procedure to use syntax tokens.

Procedure

1. Double-click an existing field, element, or TFD, or create a new one.
The system displays the Field (or Element or CII Record) Properties dialog box.
2. Select the **Validation** tab.
3. From the data-type list, select **String**. This indicates that the field, element, or TFD contains characters.
4. From the data format list, select free format or a predefined syntax token to denote that this field, element, or TFD must be formatted as the specified syntax token dictates.

Notes:

- When you installed Sterling Gentran:Server, you assigned a default format to the string fields. This format serves as the basis for character validation. Most U.S. users use a default format that corresponds to ASCII characters (the X syntax token). Most users of Asian or European languages and encoded character sets should use the Free Format (0x01-0xFF).
 - Free Format indicates that any characters are acceptable in the field. The translator does not check the characters for compliance.
5. Click **OK** to exit the dialog box.

Using the Number Type

A number type field, element, or TFD contains either an implied decimal or real number that can be mathematically manipulated. If you specify that a field, element, or TFD is a number type, you must designate the format by specifying a format of either "N" (implied decimal) or "R" (real) and the number of decimal places.

About this task

Use this procedure to use the number type.

Procedure

1. Double-click an existing field, element, or TFD, or create a new one.
The system displays the Element (or Field or CII Record) Properties dialog box.
2. Select the **Validation** tab.
3. From the data-type list, select **Number**. This indicates that the field, element, or TFD is a numeric or real number that can be mathematically manipulated.
4. From the data format list, select the appropriate real or numeric option.
5. Click **OK** to exit the dialog box.

Number Formats

An N-formatted number has an implied decimal point (for example, 2.01 formatted as N2 is 201). An R-formatted number has an explicit decimal point and truncates trailing zeros (for example, 2.123 formatted as R2 is 2.12 and 3.10 formatted as R2 is 3.1). Whether you use the N or R format depends on the requirements of the document. Regardless of whether you use the N or R format, you must also indicate the number of decimal places in the field.

Table 6. Real Number Format Options

Field	Description
R0	Number formatted with an explicit decimal point and no decimal places.
R1	Number formatted with an explicit decimal point and up to one decimal place.
R2	Number formatted with an explicit decimal point and up to two decimal places.
R3	Number formatted with an explicit decimal point and up to three decimal places.
R4	Number formatted with an explicit decimal point and up to four decimal places.
R5	Number formatted with an explicit decimal point and up to five decimal places.
R6	Number formatted with an explicit decimal point and up to six decimal places.
R7	Number formatted with an explicit decimal point and up to seven decimal places.
R8	Number formatted with an explicit decimal point and up to eight decimal places.
R9	Number formatted with an explicit decimal point and up to nine decimal places.

Table 7. Numeric Number Format Options

Field	Description
N0	Number formatted with an implied decimal point and no decimal places.
N1	Number formatted with an implied decimal point and up to one decimal place.
N2	Number formatted with an implied decimal point and up to two decimal places.
N3	Number formatted with an implied decimal point and up to three decimal places.
N4	Number formatted with an implied decimal point and up to four decimal places.

Table 7. Numeric Number Format Options (continued)

Field	Description
N5	Number formatted with an implied decimal point and up to five decimal places.
N6	Number formatted with an implied decimal point and up to six decimal places.
N7	Number formatted with an implied decimal point and up to seven decimal places.
N8	Number formatted with an implied decimal point and up to eight decimal places.
N9	Number formatted with an implied decimal point and up to nine decimal places.

Note: For maps, if you select an implicit decimal (N format) for a field and the data in that field has less than the specified number of decimal places, the translator pads the data with zeroes to the left so that it still interprets the data within the specified format. For example, if you specify a format of N3 for a field, and the data in that field is 1, the translator interprets the data as .001.

Using the Date/Time Type

A Date/Time type field, element, or TFD contains a date or time. If you specify that a field, element, or TFD is a Date/Time type, you must specify exactly how the date or time must be formatted. Additional date and time formats can be created and added to the Map Editor.

About this task

Note: See “Date/Time Formats” on page 12 for details.

Use this procedure to use the Date/Time type.

Procedure

1. Double-click an existing field, element, or TFD, or create a new one.
The system displays the Element (or Field or CII Record) Properties dialog box.
2. Select the **Validation** tab.
3. From the data-type list, select **Date/Time**. This indicates that the field, element, or TFD is a date or time.
4. From the data format list, select the appropriate date or time option.
5. Click **OK** to exit the dialog box.

Completing a Map

Creating Simple Links

The Link function enables you to map a field, element, or TFD from the Input side of the map to a field, element, or TFD on the Output side of the map. The link between two map components (hereafter referred to as fields) is visually represented with a line connecting the two fields.

About this task

If you need to link two input fields to the same output field because of conditions established in your map, you must use an extended rule.

Note: You must have at least one direct link (using the Link function) to every record or segment on the Output side of the map, so the translator can create the record/segment.

Use this procedure to link two fields.

Procedure

1. Select **Functions > Link**.
2. Select a field on the Input side of the map (or TFD if the Input side is CII).
3. Select the field on the Output side of the map (or TFD if the Output side is CII) that the data from the field selected on the Input side of the map is mapped to.
A line connects the two fields to visually represent the link. If a line does not appear when the two fields are linked, check the Preferences dialog box.

Binary Segments

Certain EDI standards use binary segments. Sterling Gentran:Server enables you to create and configure map components to process binary data.

When you load an EDI standard that contains binary segments from the Standards DVD supplied with Sterling Gentran:Server, your system automatically creates the required elements. However, you sometimes may need to manually create binary segments.

For example, when you load the ANSI X12 275 transaction set into the Output side of a map, you will notice that it contains a binary data segment with two elements—one for the length of the binary data and one for binary data. The default data type for the length element is "Number" and the maximum length is set to "15." The default data type for the binary data element is "String" and the default length is "760."

To handle binary data received for an inbound EDI segment, you must create or configure regular segments and elements on the Input side of your map.

To generate an EDI segment containing binary data, you must create a binary data segment on the Output side of your map. Then, you must create two elements in the binary segment for these data types:

- Bin_Len, which is for the length (in characters) of the binary segment
- Bin_Data, which is for the binary data

Table 8. Input and output elements of your map for binary data

Input Data Type	Link Status	Output Data Type
Number	Not Linked	Bin_Len
String	Linked	Bin_Data

Setting up the Input Side

About this task

Use this procedure to create the components on the Input side of the map.

Procedure

1. On the Input side of the map, create or select a segment for the data.
2. Right-click the component and select **Properties**.
3. Set the component properties.

Note: You must complete the Name tab and Looping tab.

4. Add a new element or field to the component you just created.
5. Right-click the element or field and select **Properties**.
6. Set the component properties.

Note: On the Validation tab, you must specify **Number** as the data type.

7. Add a second new element or field to the segment or record.
8. Right-click the component and select **Properties**.
9. Set the component properties.

Note: On the Validation tab, you must specify **String** as the data type.

Setting up the Output Side

About this task

Use this procedure to create the components on the Output side of the map.

Procedure

1. On the Output side of the map, create or select a segment or record for the binary data.
2. Right-click the component and select **Properties**.
3. Set the component properties.

Note: On the Special tab, select **Binary**.

4. Add a new element or field to the binary component or select the existing element or field.
5. Right-click the component and select **Properties**.
6. Set the component properties.

Note: On the Validation tab, you must specify **Bin_Len** as the data type.

7. Add a second new element or field to the segment or select the existing element or field.
8. Right-click the component and select **Properties**.
9. Set the component properties.

Note: On the Validation tab, you must specify **Bin_data** as the data type.

10. Link the components on the input side of the map to the components on the output side of the map.

Important: You must link the string field on the input side of the map (the field created to contain the file name of the binary data file) to the bin_data field on the output side of the map.

What to do next

Complete the map and compile it to create the translation object.

Setting the Auto-register Option

The auto-register option allows you to automatically register a translation object when you compile it.

About this task

Use this procedure to set the auto-register option.

Procedure

1. Select **Options > Preferences**.
2. On the **Files** tab, select **Automatically register template after compilation**.
3. Enter (or navigate) the path name for the location of the maps.

Important: If you are on a client in a distributed environment, the path name for the map location must be a network path on the primary server. If not, you will receive an error when you compile.

4. Click **OK** to exit the dialog box.

Compiling a Map

The Compile function compiles the map and creates a translation object. The map that you create using Sterling Gentran:Server is a source map. When that source map is compiled, the result is a compiled translation object.

About this task

This translation object needs to be registered with the Sterling Gentran:Server system before you can use it. You use the Compile function after the map is completed and saved.

Use this procedure to compile a map and generate a translation object.

Procedure

1. Select **File > Compile**.
2. If necessary, navigate to where the compiled translation objects are stored. Enter the name of the translation object and click **Save**.

Note: The name of the translation object is 1-8 characters. Use the default .TPL file extension. You should name the translation object the same as you named the map. Using the same file name (with different file extensions) means that the relationship between the source map and the compiled translation object remains evident. For example, if the source map name is MWT_850.MAP, the compiled translation object name is MWT_850.TPL.

Important:

Be careful not to overlay the source map with the compiled translation object. Use the .TPL file extension to distinguish the translation object. Do not store the compiled translation object in the GENSRVNT\RegTransObj folder. This folder is reserved for storing a copy of each translation object you register with Sterling Gentran:Server.

The system compiles the map and generate a translation object. Any errors are displayed in the Compile Error dialog box.

3. Verify that no errors occurred and click **OK** to exit the dialog box.

Note: The date the translation object was compiled is automatically populated in the "Compiled on" box on the Translation Object Details dialog box.

4. Select **File > Save** to save the source map with the Compiled on date.

If you are using the auto-register option, the map will automatically be registered after it compiles and a confirmation dialog box will display.

What to do next

Register the translation object with the Sterling Gentran:Server system.

How to Compile Maps Using the Command Line

The Sterling Gentran:Server Application Integration subsystem (MAPPER.EXE) enables you to automatically compile a single map (or maps located in a specified directory) from the command line.

The compiled translation objects are written to same directory as the map source files.

The command line syntax (from the GENSRVNT\Bin subdirectory) is as follows:

```
mapper.exe -c [mapSourceFile]
```

The [mapSourceFile] is either a single map or a map filename that contains a wildcard character (*). Using the wildcard character causes MAPPER.EXE to compile all maps in the designated directory with that filename pattern. MAPPER.EXE does not recurse through subdirectories.

In the following example, all maps in the tutorial directory are compiled:

```
mapper.exe -c c:\GENSRVNT tutorial\*.map
```

In the following example, all maps with names beginning with "pet" in the tutorial directory are compiled:

```
mapper.exe -c c:\GENSRVNT tutorial\pet*.map
```

Printing a Mapping Report

The Print function enables you to print a mapping report for the current map.

About this task

Note: You can print a mapping report to a file by selecting Print to File from the File menu. Change the defaults on the Print Options dialog box and click **OK** to display the Print to File dialog box. Type the name of the file to which you want the report printed and click **OK**.

Use this procedure to print a mapping report.

Procedure

1. Select **File > Print**.

The system displays the Print Options dialog box.

2. Select the reports that you want by clicking the appropriate check boxes in the Report section, and click **OK**.
3. Set the appropriate options and click **OK** if you do not need to change Setup options.

The system prints the mapping report.

Chapter 3. Standard Rules

About Standard Rules

Sterling Gentran:Server provides standard rules that you can apply to fields and elements.

Standard rules give you access to functions that are necessary for mapping operations that are less involved than extended rules. Each of the standard rules are mutually exclusive (you can use only one on a particular field, element, or TFD).

Note: When an element contains a standard rule, a black asterisk appears to the right of the element icon.

Extended rules enable you to use a Sterling Gentran:Server proprietary programming language to perform virtually any mapping operation you require.

Standard Rule Tab - select Function

The Select function enables you to select entries from a location table, cross-reference table, partner table, or lookup table (all tables created in the Sterling Gentran:Server Partner Editor). You can then map the fields/elements in those tables to one or more fields/elements in the data.

The Select function uses the value of the current field, element, or TFD to perform the selection.

The Select function allows you to use information from the Sterling Gentran:Server Partner Editor in your maps or forms. You can map information from your partner's profile or Internal System Partner in the Partner Editor to a selected element or field in the map or form. The definition information and location tables that you define for the internal system partner can be used for all partners. The information that you can use includes:

- Partner or Internal System Partner Definition (Name, EDI Code, Application Code)
- Partner or Internal System Partner Location Table Fields
- Lookup Table Fields
- Cross-Reference Table Fields
- Interchange, Group, or Document Record

Table 9. Select standard rule parts and functions

Part	Function
table and key (or group)	Specifies the table and key the system uses to select data. This table lists the valid values.
name of sub-table	Contains the sub-table name, if appropriate. (See the table below.) The sub table box is only active if you select Partner Lookup, Partner cross-reference by my item, Partner xref by partner item, Division lookup, Division cross-reference by my item, or Division cross-reference by partner item from the table and key list.

Table 9. Select standard rule parts and functions (continued)

Part	Function
Raise compliance error	Specifies whether you want the system to generate an error if the Select does not find a valid table entry. Note: The default is cleared (do not generate an error if the Select is not successful).
Map from	Specifies the element/field/TFD from the specified table entry from which you want to map the contents. Note: Entries are displayed in the map from list only after you select a table and key.
Map to	Specifies the element/field/TFD to which you want to map the contents of the map from box. Notes: <ul style="list-style-type: none"> A total of eight fields can be mapped using one Select rule. Entries are displayed in the map to list only after you select a table and key, and these entries are activated only after you highlight an entry in the map from box.

Table 10. Types of sub-tables

Name	Description
Partner by EDI code	Indicates that the key is the partner's EDI Code and enables you to map from one of three fields for this partner (Name, EDI Code, Alternate Code) from the partner table.
Partner by alternate code	Indicates that the key is the partner's Application Code and enables you to map from one of three fields for this partner (Name, EDI Code, Alternate Code) from the partner table.
Partner by partner key	Indicates that the key is the partner's Profile ID enables you to map from one of three fields for this partner (Name, EDI Code, Alternate Code) from the partner table.
Partner location by name	Indicates that the name on the Locations dialog box (for the current partner) of the Partner Editor is the key and enables you to map from any field in that table.
Partner lookup	Indicates that the key is the partner lookup table name for this partner and enables you to map from any field in that table.
Partner cross-reference by my item	Indicates that the key is the your item (value) on the partner cross-reference table for this partner and enables you to map from any field in that table.
Partner cross-reference by partner item	Indicates that the key is the your partner's item (value) on the partner cross-reference table for this partner and enables you to map from any field in that table.
Division	Enables you to map from one of three fields (Name, EDI Code, Alternate Code) for the <Internal System User> (system partner) in that table.
Division location by name	Indicates that the key is the name on the Locations dialog box of the Partner Editor for the <Internal System User> and enables you to map from any field in that table.
Division lookup	Indicates that the key is the partner lookup table name for the <Internal System User> partner and enables you to map from any field in that table.
Division cross-reference by my item	Indicates that the key is the your item (value) on the partner cross-reference table for the <Internal System User> partner and enables you to map from any field in that table.

Table 10. Types of sub-tables (continued)

Name	Description
Division cross-reference by partner item	Indicates that the key is the your partner's item (value) on the partner cross-reference table for the <Internal System User> partner and enables you to map from any field in that table.
Document record	<p>Enables you to map from any of the fields within the current document record.</p> <p>Notes:</p> <ul style="list-style-type: none"> • This sub-table can only be accessed from the following map types: Transaction Break/Build, Import, Export, Print, Screen, Group Build, and Interchange Build. • The information that is accessed will be the information of the last document that was processed at the table level accessed by the rules. This must be taken into consideration when using rules that access these tables.
Generic envelope segment	<p>Enables you to map from any of the fields within the current envelope information, regardless of the EDI standard used.</p> <p>Note: This function is normally not used unless you are familiar with the generic envelope table structure for a given standard.</p>
Interchange	<p>Enables you to map from any of the fields within the current interchange record.</p> <p>Notes:</p> <ul style="list-style-type: none"> • This sub-table can only be accessed from the following map types: Interchange Break or Build, Group Break, Transaction Break, and Export. • The information that is accessed will be the information of the last interchange that was processed at the table level accessed by the rules. This must be taken into consideration when using rules that access these tables.
Group	<p>Enables you to map from any of the fields within the current group record.</p> <p>Notes:</p> <ul style="list-style-type: none"> • Can only be accessed from the following map types: Group Break or Build, Transaction Break, Export, and Interchange Build. • The information that is accessed will be the information of the last group that was processed at the table level accessed by the rules. This must be taken into consideration when using rules that access these tables.

Table access examples

This topic describes examples of accessing the Document table and the Group table.

Example 1

When accessing the Document table from the Interchange build map, the information retrieved will be from the last document processed through the Transaction Build map. If an import map updates a unique value to appfield6 for each document and the interchange build map tries to retrieve this information, and if all documents are sent in the same interchange, the only information retrieved in the Interchange Build map will be the information from appfield6 for the last document sent.

To retrieve the value in appfield6 for each document, the documents must be sent in their own interchange. This is set in the outbound partner relationship in the Interchange Definition under advanced settings (**Set Max number of documents per interchange** must be set to 1).

Example 2

When accessing the Group table from an Export map, the information retrieved will be from the last group processed through the Group Break map. If an inbound interchange file contained two groups: the first group contains 850 information and the second group contains 810 information, the rule in the 850 Export map will retrieve 810 information because the last group processed through the Group Break map was the 810 group.

Using Information from the Partner Definition

The Sterling Gentran:Server Application Integration subsystem allows you to use selected information from the Sterling Gentran:Server Partner Editor in your maps using a select standard rule. You can map information from your trading partner's profile in the Partner Editor to a selected element, field, or TFD in the map. The information that you can use in the map includes any field in a table (location, lookup, or cross-reference) or data from the partner table.

About this task

For this example, you need to pull information from the partner definition in order to populate the Customer Number field (CUSTNUMHDR) on the application side of the map. Your partner's customer number identifies which trading partner sent the purchase order. This customer number is already part of the partner definition for this partner, so you can map it from the partner definition to the customer number field.

Use this procedure to use the Select function to map the customer number from the partner definition.

Procedure

1. Double-click an existing element or field, or create a new element or field.
The system displays the Field (or Element or CII TFD) Properties dialog box.
 2. Select the **Standard Rule** tab to access standard rule options.
 3. From the standard rule list, choose **Select**.
 4. From the table and key list, select **Document record**. This indicates that the system updates this field with selected information from the document record.
 5. Select the compliance error check box to indicate that a compliance error should be generated if the select does not find a valid entry.
 6. From the map from list, choose **Partner Key**. This is the field that contains the customer number for this partner.
 7. From the map to list, choose **CUSTNUMHDR**. This is the field to which the information from the partner key of the document record is mapped.
- The following screen illustrates how the Standard Rules tab should look.

8. Click OK to add the standard rule.

Using Information from a Cross-reference Table

For this example, you need to map information from a cross-reference table to translate your partner's code for a purchased item into your code for the item. Your partner uses a customer product code (CUSTPROCEDURE) for each item that is meaningful to him. However, your company uses a UPC code (UPCCODE) for each item. After you and your partner determine what the equivalent customer product code is for each UPC code, you can create a cross-reference table in Partner Editor and use that information in your map.

About this task

A cross-reference table is used when you and your partner each reference an item by a different name (or number). For example, if your unique name for a "widget" is "wid" and your partner's unique name for the same "widget" is "1234." The system uses the cross-reference table to translate the two names for the "widget" item.

In this example, you have one code (UPC code) for each item ordered. Your partner refers to the same item by a different code (customer product code). You need to map the customer product code from the Product/Service ID element to the customer product code field (CUSTPROCEDURE) on the application side of the map. Also, you must create a cross-reference table to translate the customer product code to your UPC code.

Tip: Before you create the cross-reference table, you and your partner must determine what the equivalent customer product code is for each UPC code. Then you can create the cross-reference table in Partner Editor and use that information in your map.

Finally, you need to use a standard rule to map the translated value from the CUSTPROCEDURE field to the UPCCODE field.

Use this procedure to translate your partner's customer product code to your UPC Code for each item your partner ordered.

Procedure

1. Map the customer product code from the Product/Service ID element to your CUSTPROCEDURE field.
2. In the Partner Editor, create a cross-reference table named **PRODCODE**.

Notes:

- The cross-reference table you create only contains six entries. This is an example – an actual table would contain all of the items manufactured by your company that your partner purchases.
 - In the cross-reference table, the "My Value" box contains your UPC code for the item. The "Partner Value" box contains your partner's customer product code for the item. The "Description" box contains a description (and size, if applicable) of the item, and the Reference Data box contains additional quantity or color information.
3. Double-click the **CUSTPROCEDURE** field.
The system displays the Field Properties dialog box.
 4. Select the **Standard Rule** tab.
 5. From the standard rule list, choose **Select**.
 6. From the table and key, select **Partner cross-reference by partner item**. This indicates that you are using a cross-reference table and translating by your partner's item.
 7. In the sub table box, type **PRODCODE**. This is the name of the cross-reference table you created.
 8. Select the compliance error check box to signal the system that if this translation fails, you want the system to generate an error.
 9. From the map from list, choose **My Item**. This is the field from which the contents is mapped.
 10. From the map to list, choose **UPCCODE**. This is the field to which the information from the lookup table is mapped.

This diagram illustrates how the Standard Rule tab should look.

The screenshot shows the 'Field Properties' dialog box with the 'Standard Rule' tab selected. The dialog has several tabs: Name, Validation, Position, Extended Rule, Standard Rule, and Conditions. The 'Standard Rule' tab is active, showing the following fields and options:

- 'Please select the standard rule to use:' with a dropdown menu set to 'Select'.
- 'Please choose the table and key (or group) to use when selecting data:' with a dropdown menu set to 'Partner cross-reference by partner item'.
- 'Please enter the name of the sub-table to use (if appropriate):' with a text box containing 'PRODCODE'.
- A checked checkbox labeled 'Raise a compliance error if matching data is not found'.
- Below the checkbox, a message states: 'You can map the data that was selected. Click on data in the left list then click the fields in the right list that you would like it to be mapped to. Repeat to map more data.'
- Two lists for mapping data:
 - Left List (My Item):** Partner Item, Description, Text 1, Text 2, Text 3, Text 4.
 - Right List:** CUSTNUMDET Customer Number, PODATEDET Purchase Order Date, PONUMDET Purchase Order Number, CUSTPROCEDURE Customer Product Code, **UPCCODE UPC Code** (highlighted), UOM Unit of Measure, UNITPR Unit Price, QTYORD Quantity Ordered.
- Buttons at the bottom: OK, Cancel, Apply, Help.

Using Information from Location Tables

Each Partner Profile and the Internal System Partner Profile can have many associated location tables. The location table can be used to contain address-related information about the partner. You can store your partner's store addresses, warehouse addresses, or "invoice to" addresses.

About this task

To use information from a location table, you must have created that table already in the Partner Editor.

Note: For a screen entry translation object, the system displays a list that allows you to select the entry from the table. For a print translation object, the system prints the information on the report.

Use this procedure to map information from a location table.

Procedure

1. Double-click an existing element or field, or create a new element or field.
The system displays the Field (or Element or CII TFD) Properties dialog box.
2. Select the **Standard Rule** tab.
3. From the standard rule list, choose **Select**.
4. From the table list, select the key that is used to look up the Partner Location entry. You are allowed to map from any field on the Locations dialog box.
5. If you want the system to generate an error if the lookup fails, select the compliance error check box.
6. From the map from list, select the field from which the contents are mapped.
7. From the map to list, select the element, field, or TFD to which the information from the Partner Editor is mapped.
8. Click **OK** to add the standard rule.

Using Information from Lookup Tables

A lookup table is used to select information related to a value in inbound or outbound data. Each partner profile and the internal system partner profile can have many lookup tables associated with it.

About this task

To use information from a lookup table, you must have created that table already in the Partner Editor. See *How to Create a Table in the IBM Sterling Gentrans:Server for Microsoft Windows User Guide* for more information on creating Partner lookup tables.

Note: For a screen entry translation object, the system displays a list which allows you to select the entry from the table. For a print translation object, the system prints the information on the report.

Use this procedure to map information from a Lookup Table.

Procedure

1. Double-click an existing element or field, or create a new element or field.
The system displays the Field (or Element or CII TFD) Properties dialog box.

2. Select the **Standard Rule** tabs.
3. From the standard rule list, choose **Select**.
4. From the table list, select the key from which the system looks up the trading partner.
5. In the sub table box, type the name of the lookup table.
6. If you want the system to generate an error if the lookup fails, select the compliance error check box.
7. From the map from list, select the field from which the contents are mapped.
8. From the map to list, select the element, field, or TFD to which the information from the Partner Editor is mapped.
9. Click **OK** to add the standard rule.

Standard Rule Tab - update Function

The update function enables you to update a specific field in a document record, envelope segment, interchange, group, current partner, or document (for maps, if the format type is EDI), with the contents of the element, field, or TFD.

Important: This function updates the internal Sterling Gentran:Server database tables. We recommend that you use this function only if you want to update the internal database tables. Typically, you use this function only if you want to update the document name and reference in the document table.

You can select the table (group) to update and then select the column (field). The fields that are available depend on the table selected.

The following are the tables you can select to update with the contents of the current field/element/TFD:

Table	Description
Document record (use to update the document name and reference)	<p>This table can only be accessed from the following map types; Transaction Break/Build, Import, Export, Print, Screen, Group Build, and Interchange Build.</p> <p>The information that is accessed will be the information of the last document that was processed at the table level accessed by the rules. This must be taken into consideration when using rules that access these tables.</p>
Generic envelope segment (advanced use only)	<p>This sub-table can only be accessed from the following map types: Interchange Break or Build, Group Break, Transaction Break, and Export.</p> <p>The information that is accessed will be the information of the last interchange that was processed at the table level accessed by the rules. This must be taken into consideration when using rules that access these tables.</p>
Interchange	
Group	<p>Can only be accessed from the following map types: Group Break or Build, Transaction Break, Export, and Interchange Build.</p> <p>The information that is accessed will be the information of the last group that was processed at the table level accessed by the rules. This must be taken into consideration when using rules that access these tables.</p>

Table	Description
Partner (advanced use only)	
Document (advanced use only) - if the format type of this side of the map is EDI)	

Document Name and Reference Data

The Document Name and Reference Data are two recommended fields that you can specify in the Application Integration subsystem. For an invoice, this field is typically the invoice number. For a purchase order, this field is typically the purchase order number.

If you specify either the Document Name or Reference Data, the corresponding field in Sterling Gentran:Server document browsers (In Documents, ?In Documents, Out Documents, ?Out Documents, Workspace, In Drawer, Out Drawer, Send Queue, Interchanges) is populated. The Document Name and Reference Data makes the identification of a document created by the map easier in Sterling Gentran:Server.

Setting up the Document Name is strongly recommended if you want to be able to differentiate between documents in the document browsers in Sterling Gentran:Server. If you set up the Document Name in the Application Integration subsystem, the Name box of the document browsers that contain this document in Sterling Gentran:Server contains the data from that mapped field. If you set up the Reference Data in the Application Integration subsystem, the RefData box of the document browsers that contain this document in Sterling Gentran:Server contains the data from that mapped field.

You can select any element as the Document Name, but we strongly recommend that you choose an element that only occurs once in the map. The element that you select should have a data type of "string" (data type is specified on the Element Properties dialog box). The document name can be up to 255 characters in length.

If you need to use an element with a data type of Date/Time or Number for the Document Name or Reference Data, you can either:

- Change the data type to String on the Element Properties dialog box.
- Set up an extended rule to update the document record.

Note: If you change a Date/Time or Number element's data type to String, you lose the data type compliance checking for the original type.

Setting up the Document Name

About this task

Use this procedure to set up the Document Name.

Procedure

1. Select a non-recurring element from the header or trailer segment of the EDI side of the document.

The element that you select varies for each type of document. The selected element should be something that identifies this document meaningfully in Sterling Gentran:Server. For example:

- For an Invoice, this is typically the Invoice Number.
 - For a Purchase Order, this is typically the Purchase Order Number.
2. Right-click the appropriate element and select **Properties**.
 3. Select the **Standard Rule** tabs.
 4. From the standard rule list, select **Update**.
 5. From the table to update list, select **Document record**.
 6. From the column to update list, select **Document Name**.
 7. Click **OK** to set up the Document Name.

Setting up the Reference Data

About this task

Use this procedure to set up the Reference Data.

Procedure

1. Select an element from the first segment of the EDI side of the document.
The element that you select varies for each type of document. The selected element should be something that identifies this document meaningfully in Sterling Gentran:Server. For example:
 - For an Invoice this is typically the Purchase Order Number.
 - For a Purchase Order this is typically the Purchase Order Date. However, to use Purchase Order Date, you must use an extended rule on that element, instead of following the steps below.
2. Right-click the appropriate element and select **Properties**.
3. Select the **Standard Rule** tabs.
4. From the standard rule list, select **Update**.
5. From the table to update list, select **Document record**.
6. From the column to update list, select **Reference Data**.
7. Click **OK** to set up the Reference Data.

Standard Rule Tab - Use System Variable Function

Using the System Date and Time

The Use System Variable function enables you to set a variable that maps the current date and time to the selected element, field, or TFD. The selected component must have a data type of Date/Time.

About this task

Use this procedure to use the system date and time.

Procedure

1. Double-click an existing element, field, or TFD, or create a new one.
The system displays the Field (or Element or CII TFD) Properties dialog box.
2. Select the **Standard Rule** tab.
3. From the standard rule list, select **Use System Variable**.
4. Select **Current date and time** from the system variable list to map those variables to the element, field, or TFD.
5. Click **OK** to set up the system variable.

Standard Rule Tab - Use Constant Function

The Use Constant function enables you to move a literal constant value to the specified element, field, or TFD and indicate a qualifying relationship with another element, field, or TFD.

Constants are used in maps and forms to hold information that is needed later in the map/form, either to be moved to an output field or used in a conditional statement. Typically, constants are used to move a literal constant value to the specified element, field, or TFD on an inbound map. Outbound EDI qualifiers are typically generated from constants (to indicate a qualifying relationship with another element or field).

For an inbound map, constants are typically used to define the record types. For an outbound map, constants are typically used to define qualifying relationships.

A qualifying relationship establishes a relationship between an element and its qualifier. A qualifier contains a code that further defines the element. Qualifying relationships are typically defined in outbound maps.

Table 11. Use Constant standard rule parts and functions

Part	Function
constant	Lists the available constants. The selected constant is mapped to the current element, field, or TFD. Note: If the necessary constant is not present in the constant list, you need to create it by using the Map Constants dialog box, which you can access by clicking Edit .
Edit	Access the Map Constants dialog box, which allows you to create, edit, and delete literal constants.
qualifier	Lists the elements, fields, or TFDs that you can use to qualify the selected component to establish a qualifying relationship between the two components. Note: If the qualified element is not generated because of a lack of data, the constant is not moved to the current element, field, or TFD.

Table 12. Map Constants dialog box parts and functions

Part	Function
Constants list	Contains a list of all literal constants currently defined in the system.
Close	Exits the Map Constants dialog box.
New	Access the Edit Constant dialog box, which allow you to create a new constant.
Edit	Access the Edit Constant dialog box, which allows you to edit the selected constant.
Delete	Removes the selected constant from the system.

Table 13. Edit Constant dialog box parts and functions

Part	Function
ID	Contains the literal constant identifier. This is typically a description of the field or element in which the constant is used. If you need to refer to the constant in an extended rule, you should use the data from this field.
Type	Specifies the category of this literal constant. Valid selections are: <ul style="list-style-type: none"> • Integer - Select for numeric constants that are a positive or negative natural (non-fraction) number or 0. • Real - Select for numeric constants that are a positive or negative integer with an explicit decimal point. • String - Select for alphanumeric constants.
Value	Specifies the actual constant expression. This is the value of the literal constant.

Using a Constant in a Map

About this task

Use this procedure to use a constant in your map.

Procedure

1. Double-click an existing element, field, or TFD, or create a new one.
The system displays the Field (or Element or CII TFD) Properties dialog box (Name tab).
2. Select the **Standard Rule** tab.
3. From the standard rule list, select **Use Constant**.
The system displays the constant options.
4. To create or edit a constant to use for the current element, field, or TFD, click **Edit** to access the Map Constants dialog box.
5. From the constants list, select the constant that you want to map to the current element, field, or TFD.
6. To establish a qualifying relationship between the current map component and another component, from the qualifies list, select the element, field, or TFD that the system uses to determine whether to execute this standard rule (if the qualifying component contains data).
7. Click **OK** to save the standard rule and exit the Properties dialog box.

Defining a Qualifying Relationship

Literal constants are used by the system as a repository to store information that is used at a later point in the map. Typically, constants are used in an outbound map to generate a qualifier. A qualifier is an element that has a value expressed as a code that gives a specific meaning to the function of another element. A qualifying relationship is the interaction between an element and its qualifier. The function of the element changes depending on which code the qualifier contains.

About this task

In this example, you use a constant to define a qualifier for a Product/Service ID.

Use this procedure to define a qualifying relationship.

Procedure

1. Double-click the Product/Service ID Qualifier that you want to use to further define (qualify) the Product/Service ID.
The system displays the Element Properties dialog box (Name tab).
2. Select the **Standard Rule** tab.
3. From the standard rule list, select **Use Constant**.
4. Click **Edit**.
The system displays the Map Constants dialog box.
5. Click **New**.
The system displays the Edit Constant dialog box.
6. In the ID box, type the literal constant identifier. This is typically a description of the field or element in which the constant is used. If you need to refer to the constant in an extended rule, you should use the data from this box.
7. From the Type list, select **String**. The Product/Service ID Qualifier is formatted as a string data type. This indicates the category of the literal constant.
8. In the Value box, type **UI**. This indicates that the Product/Service ID field must contain the UPC Code. This is the value of the literal constant.
9. Click **OK** to add the constant to the system.
10. Click **Close** to exit the Map Constants dialog box.
11. From the qualifies list, select Product/Service ID. This is the element that the Product/Service ID Qualifier qualifies. This list contains only the other active fields or elements in the same record or segment as the qualifying field.
12. Click **OK** on the Element Properties dialog box and the qualifying relationship between the two elements is established.

Defining and Editing Literal Constants

About this task

Use this procedure to create or edit a literal constant so you can use it to store information.

Procedure

1. Select **Edit > Constants**.
The system displays the Map Constants dialog box.
2. To create a new constant, click **New**. To edit an existing constant, select the constant and click **Edit**.
The system displays the Edit Constant dialog box.
3. In the ID field, type the literal constant identifier.
4. From the Type list, select the category of this literal constant.
5. In the Value field, type the actual constant expression.
6. Click **OK** to add the constant to the system.
7. Click **Close** to exit the Map Constants dialog box.

Deleting Literal Constants

About this task

Use this procedure to delete a literal constant.

Procedure

1. Select **Edit > Constants**.
The system displays the Map Constants dialog box.
2. Select the constant you want to delete.
3. Click **Delete**.

Important: The system removes the constant without a warning message.

4. Click **Close** to exit the Map Constants dialog box.

Mapping Literal Constants

About this task

Use this procedure to map a constant in which you previously stored information using an extended rule.

Procedure

1. Double-click the element, field, or TFD in which you want to use the constant.
The system displays the Field, Element, or CII TFD) Properties dialog box.
2. Select the **Standard Rule** tab to access standard rule options.
3. From the standard rule list, select **Use Constant**.
4. From the constant list, select the constant that you want to use.
5. Click **OK**.

The data that was stored in the selected constant is loaded in the element, field, or TFD.

Generating Qualifiers

A qualifier is an typically an element that has a value expressed as a code that gives a specific meaning to the function of another element. A qualifying relationship is the interaction between an element and its qualifier. The function of the element changes depending on which code the qualifier contains.

About this task

Use this procedure to define qualifying relationships.

Procedure

1. Double-click the element you want to use to further define (qualify) another element.
The system displays the Element Properties dialog box.
2. Select the **Standard Rule** tab.
3. From the standard rule list, select **Use Constant**.
4. From the qualifies list, select the element that this element qualifies. This list contains only the other active elements in the same record or segment as the qualifying element.
5. Click **OK**.

Note: For a form, when you set a constant value for a qualifier field, you know that you never need to override the value in the field. In this instance, you would probably hide the field as well.

The system establishes the qualifying relationship between the two elements.

Standard Rule Tab - Loop Count Function

The Loop Count function enables you to count the number of times a loop is repeated, if the element, field, or TFD is part of a loop.

If the loop is a nested loop, you can track the current loop or the outer loop. For example, if the Y loop is nested within the X loop, and the Y loop has cycled through 15 iterations and the X loop has cycled through 3 iterations, you can choose to count either the 15 (Y loop) or the 3 (X loop).

Table 14. Loop Count standard rule parts and functions

Part	Function
group	Contains the loop that you want to count. Note: If the loop is a nested loop, you can track the current loop or the outer loop.

Using the Loop Count Function

About this task

Use this procedure to use the Loop Count function.

Procedure

1. Double-click an existing element, field, or TFD, or create a new one.
The system displays the Field (or Element or CII TFD) Properties dialog box.
2. Select the **Standard Rule** tab.
3. From the standard rule list, select **Loop Count**.
4. Select the loop that you want to count.
5. Click **OK** to add the standard rule.

Standard Rule Tab - Use Accumulator Function

The Use Accumulator function gives you access to a set of numeric variables that you can manipulate via numeric operations, and then transfer to and from elements or fields. This function enables you to add, change, or delete calculations for the element, field, or TFD, including hash totals (used to accumulate numeric field values, for example, quantity and price).

This function also enables you to map or load the accumulated total into a control total field or element. Accumulators are used generally for counting the occurrences of a specific element or generating increasing or sequential record or line item numbers.

Notes:

- Accumulators are global variables.
- Accumulators are set to zero before being used in calculations.
- The order in which accumulator operations are performed depends on the hierarchical order of the components.

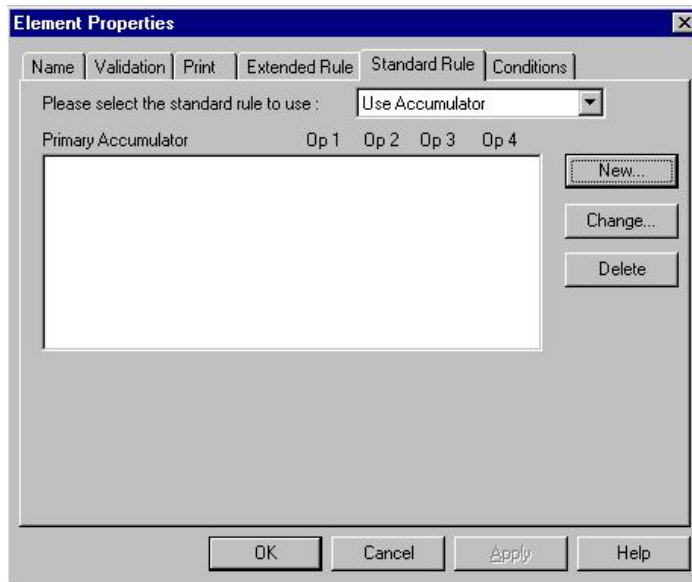


Table 15. Use Accumulator standard rule parts and functions

Part	Function
Primary accumulator	Lists existing accumulators and their associated operations that were created for this element, field, or TFD.
New	Accesses the Edit Accumulator Entry dialog box to define an accumulator for this element, field, or TFD.
Change	Highlight a calculation in the Primary Accumulator list and click this to access the Edit Accumulator Entry dialog box to edit the selected accumulator.
Delete	Highlight a calculation in the Primary Accumulator list and click this to delete the selected accumulator. Note: The selected calculation is deleted without warning.

Table 16. Edit Accumulator Entry dialog box parts and functions

Part	Function
Primary accumulator	Specifies the primary accumulator. Notes: <ul style="list-style-type: none"> Before any calculations are performed on an accumulator, its content is 0. When you use an accumulator, the system adds a new accumulator to the bottom of this list. There is only one set of accumulators for each map or form. This means that accumulator 0, whether it is used in the Primary Accumulator or Alternate Accum box is the same accumulator with the same contents. If you assign calculations to accumulator 0 at the beginning of the map/form and then use accumulator 0 again later, the content of that accumulator is the result of the earlier calculation. Any additional calculations you assign to that accumulator are performed on the contents resulting from an earlier calculation.
Name	Contains a descriptive alias that enables you to differentiate what the accumulators you create are used for.

Table 16. Edit Accumulator Entry dialog box parts and functions (continued)

Part	Function
First	Specifies the first operation that the system performs. Notes: <ul style="list-style-type: none"> The First box is active only after you select a Primary Accumulator. Before any calculations are performed on an accumulator, its content is 0. When you use an accumulator, the system adds a new accumulator to the bottom of this list.
Second	Specifies the second operation that the system performs, after the First operation is completed. The Second box is active only after you select a First operation that does not involve the Alternate Accum.
Third	Specifies the third operation that the system performs, after the Second operation is complete. The Third box is active only after you select a Second operation.
Fourth	Specifies the fourth operation that the system performs, after the Third operation is complete. The Fourth box is active only after you select a Third operation.
Alternate Accum	Specifies an alternate accumulator that participates in the accumulator operation. The Alternate Accum box is only active if you select an operation from the First field that involves an alternate operand.

Table 17. Accumulator operations and functions

Part	Function
Increment primary	Adds 1 to the contents of the Primary Accumulator (Primary = Primary + 1).
Decrement primary	Subtracts 1 from the contents of the Primary Accumulator (Primary = Primary - 1).
Sum in primary	Adds the numeric value (takes the positive or negative sign of the numbers into account) of the field to the contents of the Primary Accumulator (Primary = (+/-)Primary + (+/-)Field).
Hash sum in primary	Adds the absolute value (does not take the positive or negative sign of the numbers into account) of the field to the contents of the Primary Accumulator (Primary = Primary + Field).
Load primary	Loads the contents of the field into the Primary Accumulator (Primary = Field).
Use primary	Loads the contents of the Primary Accumulator into the field (Field = Primary).
Zero primary	Sets the value of the Primary Accumulator to zero (Primary = 0).
Multiply with primary	Multiplies the field with the contents of the Primary Accumulator, and stores the result in the Primary Accumulator (Primary = Primary * Field).
Divide by primary	Divides the field with the contents of the Primary Accumulator, and stores the result in the Primary Accumulator (Primary = Field / Primary).
Divide primary by field	Divides the contents of the Primary Accumulator with the field, and stores the result in the Primary Accumulator (Primary = Primary / Field).

Table 17. Accumulator operations and functions (continued)

Part	Function
Modulo with primary	Divides the contents of the Primary Accumulator with the contents of the field, and stores the remainder of that operation in the Primary Accumulator (Primary = Field % Primary).
Modulo with field	Divides the contents of the field with the contents of the Primary Accumulator, and stores the remainder of that operation in the Primary Accumulator (Primary = Primary % Field).
Negate primary	Makes the contents of the Primary Accumulator negative (Primary = Primary * -1). Note: The only way to subtract the Primary Accumulator from the field is to "Negate" the Primary Accumulator and then use the "Sum in primary" operation to add the negative Primary Accumulator to the field.
Move primary to alternate	Copies the contents of the Primary Accumulator to the Alternate Accum. This overwrites the current contents of the Alternate Accum field (Alternate = Primary).
Add primary to alternate	Adds the contents of the Primary Accumulator to the contents of the Alternate Accum and stores the result in the Primary Accumulator (Primary = Primary + Alternate).
Multiply primary by alternate	Multiplies the contents of the Primary Accumulator with the contents of the Alternate Accum and stores the result in the Primary Accumulator (Primary = Primary * Alternate).
Divide primary by alternate	Divides the contents of the Primary Accumulator with the contents of the Alternate Accum and stores the result in the Primary Accumulator (Primary = Primary / Alternate).
Modulo primary with alternate	Divides the contents of the Primary Accumulator with the contents of the Alternate Accum and stores the remainder of that operation in the Primary Accumulator (Primary = Primary % Alternate).

Counting Line Items

In this example, you want an incremental count of the number of line items, and you want to use that total value in the Number of Line Items Total field.

About this task

Use this procedure to count line items and generate a control total for an outbound ANSI X12 purchase order.

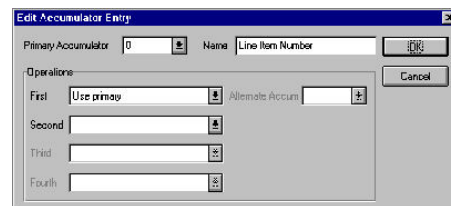
Procedure

1. Double-click the P0101 element (in the P01 segment in the P01 group). This is the element that you typically use to count the line items.
The system displays the Element Properties dialog box.
2. Select the **Standard Rule** tab.
3. From the standard rule list, select **Use Accumulator**.
4. Click **New**.
The system displays the Edit Accumulator Entry dialog box to create a new calculation for this element.
5. From the Primary Accumulator list, select **0**.

Note: There is only one set of accumulators for each map. This means that accumulator 0, whether it is used in the Primary Accumulator or Alternate Accum box is the same accumulator with the same contents. If you assign calculations to accumulator 0 at the beginning of the map and then use accumulator 0 again later in the map, the content of that accumulator is the result of the earlier calculation. Any additional calculations you assign to that accumulator are performed on the contents resulting from an earlier calculation.

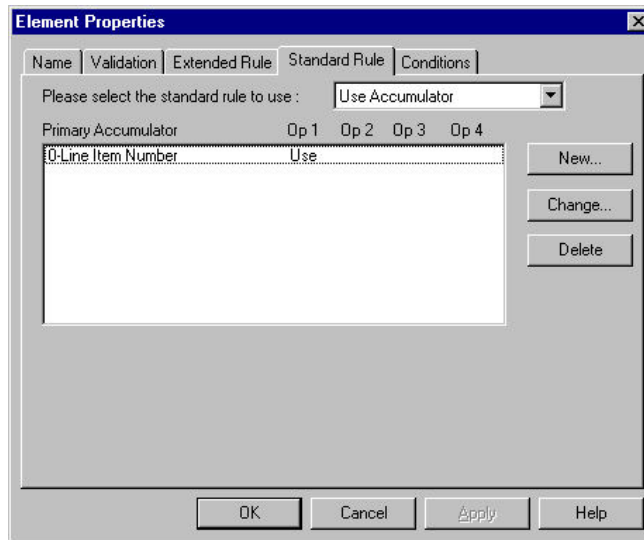
6. In the Name box, type **Line Item Number**. This is a descriptive alias that enables you to differentiate what the accumulators you create are used for.
7. From the First list, select **Increment Primary**. This is the first operation that the system performs. This specifies that the system will increment the PO101 element by one.
8. From the Second list, select **Use primary**. This is the second operation that the system performs, after the First operation is completed. This specifies that the system loads the current value of the accumulator into the P0101 (Assigned Identification) element.
9. Click **OK** to add the accumulator.
10. Click **OK** on the Element Properties dialog box to add the standard rule to the P0101 element.
11. Double-click the CTT01 element (in the CTT segment). This is the element that typically contains the total number of line items.
The system displays the Element Properties dialog box.
12. Select the **Standard Rule** tab.
13. From the standard rule list, select **Use Accumulator**.
14. Click **New**. The system displays the Edit Accumulator Entry dialog box to create a new calculation for this element.
15. From the Primary Accumulator list, select primary accumulator 0. This accumulator currently contains the total number of line items.
16. From the First list, select **Use primary**. This specifies that the system loads the current value of the accumulator into the CTT01 (Number of Line Items Total) element.

This diagram illustrates how the Edit Accumulator dialog box should look.



17. Click **OK** to add the accumulator.

This diagram illustrates how the Standard Rule tab should look.



18. Click **OK** on the Element Properties dialog box to add the standard rule to the CTT01 element.

Note: The CTT01 element now contains the total number of line items in the purchase order.

Calculating Hash Totals

In this example, you want to count the quantity ordered for each line item and load the total quantity in the CTT02 (Hash Total) element.

About this task

Use this procedure to count the quantity ordered and generate a hash total for an outbound ANSI X12 purchase order.

Procedure

1. Double-click the P0102 element (in the P01 segment in the P01 group). This is the element that you typically use to count the line items.
The system displays the Element Properties dialog box.
2. Select the **Standard Rule** tab.
3. From the standard rule list, select **Use Accumulator**.
4. Click **New**.
The system displays the Edit Accumulator Entry dialog box to create a new calculation for this element.
5. From the Primary Accumulator list, select **1**.
6. In the Name box, type **Total Quantity**.
7. From the First list, select **Hash Sum in Primary**. This is the first operation that the system performs. This specifies that the system will add the numeric value of the P0102 element to the contents of the Primary Accumulator.
8. Click **OK** to add the accumulator.
9. Click **OK** on the Element Properties dialog box to add the standard rule to the P0102 element.
10. Double-click the CTT02 element (in the CTT segment). This is the element that typically contains the total quantity of the purchase order.

The system displays the Element Properties dialog box.

11. Select the **Standard Rule** tab.
12. From the standard rule list, select **Use Accumulator**.
13. Click **New**. The system displays the Edit Accumulator Entry dialog box to create a new calculation for this element.
14. From the Primary Accumulator list, select primary accumulator 1. This accumulator currently contains the total quantity.
15. From the First list, select **Use primary**. This operation specifies that the system loads the current value of the accumulator into the CTT02 (Hash Total) element.
16. Click **OK** to add the accumulator.
17. Click **OK** on the Element Properties dialog box to add the standard rule to the CTT02 element. The CTT02 element now contains the total quantity of the purchase order.

Multiplying Quantity Invoiced by Unit Price

About this task

Use this procedure to multiply the quantity invoiced for each line item by the unit price to obtain the extended price for an outbound ANSI X12 invoice.

Procedure

1. Double-click the IT102 element (in the IT1 segment in the IT1 group). This is the element that you typically use to count the quantity invoiced.
The system displays the Element Properties dialog box.
2. Select the **Standard Rule** tab.
3. From the standard rule list, select **Use Accumulator**.
4. Click **New**.
The system displays the Edit Accumulator Entry dialog box to create a new calculation for this element.
5. From the Primary Accumulator list, select 2.
6. In the Name box, type **Extended Price**.
7. From the First list, select **Load primary**. This operation specifies that the system loads the contents of the element into the Primary Accumulator for each iteration of the IT1 group.
8. Click **OK** to add the accumulator.
9. Click **OK** on the Element Properties dialog box to add the standard rule to the IT102 element.
10. Double-click the IT104 element (in the IT1 segment in the IT1 group). This is the element that contains the unit price for each line item.
The system displays the Element Properties dialog box.
11. Select the **Standard Rule** tab.
12. From the standard rule list, select **Use Accumulator**.
13. Click **New**.
The system displays the Edit Accumulator Entry dialog box to create a new calculation for this element.
14. From the Primary Accumulator list, select primary accumulator 2.
15. From the First list, select **Multiply with primary**. This operation specifies that the system multiplies the value of the IT104 (Unit Price) element with the

contents of the primary accumulator, and store the result in the primary accumulator for each iteration of the IT1 group.

16. Click **OK** to add the accumulator.
17. Click **OK** on the Element Properties dialog box to add the standard rule to the IT104 element.

Tip: If there is an extended price element in your EDI file, you could load the total from the extended price calculation into that element. To do this, you need to use an accumulator on that extended price element that specifies **Use primary** for accumulator 2.

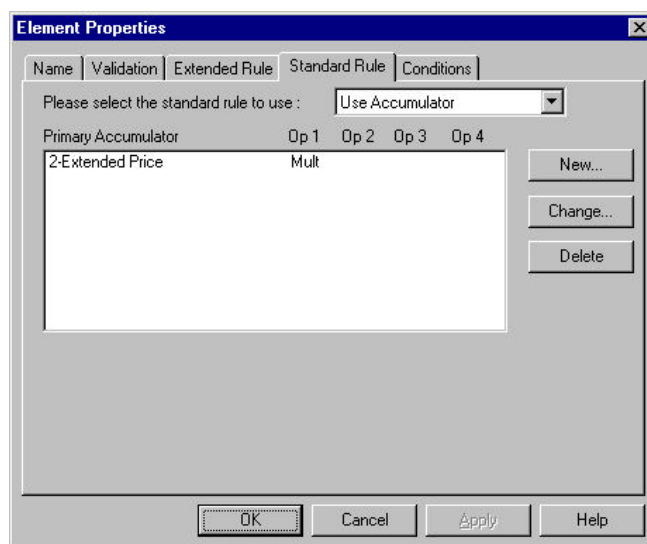
Generating a Running Total of Extended Price

About this task

Use this procedure to generate a running total of the extended price.

Procedure

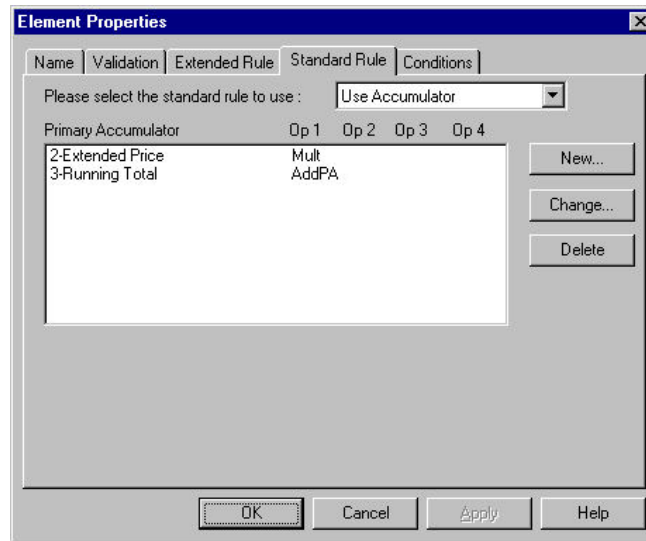
1. Double-click the IT104 element (in the IT1 segment in the IT1 group). This is the element that contains the unit price for each line item.
You already established one accumulator that the system displays in the list on the Standard Rule tab.



2. Click **New**.
The system displays the Edit Accumulator Entry dialog box to create a new calculation for this element.
3. From the Primary Accumulator list, select **3**.
4. In the Name box, type **Running Total**.
5. From the First list, select **Add primary to alternate**. This operation specifies that the system adds the contents of the primary accumulator to the contents of the alternate accumulator, and stores the result in the primary accumulator for each iteration of the IT1 group.
6. From the Alternate Accum list, select **2**. This operation specifies that the system add the value of accumulator 2 (which contains the extended price for a line item) to the value of accumulator 3. The system stores the sum in accumulator 3, which therefore contains a running total of the extended price with each iteration of the IT1 group.

- Click **OK** to add the accumulator.

This diagram illustrates how the Standard Rule tab should look.



- Click **OK** on the Element Properties dialog box to add the standard rule to the IT104 element.

Loading a Running Total of Extended Price

About this task

Use this procedure to load the running total of the extended price into the TDS01 (Total Invoice Amount) element.

Procedure

- Double-click the TDS01 element. This is the element that contains the total invoice amount for each line item.
The system displays the Element Properties dialog box.
- Select the **Standard Rule** tab.
- From the standard rule list, select **Use Accumulator**.
- Click **New**.
The system displays the Edit Accumulator Entry dialog box to create a new calculation for this element.
- From the Primary Accumulator list, select **3**.
- From the First list, select **Use primary**. This operation specifies that the system loads the contents of the primary accumulator into the TDS01 (Total Invoice Amount) element.
- Click **OK** to add the accumulator.
- Click **OK** on the Element Properties dialog box to add the standard rule to the TDS01 element.

Standard Rule Tab - Use Code Function

The Use Code function enables you to match an element, field, or TFD against a predefined code table and specify whether or not a compliance error is generated if the map component does not contain one of the values in the code table. This function also allows you to store a code's description in another element, field, or TFD.

You can also create a unique code table, use code values from a code table, and flag whether or not the system generates an error if a validation against the code table fails. You can import and export code lists and copy and paste code lists between maps.

Sterling Gentran:Server enables you to create code tables to be used with the current map or form. You can set up code tables to function like the partner cross-reference and lookup tables in Sterling Gentran:Server. However, code tables that are set up in the Application or Forms subsystem can be used only for the current map or form. Code tables that you create in Sterling Gentran:Server can be used globally for all maps/forms.

Code List Tables are used by EDI standards as repositories for lists of codes. Each EDI standard provides a code list for each element that can be further defined with a code. Sterling Gentran:Server allows you to load code lists from the standard. You can either load all the codes in the table, or you can select only one or more codes from the table. Once you load a code table, you can use a "Use Code" standard rule to either look up a value from a code table or validate the contents of an element, field, or TFD against the values in the code table.

A element, field, or TFD with a Use Code rule enables values to either be checked against or selected from the codes in a specified code table. Codes are typically used to further qualify another element. For example, if the XX element contains address information, you can further qualify that element by choosing the code "SU" from the 0222 table. In the 0222 table, the code "SU" is described as a "supplier's address." Therefore, by using this code with the XX element, you are indicating that the XX element is not just address information, but address information for the supplier.

Table 18. Standard rule tab (Use Code) parts and functions

Part	Function
Code list	Contains all the code tables. If the necessary code table is not listed, click Edit to load or create a code table.
Edit	Accesses the Edit Code List dialog box, which enables you to add, edit, delete, and load code list tables.
Raise compliance error	<p>Indicates that, for compliance reasons, the element, field, or TFD must contain one of the codes from the specified table (nothing else is valid for that field).</p> <p>For example, if a field is defined as containing only YES or NO, you can set up an exclusive code table that contains only YES and NO. Then if you receive a MAYBE in that field, the system flags it as an error.</p>

Table 18. Standard rule tab (Use Code) parts and functions (continued)

Part	Function
Code description	<p>Contains the element, field, or TFD where you want the description of the code that is used to appear when the selection is made.</p> <p>For example, if the code is SU, it is much more useful to view the description of the code (Supplier's Address). If you selected element XX from the store description list, the description for the code used is mapped to element XX.</p>

Table 19. Edit Code List dialog box parts and functions

Part	Function
Table list	Specifies the table identifier.
New	Accesses the Edit Code List dialog box, which allows you to create a new code list.
Change	Accesses the Edit Code List dialog box, which allows you to edit the selected code list.
Delete	Deletes the selected code list
Import	Accesses the Open dialog box, which allows you to import a code list.
Export	Accesses the Save As dialog box, which allows you to export the selected code list.
Copy	Copies the selected code list.
Paste	Pastes a previously-copied code list in a map.

Table 20. Edit Code List dialog box parts and functions

Part	Function
Table ID	Contains the name of the field or element for which this code list table is used.
Desc	Contains the description of the field or element for which this code list table is used.
Allowed Codes	Specifies the codes that are allowed for this table.
New	Accesses the Edit Code List Entry dialog box, which allows you to create a new code.
Change	Accesses the Edit Code List Entry dialog box, which allows you to edit the selected code.
Delete	Accesses the Edit Code List Entry dialog box, which allows you to delete the selected code.
Load	Accesses the Load Code List dialog box, which allows you to select, from a standard code table, specific codes that you want to load or select the entire list of codes.

Table 21. Edit Code List Entry dialog box parts and functions

Part	Function
Value	Specifies the actual value of the code.
Description	<p>Contains the code value description.</p> <p>Note: The description is used if you specify an element or field (in the Store Fields list on the Field Properties dialog box) to which you want the code description mapped.</p>

Defining and Modifying a Code List

About this task

Use this procedure steps to define or modify a code list table.

Procedure

1. Select **Edit > Code Lists**.
2. To create a new code list, click **New**. To edit a code list, select a code list and click **Change**.

The system displays the Edit Code List dialog box.

3. In the Table ID box, type the name of the field or element for which this code list table is used.
4. In the Description box, type the description of the field or element for which this code list table is used.
5. To create a new code, click **New**. To edit a code, select a code and click **Change**.

The system displays the Edit Code List Entry dialog box.

6. In the Value box, type the value of the code.
7. In the Description box, type a description of the code value.
8. Click **OK** to save the code list entry.
9. Repeat steps 5 through 8 to add more code list entries to the code list table.
10. Click **Close** to save and exit the Edit Code List dialog box.
11. Click **Close** to exit the Code Lists dialog box.

Deleting a Code List

About this task

Use this procedure steps to delete a code list table.

Procedure

1. Select **Edit > Code Lists**.
The system displays the Code Lists dialog box.
2. Select the code list you want to delete.
3. Click **Delete** to delete the code list table.

Important: The selected table is deleted without a warning message.

Deleting a Code List Entry

About this task

Use this procedure steps to delete an entry from a code list table.

Procedure

1. Select **Edit > Code Lists**.
The system displays the Code Lists dialog box.
2. Select the code list from which you want to delete an entry and click **Change**.
The system displays the Edit Code List Entry dialog box.

3. Select the entry and click **Delete**.

Important: The selected entry is deleted without warning.

4. Click **OK** to save the code list table.

Importing a Code List

The Code List Import function enables you to import code lists created for another map or form and share code lists with other users of Sterling Gentran:Server.

About this task

Use this procedure steps to import a code list table.

Procedure

1. Select **Edit > Code Lists**.

The system displays the Code Lists dialog box.

2. Click **Import**.

The system displays the Open dialog box.

3. Select the location of the code list file.

Note: The default location is Application Integration install folder (the default is GENSRVNT). The default file extension for code lists is .CDE.

4. Select the code list file from the list and click **Open**.

The system imports the code list and returns to the Code Lists dialog box. The imported code list is available for your use.

5. Click **OK** to exit the Code Lists dialog box.

Exporting a Code List

The Code List Export function enables you to export code lists to file. This enables you to define a code list for one map or form and use that code list in another map or form. This function also allows you to share code lists with other users of Sterling Gentran:Server.

About this task

Use this procedure steps to export a code list table.

Procedure

1. Select **Edit > Code Lists**.

The system displays the Code Lists dialog box.

2. Select a code list and click **Export**.

The system displays the Save As dialog box.

3. If you want, change the name of the export file.

Note: The filename defaults to the table ID with a .CDE file extension. The default location is Application Integration install folder (the default is GENSRVNT).

4. Click **Save**.

The system exports the code list and returns to the Code Lists dialog box.

5. Click **Close** to exit the Code Lists dialog box.

Loading a Code List Table from the Standard

About this task

Use this procedure steps to load a code list table from the standard.

Procedure

1. Double-click the element for which you need to use a code table.

The system displays the Element Properties dialog box.

Note: The standards provide code list tables only for elements that use them. For example, in the TD4 segment, the TD401 (Special Handling Code) and TD403 (Hazardous Material Class Code) have code tables provided by the standard.

2. On the **Standard Rule** tab, select **Use Code**.
3. Click **Edit**.

The system displays the Edit Code List dialog box.

4. Click **Load**.

The system prompts you to select a data source name (DSN) from which to access the EDI standard.

5. Select the appropriate DSN and click **OK**.

The system displays the Load Code List dialog box.

6. Select either specific codes that you want to load or select the entire list of codes.
 - To select specific codes only, from the Codes in Standard list, highlight each code that you want to be loaded. Click **Add** to move it to the Codes Selected list.
 - To load the entire code list, click **Add All** to select all the codes and move them to the Codes Selected list.

Note: Only add the codes that you and your trading partners are able to create or accept. Adding all codes in the code table (using the **Add All**) creates a much larger translation object than if you only use selected codes.

7. Click **OK** to load the code list.
8. Click **Close** to exit the Code Lists dialog box.

Copying and Pasting Code Lists

The Code List Copy and Paste function enables you to copy code lists to from one map or form to another.

About this task

Use this procedure steps to copy and paste a code list table.

Procedure

1. Select **Edit > Code Lists**.

The system displays the Code Lists dialog box.

2. Select a code list and click **Copy**.

The system copies the code list to the clipboard.

3. Click **Close** to exit the Code Lists dialog box.

4. Open the map or form in which you want to use the code list, if the map is not already open.
5. Select **Edit > Code Lists**.
The system displays the Code Lists dialog box.
6. Click **Paste**.
The system adds the copied code list to this map.
7. Click **Close** to exit the Code Lists dialog box.

Validating Data Against Code List Tables

About this task

Use this procedure steps to validate data against a code list table.

Procedure

1. Double-click the element for which you need to validate data against a code table.

The system displays the Element Properties dialog box.

Note: The standards provide code list tables only for elements that use them. For example, in the TD4 segment, the TD401 (Special Handling Code) and TD403 (Hazardous Material Class Code) have code tables provided by the standard.

2. Select the **Standard Rule** tab.
3. From the standard rule list, select **Use Code**.
4. From the code list, select the code list table that the data in this element is validated against.

Note: If this list is empty, you need to load a code table.

5. If you need to specify (for compliance reasons) that the element must contain one of the codes from the specified table (nothing else is valid for that field), select the compliance error check box.
6. Click **OK** to add this standard rule to the element.

Validating Data Example

For this example you need to map the customer product code from the Product/Service ID to the customer product code field (CUSTPROCEDURE) on the application side of the map. To ensure that the data that is mapped from the Product/Service ID element really is your partner's customer product code, you need to establish and use a code table with the Product Service ID Qualifier.

About this task

Use this procedure to validate the data in the Product/Service ID field against a code list table.

Procedure

1. Load the code list table from the standard for the Product/Service ID Qualifier element.

Note: The standards provide code list tables only for elements that use them. For example, in the TD4 segment, the TD401 (Special Handling Code) and TD403 (Hazardous Material Class Code) have code tables provided by the standard.

2. Define a code list table for the Product/Service ID Qualifier element that only contains the code value "BP" (Buyer's Part Number).
3. Double-click the element for which you need to validate data against a code table (Product/Service ID Qualifier).
The system displays the Element Properties dialog box.
4. Select the **Standard Rule** tab.
5. From the standard rule list, select **Use Code**.
6. From the code list, select the code list table that the data in this field is validated against (0235).
7. If you need to specify that, for compliance reasons, the element must contain one of the codes from the specified table (nothing else is valid for that field), select the compliance error check box.
8. Click **OK** to add this standard rule to the field.

Mapping Code Item Descriptions

About this task

Use this procedure steps to map a code item description.

Procedure

1. Double-click the element for which you need to map a code description.
The system displays the Element Properties dialog box.
2. Select the **Standard Rule** tab.
3. From the standard rule list, select **Use Code**.
4. From the code list, select the code list table that the data in this element is validated against.

Note: If this list is empty, you need to load a code table.

5. If you need to specify (for compliance reasons) that the element must contain one of the codes from the specified table (nothing else is valid for that field), select the compliance error check box.
6. From the store description list, the element to which you want the description of the code item (that is used) to be mapped to when the selection is made.
7. Click **OK** to add this standard rule to the element.

Chapter 4. The Import Process

About the Import Process

The import process is part of the outbound translation process. In the outbound translation process, the system translates your application file format to EDI standard formats, so you can send documents to your partners.

To translate outbound data, you need to create an import translation object and a system import translation object in the Sterling Gentran:Server Application Integration subsystem. The import translation object defines how to move data from your application file (flat file definition), which may contain multiple documents, to the EDI standard-formatted documents that your partners expect to receive from you.

You need to create a system import translation object to determine which trading relationship (established in Partner Editor) corresponds to each document in the application file, so the system knows which import translation object to use to process the document.

Importing Data

During the import process, Sterling Gentran:Server imports a specified application file into the Workspace. An import file can contain multiple flat file definitions (documents) that need to be sent to multiple trading partners.

You can import any type of flat file, including TDF (Transaction Data File). TDF is a flat file representation of the EDI standard document. The flat file characteristics can include multiple record types (more than one record definition) and records delimited by carriage returns/line feeds.

You cannot import EDI or field-delimited files. The field and record lengths for import files are defined in the Application Integration subsystem application file definition. For example, the maximum length of a field in an import file must not exceed the maximum length (Max) specified on the Field Properties dialog box for that field. And, the maximum length of a record in an import file must not exceed the record length (if specified) on the Positional File Format Properties dialog box for that record. The system default is variable length records. If you specify record delimiters on the Positional File Format Properties dialog box, the records in the import file must begin and end with those delimiters. If you do not specify record delimiters, the default delimiters are carriage return and line feed.

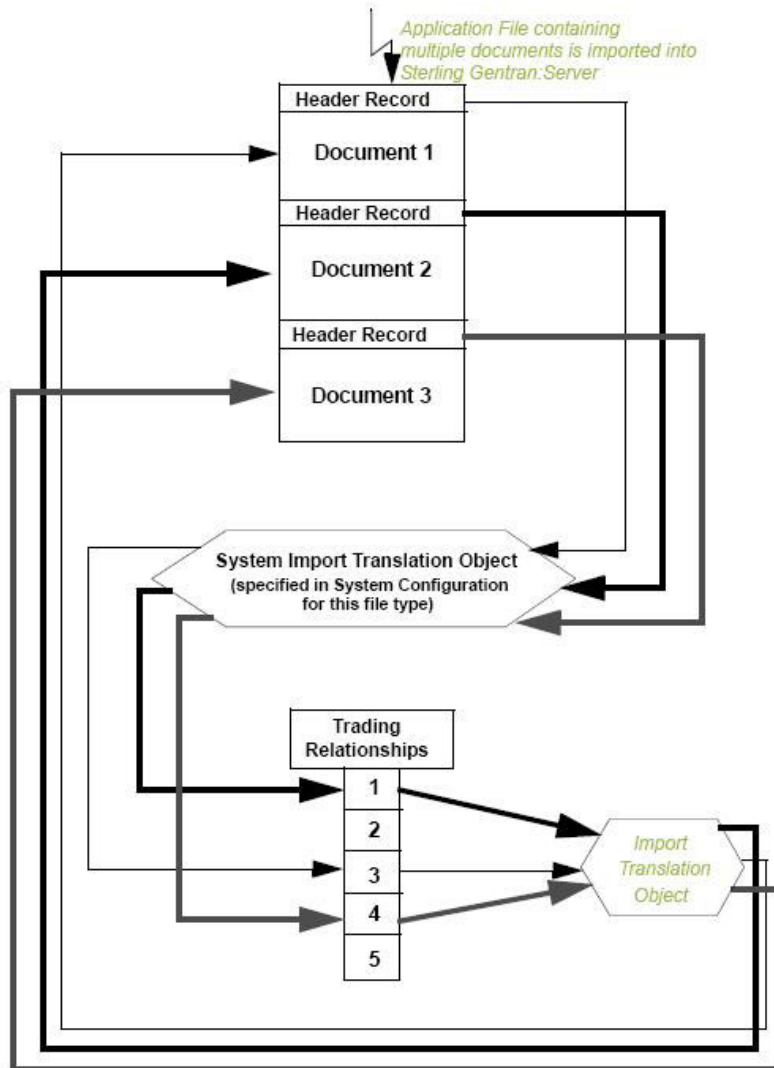
The system must determine which trading relationship (established in Partner Editor) corresponds to each document in the application file, so the system knows which import translation object to use to process the document. Each application file must have a corresponding system import map. For each document in the application file, the system import translation object identifies the partner and locates the trading relationship. The trading relationship for that partner must have an import translation object associated with it.

After the system import translation object determines which trading relationship corresponds with a document, it ascertains which import translation object is specified in that relationship. The translator uses the import translation object to

translate the document. If the document is compliant (valid), it is moved to the Workspace. If the document is not compliant (invalid), it is moved to the ?Out Documents. Then, if there is another document remaining in the application file, the translator repeats this process until all the documents are processed.

Import process diagram

This diagram illustrates the import process (translator not pictured).



How to Modify the System Configuration

When you import a file, the system checks the Imports tab of the System Configuration program (shipped with Sterling Gentran:Server) to verify that the file is matched with a registered system import translation object. Your system administrator may need to add the match to the System Configuration program.

Creating a System Import Map

A system import map is used by the system to find the partner relationship for a document (flat file definition), to determine which import map is used to translate the data. The system import map builds the key that the translator uses to find the partner relationship. The sole function of the system import map is to identify the appropriate partner relationship; the system import map does not map any data.

About this task

There are two ways to build the key in a system import map. The method that we recommend requires six EDI-specific fields in the header record: partner key, standard, version, transaction set, release (for TRADACOMS only), and test/production status. The combination of these six fields defines a unique key that identifies the appropriate partner relationship. We recommend using this method because it is very flexible. You typically use this method when you are defining your application from scratch and can easily add the EDI-specific fields that are not already present in the header record.

The second method is easier to build because it only requires three fields in the header record (partner key, application ID or application alias value, and test/production status). The combination of these three fields defines a unique key that identifies the appropriate partner relationship. However, because only three fields are combining to build the key, this method is not as flexible as the preferred method. You typically use this method when you are using a legacy (existing) application definition, and you do not want to add EDI-specific information.

Use this procedure to create a system import map.

Procedure

1. Select **File > New**.
The system displays the New Map Wizard.
2. From the type of map list, select **System Import**.
3. In the name box, type the name of the map.
4. Verify that your name is already entered in the name box and click **Next**.
The system displays the New Map Wizard - Input Format dialog box.
5. From the Create a new data format list, select **Positional** and click **Next**.
The system displays the New Map Wizard - Output Format dialog box.
6. From the Create a new data format list, select **Positional** and click **Next**.
The system actually ignores the Output side of the map and only processes the Input side.
7. Click **Finish** to create the new system import map (this may take a few seconds).
The system displays the new map in the Sterling Gentran:Server - Application Integration Window.
8. After creating the system import map, you need to define the header record for the system import map. You can use either a six-field key (this is the preferred method) or the alternate key method.

How to Define the Six-Field Key

To use the preferred method of building the system import key, you need to define at least five fields for the header record (partner key, standard, version, transaction, and test/production status). If it is a TRADACOMS map, you must also define a release field. These fields do not have to be in any order or sequence, but they must be part of the header record.

If you want to also define the other fields that the corresponding import map header record contains, you can do so at this time.

Once you define the header record and fields, you can set up the mapping operations that defines the key that the translator uses to find the partner relationship.

Defining the Partner Key

About this task

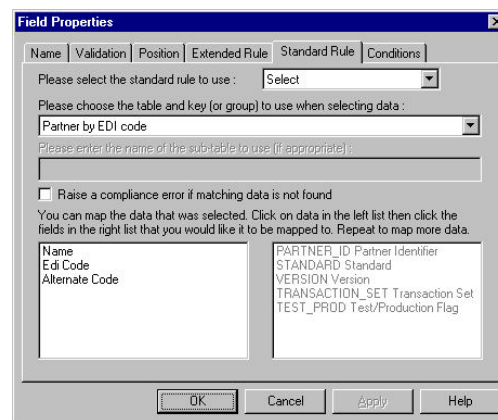
Use this procedure to define the partner key.

Procedure

1. Double-click the partner key field.
The system displays the Field Properties dialog box.
2. Select the **Standard Rule** tab.
3. From the standard rule list, choose **Select**.
4. From the table and key list, select one of the following:
 - Partner by EDI Code
 - Partner by Alternate code
 - Partner by Partner Key

Note: This indicates that the system updates this field with the indicated partner key.

The following is an example how the Standard Rule tab of the Field Properties dialog box should look.



5. Click **OK** to add the standard rule to the partner key field.

Defining the Standard Field

About this task

Use this procedure to define the EDI standard field.

Procedure

1. Double-click the standard field.
The system displays the Field Properties dialog box.
2. Select the **Standard Rule** tab.
3. From the standard rule list, choose **Update**.
4. From the table list, choose **Document record**.
This indicates that you are updating the document record in the Sterling Gentran:Server internal system buffer (because the translator is not mapping data at this point in the process).
5. From the column list, choose **Agency**.
This indicates that you are updating the agency (standard) field in the system buffer.
6. Click **OK** to add the standard rule to the standard field.

Defining the Version Field

About this task

Use this procedure to define the version.

Procedure

1. Double-click the version field.
The system displays the Field Properties dialog box.
2. Select the **Standard Rule** tab.
3. From the standard rule list, choose **Update**.
4. From the table list, choose **Document record**.
This indicates that you are updating the document record in the Sterling Gentran:Server internal system buffer (because the translator is not mapping data at this point in the process).
5. From the column list, choose **Version**.
This indicates that you are updating the version field in the system buffer.
6. Click **OK** to add the standard rule to the version field.

Defining the Transaction Field

About this task

Use this procedure to define the transaction set (document).

Procedure

1. Double-click the transaction set field.
The system displays the Field Properties dialog box.
2. Select the **Standard Rule** tab.
3. From the standard rule list, choose **Update**.
4. From the table list, choose **Document record**.

This indicates that you are updating the document record in the Sterling Gentran:Server internal system buffer (because the translator is not mapping data at this point in the process).

5. From the column list, choose **Transaction Set ID**.

This indicates that you are updating the transaction set field in the system buffer.

6. Click **OK** to add the standard rule to the transaction set field.

Defining the Release Field

This procedure applies only to TRADACOMS.

About this task

Use this procedure to define the release of the selected standard version (TRADACOMS only).

Procedure

1. Double-click the release field.

The system displays the Field Properties dialog box.

2. Select the **Standard Rule** tab.

3. From the standard rule list, choose **Update**.

4. From the table list, choose **Document record**.

This indicates that you are updating the document record in the Sterling Gentran:Server internal system buffer (because the translator is not mapping data at this point in the process).

5. From the column list, choose **Release**.

This indicates that you are updating the release field in the system buffer.

6. Click **OK** to add the standard rule to the release field.

Defining the Test/Production Field

If you do not define the test/production status field, the document record is set to production by default. In this situation, the system translation object can only locate the correct relationship if the test mode of the partner relationship is actually set to production.

About this task

Use this procedure to define the test or production status field.

Procedure

1. Double-click the test or production field.

The system displays the Field Properties dialog box.

2. Select the **Standard Rule** tab.

3. From the standard rule list, choose **Update**.

4. From the table list, choose **Document record**.

This indicates that you are updating the document record in the Sterling Gentran:Server internal system buffer (because the translator is not mapping data at this point in the process).

5. From the column list, choose **Test Mode**.

This indicates that you are updating the test field in the system buffer.

6. Click **OK** to add the standard rule to the test field.

How to Define the Alternate Key

To use the alternate key method of building the system import key (formerly referred to as the three-key method), you need to define (create new or update existing) three or four fields for the header record.

You must define a field for partner key, a field for either one or both application ID and application alias value (you only need to use one of these two fields), and a field for test/production status. If you define all four fields, you are using the alternate key method; if you use only one of the application ID or application alias value fields, you are actually using a three-field key. These fields do not have to be in any order or sequence, but they must be part of the header record.

Tip: You do not have to define new fields to use this key. You can add the necessary standard rules to three fields that already exist in your application file.

If you want to also define the other fields that the corresponding import map header record contains, you can do so at this time. Once you define the header record and fields, you can set up the mapping operations that defines the key that the translator uses to find the partner relationship.

Defining the Partner Key

About this task

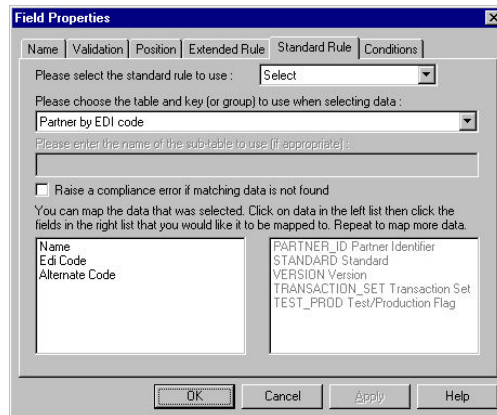
Use this procedure to define the partner key.

Procedure

1. Double-click the partner key field.
The system displays the Field Properties dialog box.
2. Select the **Standard Rule** tab.
3. From the standard rule list, choose **Select**.
4. From the table and key list, select one of the following:
 - Partner by EDI Code
 - Partner by Alternate code
 - Partner by Partner Key

Note: This indicates that the system updates this field with the indicated partner key.

This is an example how the Standard Rule tab of the Field Properties dialog box should look.



5. Click **OK** to add the standard rule to the partner key field.

Defining the Application ID Field

You must define either the application ID field or the Application Alias Value field (three-field key). You may define both of these fields if you wish to use the alternate key method.

About this task

Use this procedure to define the application ID field.

Procedure

1. Double-click the application ID field.
The system displays the Field Properties dialog box.
2. Select the **Standard Rule** tab.
3. From the standard rule list, choose **Update**.
4. From the table list, choose **Document record**.
This indicates that you are updating the document record in the Sterling Gentran:Server internal system buffer (because the translator is not mapping data at this point in the process).
5. From the column list, choose **Application Field 1**.
This indicates that you are updating an optional key field in the system buffer.
6. Click **OK** to add the standard rule to the application ID field.

Defining the Application Alias Value Field

You must define either the application ID field or the Application Alias Value field (three-field key). You may define both of these fields if you wish to use the alternate key method.

About this task

Use this procedure to define the application alias value field.

Procedure

1. Double-click the application alias value field.
The system displays the Field Properties dialog box.
2. Select the **Standard Rule** tab.

3. From the standard rule list, choose **Update**.
4. From the table list, choose **Document record**.
This indicates that you are updating the document record in the Sterling Gentran:Server internal system buffer (because the translator is not mapping data at this point in the process).
5. From the column list, choose **Application Field 2**.
This indicates that you are updating an optional key field in the system buffer.
6. Click **OK** to add the standard rule to the application alias value field.
7. Define the test/production field. See Defining the Test/Production Field.

Compiling the System Import Translation Object

After you create, define, and save the system import map, you need to compile the map and create a system import translation object.

About this task

Use this procedure to generate the system import translation object.

Procedure

1. Select **File > Compile**.
The system displays the Run-time Translation Object Name dialog box.
2. If necessary, navigate to the folder where compiled translation object is stored. Enter the name of the translation object and click **Save**.

Important: Be careful not to overlay the source map with the compiled translation object. Use the .TPL file extension to distinguish the translation object.
The system compiles the map and creates the system import translation object.
3. Registered the translation object with the Sterling Gentran:Server system before you use it.
4. If your system administrator has not already added this system import translation object to the System Configuration program, do so now.

Chapter 5. The Export Process

About the Export Process

To translate inbound data, you need to create an export map in Sterling Gentran:Server (because the system is exporting to your application file).

The export map defines how to move data from the EDI standard-formatted documents that your partners send you to your application file (flat file definition). This map also enables you to populate your application file fields with supplementary information from the interchange, group, and transaction set envelopes.

Inbound Process Before Exporting Data

During the export process, the Communicator receives interchanges from your trading partners via a network and passes the interchanges to the translator. The translator uses a system interchange break translation object (shipped with Sterling Gentran:Server) to unwrap the interchange envelopes and separate each group into temporary storage.

The translator uses a system group break translation object (shipped with Sterling Gentran:Server) to unwrap the group envelopes and separate each transaction set into temporary storage. The translator uses a system transaction break translation object (shipped with Sterling Gentran:Server) to unwrap the transaction envelopes and separate each document into the ?In Documents in Sterling Gentran:Server.

The translator attempts to locate the trading relationship for each document. If a trading relationship is located, the translator then attempts to identify the export, document turnaround, or print translation object associated with that relationship. If the translator does not locate the trading relationship or translation object, the document stays in the ?In Documents. If the translator does locate a trading relationship and translation object, it uses that translation object to compliance check the document. If the document is not compliant with the EDI standard, it stays in the ?In Documents. If the document is compliant with the EDI standard, the translator moves it to the In Documents. From the In Documents, you can print (if there is an associated print translation object), or export valid documents.

If you specify (in the trading relationship) that the system needs to generate a functional acknowledgement (ack) for a document, the translator uses the system acknowledgement translation object (shipped with Sterling Gentran:Server) to generate the acknowledgement. If the generated acknowledgement was compliant, the translator moves the acknowledgement to the Out Documents to be sent to the trading partner. If the generated acknowledgement was not compliant or if an error occurred with the acknowledgement translation object, the translator moves the acknowledgement to the ?Out Documents.

If you specified either automatic export or automatic turnaround in the trading relationship, the translator uses the specified export or document turnaround translation object to either export or generate the appropriate response document.

Setting up the Export Process

About this task

Use this procedure to set up the export process.

Procedure

1. Create an export map.
2. Register the export map with Sterling Gentran:Server.
3. Create the appropriate inbound trading relationship.

Using Supplementary Envelope Information

The Sterling Gentran:Server Application Integration subsystem enables you to use a Select standard rule to populate your application file fields (in an export map) with information from the EDI envelopes associated with the documents your trading partner sends you.

About this task

This table lists the correspondence between the selections available for the Select standard rule and the envelope information that you want to map to your application field.

If you select this value in the Map from list	Then the system maps this information
Field 1	Transaction Set ID
Field 2	Document Name
Field 3	Group ID
Field 5	Partner Name
Field 6	Interchange Control Number
Field 7	Group Control Number
Field 8	Document Control Number
Field 9	Received Version
Field 15	Received Agency
Field 16	Used Agency
Field 17	Used Version
Field 22	ISA Test Mode
Field 23	Document Test Mode

Use this procedure to use supplementary envelope information in your application file.

Procedure

1. Double-click an existing application field or create a new field.
The system displays the Field Properties dialog box.
2. Select the **Standard Rule** tab.
3. From the standard rule list, choose **Select**.

4. From the table and key list, select **Generic envelope segment**. This enables you to map from any of the fields within the current envelope information, regardless of the EDI standard used.
5. Select the compliance error check box. This indicates that a compliance error should be generated if the select does not find a valid entry.
6. From the map from list, choose the envelope information that you want to map to your application field.

Important: Verify that you select the correct map. Any selection that is not listed in the table is reserved and should not be selected under any circumstances.

7. From the map to list, choose the field to which you want to map the contents of the Map From field. The system displays each field from the output side of the map in this list.
8. If you want to map additional envelope information, repeated steps 6 and 7 as many times as necessary. A total of eight fields can be mapped using one Select rule.
9. Click **OK** to add the standard rule to the application field.

Chapter 6. Extended Rules

About Extended Rules

Extended rules enable you to use a Sterling Gentran:Server proprietary programming language to perform virtually any mapping operation you require.

You can use these rules to define more complex translations than are available through the standard rules. You can use extended rules to define operations that are not possible using standard rules.

You define extended rules with the Sterling Gentran:Server proprietary programming language. This is a full programming language that gives you access to the entire Sterling Gentran:Server internal storage area.

Any variables that are not already defined as part of the map (input or output) or form specification that you use in a rule must be declared before you use those variables.

An extended rule consists of two sections, a declarations section followed by a statements section.

The declarations section is only required if you use additional variables. This is where you declare the names and types of any variables you use either in this rule or in any other rule that is within the scope of this rule.

The statements section is where you define the actions that you want the rule to execute.

Declarations and Initialization

The variables that you define in the declarations section are used to store values. Variables consist of a name and a data type. Variable names can include alphanumeric characters and the colon (:) and underscore (_). The first character in a variables name may not be a numeric. All variable names are case-sensitive.

Notes:

- A declaration must be terminated with a semicolon (;). To improve readability, you typically include a blank line in between the declaration and statement sections.
- In maps, the Translator does not initialize extended rule variables. You must initialize all variables after declaring them. Variables that are not initialized can cause incorrect results or translation failures. For forms, initialization is not necessary.

Table 22. Data types that supported by extended rules

Data Type	Description	Example
Integer	a whole number with no decimal component	Declare i as an integer and initialize. <code>integer i; //Declaration</code> <code>i = 0; //Initialization of 'i'</code>

Table 22. Data types that supported by extended rules (continued)

Data Type	Description	Example
Real	a whole number that may have a decimal component	Declare r as a real number and initialize. <pre>real r; //Declaration r = 0; //Initialization of 'r'</pre>
String	contains one or more printable characters	Declare s as a 20-character string and initialize. <pre>string[20] s; //Declaration s = ""; //Initialization of 's'</pre>
Datetime	contains a date or time	Declare d as a date or time and initialize. <pre>datetime d; //Declaration d = date(0,0,0); //Initialization of 'd'</pre>
Array	defines a table of multiple occurrences of a single data type	Declare a as an array of 10 integers and initialize. <pre>integer a[10]; //Declaration integer i; //Declaration of 'i', which is //used to initialize array 'a' i = 1; //Initialization of 'i' //Initialization of the variable array 'a' while i < 11 do begin a[i] = 0; i = i + 1; end Declare p as an array of 50 10-character strings and initialize. string[10] p[50]; //Declaration integer i; //Declaration of 'i', which is //used to initialize array 'p' i = 1; //Initialization of 'i' //Initialization of the variable array 'p' while i < 51 do begin p[i] = ""; i = i + 1; end</pre>
Object (for maps only)	used for user exits; exposes the internal functions of an ActiveX Automation Server to Sterling Gentran:Server	Declare ob as an object and initialize. <pre>object ob; //Declaration ob = CreateObject("ADODB.Connections"); //Initialization of 'ob'</pre>

Statements

The actual work performed by an extended rule is defined in the statements section. A rule consists of a statement or a combination of statements (to perform more complex operations). A statement is a single operation that consists of a combination of expressions, keywords, commands, operators, and symbols.

An expression is a logical unit (for example, $A = B$ or $A + B$) that the system evaluates. The statements section consists of a sensible combination of keywords, operators, and symbols.

When Extended Rules are Processed

You can specify pre- and post-session rules on the Session Rules dialog box. Pre-session extended rules are processed before the translation object, and are in scope for every extended rule defined in the translation object. Post-session rules are executed after the translation object is processed.

You can attach extended rules to three different levels of map components:

- EDI file and application file groups, using the Loop Level Extended Rules dialog box.
- Groups, sub-groups, repeating records, and repeating segments, using the Loop Level Extended Rule dialog box.
- Fields and elements, using the Field Level Extended Rule dialog box.

The scope of an extended rule determines which variables are accessible from within a given extended rule. The scope varies depending on the current state of the map. This list defines the scope of an extended rule.

- Pre-session extended rules (defined on the Session Rules dialog box) are in scope for every rule in the translation object.
- On Begin extended rules (defined on the Loop Level Extended Rules dialog box) are in scope until the conclusion of its companion On End rule (also defined on the Loop Level Extended Rules dialog box).
- Field level extended rules are only in scope for the duration of the field or element.

An extended rule attached to the current map component depends on the type and state of the map component.

For example, if a current group to which the rule is attached is subordinate to another group, the parent group is automatically in scope for the duration of the entire child group, and the current hierarchical structure is also in scope for the duration of the child by using an addressing method that is explained in Symbols.

An extended rule that is attached to a field or element is only in scope for the duration of the field or element. Field level extended rules are always processed after standard rules.

A variable is considered to be "in scope" if it was declared in the current rule, in the On Begin rule of a group that contains the current map component, or in the Pre-Session rule.

The translator builds the data storage area for a map based on the structure of the Input side of the map (the source side of the map). Therefore, extended rules address the map based on the hierarchy of the Input side. When you use extended rules, you must be careful to always address the Input side of the map so the translator can locate the map component that the rule accesses. On the Output side of a map, extended rules only have access to the current record and the entire Input side of the map. However, from the Input side of a map, you have access to the entire file structure. As long as you address the source end of a link, you can write to any field on the Output side, even fields that have already been written.

Input rule processing

The translator processes the Input side of the map first and then the Output side.

Table 23. Sequence of how the translator processes rules on the Input side of a map

Stage	Description
1	Load the Input definition.
2	Read the Input file.
3	Determine if data is present for the first/next group and then run the group On_Begin rule, if present.
4	Load each field within the first/next record within the current group and then execute field level rules (for each field) in the following sequence: <ul style="list-style-type: none">• standard rules• extended rules
5	At the end of the group, execute the On_End rule, if present.
6	Repeat steps 2 - 5 for each group in the Input file.

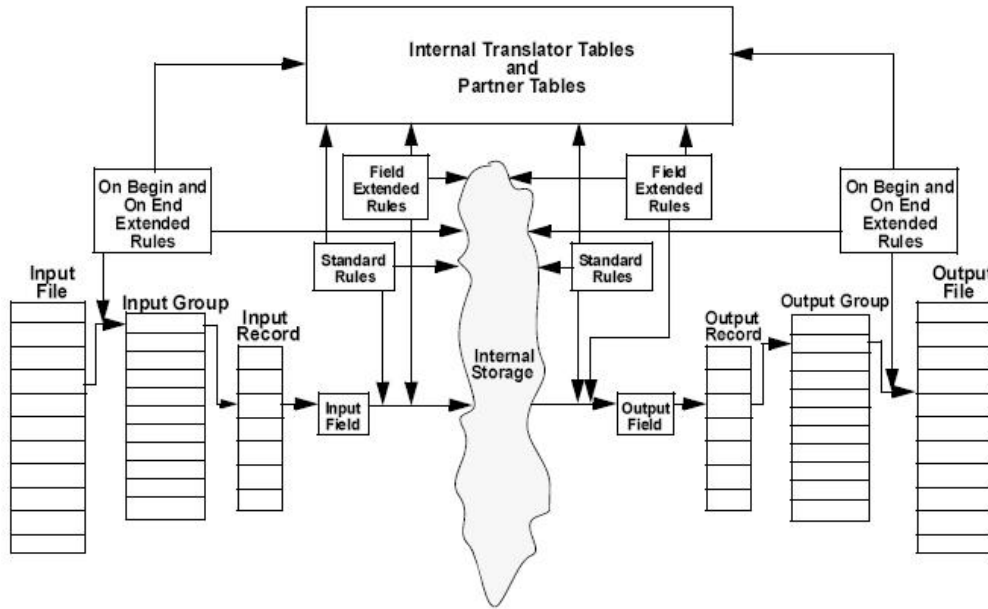
Output rule processing

Table 24. Sequence of how the translator processes rules on the Output side of a map

Stage	Description
1	Verify whether or not data exists for the first/next record.
2	If the record is the first record of a group, run On_Begin rule, if present.
3	For each field in the record, execute field level rules (for each field) in the following sequence: <ul style="list-style-type: none">• Run standard rules• Run extended rules
4	Format data according to specified field properties (on Field Properties dialog box).
5	Write the record to the Output file.
6	At the end of the group, execute On_End rule, if present.
7	Repeat steps 1 - 6 for each record in the Output file.
8	Create or update the document entry in the database.

Rule processing diagram

This diagram illustrates when loop level (On Begin and On End) extended rules and field level extended rules are processed in relation to the system process flow.



How to Define Extended Rules

The component that an extended rule accesses depends on what you want the scope of the rule to be. You also need to determine when you want the rule to be executed (for example, before or after the component is processed).

The process you need to follow to define an extended rule varies slightly, depending on whether you are defining a session rule or a rule for a map component. You can define extended rules to access three levels of components.

For maps, these components are:

- the entire session (input and output sides of the map)
- looping map components (groups, segments)
- fields

Defining a Session Rule

Pre-session rules are used to define variables that must have global scope (can be accessed from any other extended rule in the map or form). Pre-session extended rules are processed before the translation object, and are in scope for every extended rule defined in the translation object. Post-session rules are executed after the translation object is processed and thus have no permanent scope. You can define both a Pre-session and a Post-session rule for a given session.

About this task

Use this procedure to define a session rule.

Procedure

1. Select **Edit > Session Rules**.

The system displays the Session Level Extended Rules dialog box.

2. To define a pre-session rule, click **Pre-session**. To define a Post-session rule, click **Post-session**.
3. In the Editor list, type the extended rule.

Note: The Session Level Extended Rules dialog box contains line and character number (within a line) indicators. These indicators, which are displayed to the lower right of the Editor box (the first indicator references the line number and the second references the character number), enable you to easily debug compile errors.

4. To check the rule for errors, click **Compile** to compile the extended rule.
The Compile function gives you immediate feedback about the accuracy of your rule. The rule is compiled when you compile the entire translation object. Any warnings or errors are displayed in the Errors list. Double-click an error to instantly navigate to the line containing the error.
5. Correct any errors that the system flagged and click **Compile** again.

Note: Repeat this process until no errors are generated.

6. Click **OK** to add the extended rule.

Defining a Map Component Rule

Use this procedure to define an extended rule for a map component.

About this task

Procedure

1. Right-click the map component and select **Rules** or select **Extended Rule** if the map component is a field, element, or TFD.
 - If the map component is an EDI file, application file, group, sub-group, repeating record, or repeating segment, the system displays the Loop Level Extended Rules dialog box.
 - If the map component is a field or element, the system displays the Field Level Extended Rule dialog box.
2. If you want the extended rule to be executed before the system processes the map component, select **On Begin**. If you want the rule to be executed when the system concludes its processing of that map component, select **On End**.

Note: You can define both an On Begin and an On End rule for the a single map component.

3. In the Editor list, type the extended rule.

Note: The Extended Rules dialog boxes contain line and character number (within a line) indicators. These indicators, which are displayed to the lower right of the Editor box (the first indicator references the line number and the second references the character number), enable you to easily debug compile errors.

4. To check the rule for errors, click **Compile** to compile the extended rule.
The Compile function gives you immediate feedback about the accuracy of your rule. The rule is compiled when you compile the entire translation object. Any warnings or errors are displayed in the Errors list. Double-click an error to instantly navigate to the line containing the error.
5. Correct any errors that the system flagged and click **Compile** again.

Note: Repeat this process until there are no errors generated.

6. Click **OK** to add the extended rule.

Note: When an element contains an extended rule, a black asterisk appears to the right of the element icon.

Extended Rule Syntax

The statements section of an extended rule consists of a sensible combination of keywords, operators, and symbols.

The correct syntax for each of these component is explained in the following topics.

Note: You use spaces and operators to separate keywords and symbols. You cannot string two keywords sequentially together without an operator.

Keywords and Commands

A keyword is a fixed defined use of a word that indicates how the programming language should be interpreted. There are two types of keywords.

The first type of keyword controls the flow of execution of the defined rule. These keywords are used to evaluate conditions and perform looping operations. The second type of keyword is a command. Commands perform actions on variables and are responsible for the movement of data.

The following is a list of Sterling Gentran:Server execution control keywords.

- IF
- THEN
- ELSE
- BEGIN
- END
- WHILE
- DO
- CONTINUE
- BREAK

The following is a list of Sterling Gentran:Server commands.

- AUDITLOG
- GET
- SET
- STRDATE
- CONCAT
- LEN
- ATOI
- ATON
- CERROR
- EMPTY
- EXIST
- INDEX

- NTOA
- COUNT
- DELETE
- FSEEK
- FTELL
- READBLOCK
- UNREADBLOCK
- READBYTES
- LEFT
- MID
- RIGHT
- SELECT
- UPDATE
- INSERT
- WRITEBLOCK
- WRITEBYTES
- CREATEOBJECT
- DELETEOBJECT
- QUERYOBJECT
- GETIID
- WINEXEC
- DATE
- EXEC
- PARAM

Note: A statement must be terminated with a semicolon (;).

Operators

Operators define the simplest operation in an expression.

Table 25. Operators that can be used in extended rules

Part	Function
+	addition, concatenation
-	subtraction
*	multiplication
/	division
=	assignment, equality
>	greater-than
<	less-than
>=	greater-than or equal to
<=	less-than or equal to
!=	not equal to
!	logical not
&	logical and
	logical or

Table 25. Operators that can be used in extended rules (continued)

Part	Function
<<	date modification

Symbols

Operations are performed on symbols. The symbols that you can use in Sterling Gentran:Server extended rules are variables, constants, map or form components/internal storage, arrays, and accumulators.

You can address existing components and you have the ability to create additional instances of components, as long as the component is originally defined in internal storage.

For example, you can use a function to create extra line items when one line item field is already defined in internal storage.

You must address each type of symbol in the proper syntax.

String constant

To address a string constant, you must enclose the constant value in quotes:

```
#fieldname = "HDR";
```

where HDR is the constant value.

Addressing or creating a field in internal storage

To address a field or create a field in internal storage, within the scope of the current mapping action, the syntax is #FIELD_NAME.

```
#field_1 = 2;
```

where 2 is a numeric constant value.

Addressing or creating a field in a group

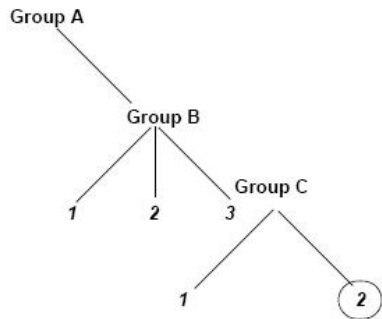
To address a field within a group or create a field within a group in internal storage within the scope of the current hierarchy, the syntax is \$GROUP.#FIELD_NAME \$N1.#0234

Addressing or creating a group in internal storage

To fully address a group in the entire internal storage area or create a group in internal storage, the syntax is \$LOOP[index1][index2][index3] where the index entries indicate the hierarchical structure of the loop and enable you to address specific instances of a group:

```
$Group_C[3][2].#Field_2
```

where you are specifying the second instance of Group_C within the third instance of Group_B:



Addressing an array

To address an array (of any type), you address each element of the array individually. For example, if `array_1` is an array (of integers) and is declared:

```
integer array_1[5]
```

With variables 0 through 4, each element of the array is addressed individually as follows:

```
array_1[0]
array_1[1]
array_1[2]
array_1[3]
array_1[4]
```

Accessing an accumulator

An accumulator can be accessed in the same manner as variables or internal storage. To address an accumulator, use the syntax `accum(n)`, where "n" is the number (not the name) of the accumulator:

```
accum(2) = 5;
```

Accessing repeating elements

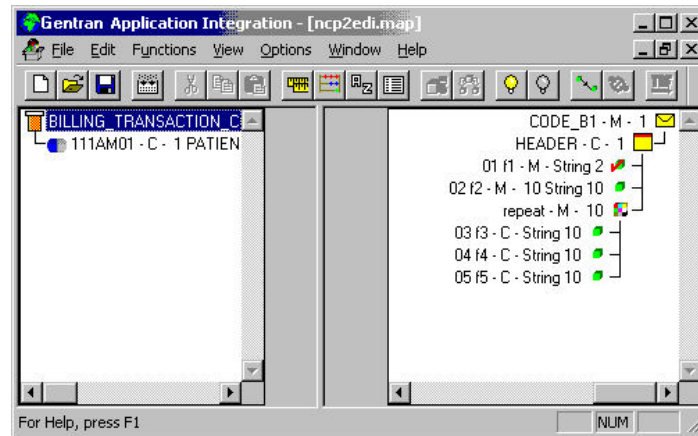
You can access a specific occurrence of a repeating element (for EDI data) and access a specific occurrence of a field within a repeating composite (for EDI data).

This is the syntax for accessing a specific occurrence of a repeating field and a field within a repeating composite.

```
field_name[index_variable] = string;
```

where `integer_variable` indicates the specific occurrence of a repeating field or field within a repeating composite.

The following screen, from the Application subsystem, is an example of accessing a specific occurrence of a repeating field and a field within a repeating composite.



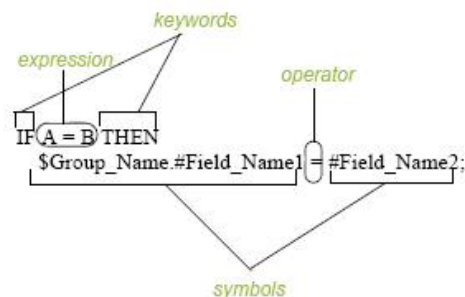
```
string [32]strMsg;

strMsg = "Test";
#f2[1] = strMsg;
//access single repeating field

#f3[1] = strMsg;
//access a field within a repeating composite
//The rule assigns a string value to 2 different fields, which are
//displayed in the diagram above, #f2 and #f3 -- #f2 is a single
//repeating field that can loop up to 10 times and #f3 is a field
//within a repeating composite where the composite can loop up to 10 times.
```

Example of a simple statement

This diagram illustrates an example of a simple statement.



Extended Rule Functions

About the Extended Rule Functions

Assignment

The assignment statement is the most powerful and most often used extended rule statement.

In the most simple form, the assignment statement is written as follows:

```
variable=expression
```

However, you can use this statement in more flexible and complex ways, defined as follows:

```
numeric_variable=numeric_expression  
numeric_field=numeric_expression  
string_variable=string_expression  
string_field=string_expression  
datetime_variable=datetime_expression  
datetime_field=datetime_expression
```

The following are some examples of assignment expressions:

```
a = 5;  
a = b + c;  
s = "hello";  
s = s + "world";
```

Definitions

A numeric expression can consist of numbers, numeric fields, numeric variables, and numeric functions combined with the standard arithmetic operators.

A string expression can consist of string constants, string fields, string variables, and string functions concatenated with the "+" operator.

A datetime expression can consist of a datetime constant, datetime field or datetime variable.

Datetime Expressions

Datetime expressions consist of a datetime variable and (optionally) datetime modifiers.

Datetime expressions can be written using datetime constants if you are using the standard syntax, as follows:

```
year/month/day  
hour:minute:second  
year/month/day/hour:minute:second
```

Datetime expressions can also be written with datetime fields, variables, or using date and time functions.

Syntax

Date functions are written as follows (month specified as 1-12):

```
datetime d;  
d = date(1995,4,6);  
d = date(1995,4,6,12,0);  
d = date("%y/%m/%d", "95/4/6");
```

The `d = date("%y/%m/%d", "95/4/6");` format enables you to convert any string format type into a datetime format type by indicating a format mask ("`%y/%m/%d`") along with the string ("`01/4/6`") you want to convert. You use this function if you are using non-standard syntax and need to specify the syntax you are using.

<< operator

You can use the << operator to modify your datetime variable by adding time increments (for example, days, weeks, years). For example:

```
datetime d;  
d=d <<weeks(2);  
//This adds 2 weeks to d.
```

Time syntax

Time functions are written as follows:

```
d = time(12,0);  
d = time(12,0,59);
```

You can use the << operator to modify your datetime variable by adding time increments (for example, seconds, minutes, years). For example:

```
datetime d;  
d=d <<seconds(1);  
//This adds 1 second to d.
```

Get and set syntax

The get and set functions enable you to access (get) or modify (set) individual components of a datetime type. These functions are used as follows:

```
integer a;  
datetime d;  
a = get days (d);  
a = get hours (d);  
set hours(d,a);  
set days (d,a);
```

Conditional Logic

Sterling Gentran:Server uses conditional logic to test conditions and then, depending on the results of the test, perform different operations. Conditions can be nested to any level. Do not end conditions with a semicolon (;) – this terminating syntax is necessary for statements only.

if...then...else

You can use the if/then keywords to execute one or more statements conditionally. The condition is typically a comparison, but it can be any expression that concludes with a numeric value. Sterling Gentran:Server interprets the value as either true or false. The system interprets a zero value as false and a nonzero value as true.

If you include more than one statement in the body of an if/then loop, you must surround the statements with the begin/end keywords. If you only use a single statement, you can omit the begin and end.

Sterling Gentran:Server evaluates the if/then condition, and if it is true, the system executes all the statements that follow the then keyword. If the condition is false, none of the statements following then are executed.

You can use the else keyword in conjunction with if/then to define several blocks of statements, one of which is executed. Sterling Gentran:Server tests the first if/then condition. If the condition is false, the system proceeds to test each sequential condition until it finds one that is true. The system executes the corresponding block of statements for the true condition. If none of the if/then conditions are true, the system executes the statements following the else keyword.

Syntax

```
IF condition THEN  
BEGIN  
    statement1;  
    statement2;
```

```

END
ELSE
BEGIN
    statement3;
    statement4;
END

```

Example

An example of when you may use conditional logic is if you need to evaluate whether an N1 or NAD group contains billing or shipping information (this depends on the qualifier that a field in the group contains), and then map that information to the appropriate application fields.

For this example, you need to add an On End extended rule to the N1/NAD. The rule is executed when the group terminates. An example of the syntax of the rule follows:

```

IF #0098 = "BT" THEN
BEGIN
    $Group_Name.#BILLTONAME = #0093;
    $Group_Name.#BILLTOADDR1 = #0166;
    $Group_Name.#BILLTOADDR2 = #0166:2;
    $Group_Name.#BILLTOCITY = #0019;
    $Group_Name.#BILLTOSTATE = #0156;
    $Group_Name.#BILLTOPCODE = #0116;
END
IF #0098 = "ST" THEN
BEGIN
    $Group_Name.#SHIPTONAME = #0093;
    $Group_Name.#SHIPTOADDR1 = #0166;
    $Group_Name.#SHIPTOADDR2 = #0166:2;
    $Group_Name.#SHIPTOCITY = #0019;
    $Group_Name.#SHIPTOSTATE = #0156;
    $Group_Name.#SHIPTOPCODE = #0116;
END

```

String Conditions and Functions

You can use string conditions in IF/THEN and IF/THEN/ELSE statements to perform comparisons between strings.

Examples of the syntax are as follows:

```

IF s1 = s2 THEN
IF s1 < s2 THEN
IF s1 > s2 THEN

```

The following string functions are also available for you to use:

- left
- right
- mid
- strdate
- concat
- strstr

left, right, mid syntax

The left, right, and mid functions enable you to extract substrings from a string. The left function extracts a specified number of characters from the left of the string variable or field and returns the result as a string. The right function extracts

a specified number of character from the right of the string variable and returns the result as a string. The `mid` function extracts from a specified position in the string to the right, for a specified number of characters. This is an example of how the statements are used.

```
string[10] s;  
string[3] s1;  
string[3] s2;  
string[4] s3;  
string[7] s4;  
  
s = "abcdefghij";  
s1 = left(s,3);  
s2 = right(s,3);  
s3 = mid(s,3,4);
```

strdate syntax

The `strdate` function converts a `datetime` type into a string using a format that you specify. This function allows you to include static characters such as a slash (/), which gives you access to full date support.

```
datetime d;  
string[8] s;  
  
strdate(d,"%y/%m/%d",s);
```

concat syntax

The `concat` function concatenates a specified number of characters from one string onto the end of another string. The following example demonstrates the syntax to concatenate five characters from string "s2" onto the end of string "s1":

```
string[10] s1,s2;  
concat(s1,s2,5);
```

strstr syntax

The `strstr` function finds a substring inside a string. This function returns the position of the first instance of the designated substring. If this function does not find the specified substring inside the string, it returns a value of -1.

```
integer d;  
  
d = strstr("hello", "el");
```

Numerical Functions

The numerical functions enable you to convert one data type to another.

The following are the available numerical functions:

- `len`
- `atoi`
- `aton`
- `ntoa`

len syntax

The `len` function counts and returns the number of characters in a string.

```
integer a;  
a = len("hello");
```

atoi, aton, ntoasyntax

The atoi function converts strings into integers.

The aton function converts string into real numbers.

The ntoas function converts integers and real numbers into strings.

```
integer a;  
real b;  
string[8] s;  
a = atoi("5");  
b = aton("5.5");  
ntoa(5.5, s);
```

atoi

The atoi function is a numerical function that converts strings into integers. The numerical functions enable you to convert one data type to another.

Common use

The atoi function is often used with SQL maps where data in the database is stored as string types. It is also used after manipulating a string value that contains both alpha and numeric characters, to attain the numeric value.

Syntax

```
int = atoi(string);
```

where:

- int = integer variable
- string = string variable

Example

```
integer a;  
string[20] s;  
s = "5";  
a = atoi(s);  
// "a" contains the value 5
```

aton

The aton function is a numerical function that converts strings into real numbers. The numerical functions enable you to convert one data type to another.

Common use

The aton function is often used with SQL maps where data in the database is stored as string types. It is also used after manipulating a string value that contains both alpha and numeric characters, to attain the numeric value.

Syntax

```
real = aton(string);
```

where:

- real = real number variable
- string = string variable

Example

```
real a;  
string[20] s;  
s = "5.5";  
a = aton(s);  
// "a" contains the value 5.5
```

auditlog

The auditlog function enables you to write user-defined audit messages to the Sterling Gentran:Server Audit Log. When an extended rule that calls an auditlog operation is executed, it writes the specified user-defined message to the Audit Log.

Syntax

```
auditlog(MessageID, Type, Key, [, string1] [, string2] [, string3] [, string4]  
[, string5] [, string6] [, string7];
```

Note: This function does not return a value.

Parameters

The first parameter, MessageID, is the user-defined audit message identifier, which must be an integer value.

The second parameter, Type, is a keyword that identifies the type of audit message. These pre-defined keywords are valid for Sterling Gentran:Server:

- AL_PROC (processing)
- AL_MSG (message)

Note: You may also supply an integer value to account for a type for which a keyword is not currently defined.

The third parameter, Key, is a keyword that is either zero (if the type of parameter two is AL_PROC) or the identification of a piece of data of the specified type. These pre-defined keywords are valid for Sterling Gentran:Server:

- AL_KEY_INPUT
- AL_KEY_OUTPUT

Note: You may also supply an integer value to account for a key for which a keyword is not currently defined.

Parameters four through ten are optional string values that the user-defined message may require to fill in variables defined in the audit message.

Examples

Example 1

```
auditlog(MessageID, AL_PROC, 0, ...);  
//Issues a processing message in any map.
```

Example 2

```
auditlog(MessageID, AL_MSG, 0atoi(param(1)), ...);  
//Issues a data audit for the message which is currently processing.
```

begin ... end

The begin/end keywords enclose a group of statements that form the body of an if/then/else statement or a while loop.

You can use the if/then keywords to execute one or more statements conditionally. If you include more than one statement in the body of an if/then loop, you must surround the statements with the begin/end keywords. If you only use a single statement, you can omit the begin and end.

Sterling Gentran:Server uses conditional logic to test conditions and then, depending on the results of the test, perform different operations. Conditions can be nested to any level.

Note: Do not end conditions with a semicolon (;) – this terminating syntax is necessary for statements only.

Syntax

```
if condition then
begin
    statement1;
    statement2;
end
```

Example

```
while MbxGetNextAtm(AtmId) != 0 do
begin
    MbxGetAtmFileName(MsgId, AtmId, FileName);
    FtpSndAtm(MsgId, AtmId, FileName);
end
```

break

The break keyword terminates the execution of the nearest enclosing while loop, and passes control to the statement that follows the end keyword. The break keyword is generally used in complex loops to terminate a loop before several statements have been executed.

Example

```
integer i
i = 0;

while i<10 do
begin
    #Total = Total + 50;
    i = i + 1;
    if #Total > 100000
        Break;
    else
        continue;
end
//While the value contained in the variable "i" is less than ten,
//50 will be added to the field Total.
//If the value in the field Total becomes greater than 100000
//before "i" equals 10, break out of the while loop else continue
//processing until "i" equals 10.
```

error

The error function raises a compliance error and reports the target statement (the statement you specify) on the translation report. You typically specify this function as an action to be performed if a condition is false.

This function is valid on the input side of a map only. There is also an optional third parameter you can supply: a string that is written to the translator report as part of the entry for the compliance error.

The error function can also be used with SWIFTNet to allow it to be called with only a code and description string (instead of code, field reference, option description string).

Table 26. When the error function is supported

If the map/form is of type ...	Then the error function is ...
Screen entry	Not valid.
Print	Not valid.
Export	Only valid on the input side of the map.
Import	Valid on the input or output side of the map.
Break (Interchange, Group, or Transaction Set)	Only valid on the input side of the map.
Build (Interchange, Group, or Transaction Set)	Not valid.

Common use

In addition to creating errors for user controlled validation, the error function is also used during debugging. The ability to pass a string to the error function allows you to use the error function the same way you would use a messagebox, with the results being written to the translator report instead.

Syntax

The error function can be specified in two ways.

Syntax 1

```
error(error_number,$GROUP_NAME[index][index][index].#FIELD_NAME,  
"Optional string with error information can be supplied here");
```

Syntax 2

```
error(code, "String with error information supplied here");
```

Examples

Syntax 1

```
error(100,$GROUPNAME[0][1][1].#FIELDNAME);  
//This raises compliance error 100 on the FIELDNAME field of the  
//specified instance of the GROUPNAME group. There is no optional error text  
//given.
```

Syntax 2

```

cerror(100, "Number not valid");
//This raises compliance error 100 with error text "Number not valid" in the
//translator report.

```

Compliance codes

Table 27. General Messages

Message Number	Message Type	Messages Generated
12	Information	Start time
13	Information	End time
14	Information	Blocks read
15	Information	Blocks written
19	Information	Execution time in milliseconds
20	Information	Translation object name
21	Information	Translation is lightweight
25	Warning	Block data unknown
100	Error	Mandatory data missing
101	Error	Insufficient repeats
102	Error	Too many repeats
110	Error	Incorrect data format
111	Error	Data not minimum length
112	Error	Data exceeds maximum length
113	Error	Invalid date
120	Error	Too many components
121	Error	Too many composite elements
122	Error	Unsupported data type
123	Error	Data conversion error
140	Error	Standard rule failure
142	Error	Standard rule: use code data missing
143	Error	Standard rule: data conversion error
170	Error	Extended rule failure
171	Error	Extended rule data conversion error
300	Error	Mandatory block missing
301	Error	Mandatory group missing
316	Error	Maximum usage exceeded
400	Error	Block processor initialization failure
401	Error	Field processor initialization failure
10001	Information	Block signature
10002	Information	Block count Note: This message will only be written if you create the cerror extended rule at the field level. If you call the cerror extended rule at any other level in your map, it will not be written because it is only applicable at the field level.

Table 27. General Messages (continued)

Message Number	Message Type	Messages Generated
10003	Information	Block name Note: This message will only be written if you create the error extended rule at the field level. If you call the error extended rule at any other level in your map, it will not be written because it is only applicable at the field level.
10004	Information	Field name Note: This message will only be written if you create the error extended rule at the field level. If you call the error extended rule at any other level in your map, it will not be written because it is only applicable at the field level.
10005	Information	Field data
10006	Information	Exception
10007	Information	Group name
10008	Information	Field ID
10009	Information	Field number
10010	Information	Instance
10011	Information	Rule type
10012	Information	On begin rule
10013	Information	On end rule
10014	Information	Repeat count
10015	Information	Block data
10016	Information	Block signature ID tag Note: This message will only be written if you create the error extended rule at the field level. If you call the error extended rule at any other level in your map, it will not be written because it is only applicable at the field level.
10017	Information	Map iteration count Note: This message will only be written if you create the error extended rule at the field level. If you call the error extended rule at any other level in your map, it will not be written because it is only applicable at the field level.
10018	Information	Additional information

Table 28. SQL Messages

Message Number	Message Type	Messages Generated
700	Error	SQL data source open error
701	Error	SQL data source rollback
702	Error	SQL data source commit error
703	Error	SQL data source rollback error
710	Error	SQL query open error
711	Error	SQL command error
712	Error	SQL cursor error
713	Error	SQL get field error
721	Error	SQL output operation error

Table 28. SQL Messages (continued)

Message Number	Message Type	Messages Generated
722	Error	SQL prepared statement error
724	Information	SQL update effected 0 rows
725	Information	SQL retrying as insert
726	Information	SQL retrying as update
10700	Information	Data source name
10701	Information	Data source pool
10702	Information	Query name
10703	Information	SQL statement
10704	Information	Cursor operation
10705	Information	Column ID

Table 29. EDI Messages

Message Number	Message Type	Message Generated
103	Information	Illegal repeating delimiter
104	Information	Illegal subelement delimiter
105	Information	Element position
106	Information	Subelement position

Table 30. XML Messages

Message Number	Message Type	Message Generated
610	Error	XML particle or group error
690	Error	XML parser error
691	Error	XML element unknown
692	Error	XML pcdata unknown
693	Error	XML attribute unknown
10060	Information	Public ID
10061	Information	System ID
10062	Information	Line number
10063	Information	Column number
10064	Information	Message
10065	Information	XML tag name
10066	Information	XML namespace URI

concat

The concat function concatenates a specified number of characters from one string onto the end of another string.

Common use

The concat function is used when values from two strings need to be concatenated together to form one string. It can also be used during debugging to create more detailed messages. For example:

```
String[50] msg;  
msg = "Field A: ";  
concat(msg,#fielda,15);  
messagebox(msg,0);  
// Instead of outputting the contents of #fielda in a messagebox,  
// it will output a label of "Field A:" along with the contents.
```

Syntax

```
concat(string,string,num_char);
```

where:

- string = string variable
- num_char = number of characters from the second string onto the end of the first string

Note: You may not use an ActiveX property as the first parameter of the concat function because the length of the property is unknown prior to compilation.

Example

```
string[20] s1,s2;  
s1 = "IBM";  
s2 = "Corporation";  
  
concat(s1,s2,8);  
  
//Concatenate eight characters from string "s2" onto the end of string "s1".  
//s1 will now contain the value "IBM Corporat".
```

continue

The continue keyword continues the execution of the innermost loop without processing the statements in the loop that follow the continue statement.

Example

```
integer i;  
  
i = 0;  
while i<10 do  
begin  
    i = i + 1;  
    if (i = 8) then  
        continue;  
    if (i = 9) then  
        break;  
end  
//As long as "i" has a value less than "10" the loop repeats.  
//If "i" has a value of "8", the loop continues. If "i" has a  
//value of "9" the loop terminates.
```

count

The count function counts and returns the number of iterations of a group.

Common use

The count function is often used in conjunction with while/do loops, so you do not need to keep track of counters for all sub-groups. See the While Do white paper for some examples that use the count function.

The count function is sometimes used with indexes, instead of using variables, but often it is less efficient than using variables.

Note: When a count extended rule is performed on an empty group, the value of -1 is returned from count(\$GROUPNAME[*]).

Syntax

```
integer i;  
i = count($N1[*]);  
//The [*] is a wildcard that counts the number of iterations of the N1 group.
```

Example

```
integer i;  
i = count($GROUPNAME[*]);  
//The [*] is a wildcard that counts the number of iterations of the  
//GROUPNAME group.
```

createobject

The createobject function enables you to create an instance of an ActiveX Automation Server.

Syntax

```
object = createobject("ProgID");
```

where:

- ProgID = programmatic identifier

Example

```
object ob;  
ob = createobject("InternetExplorer.Application");  
//Creates an instance of the default interface of an ActiveX  
//Automation Server.  
//Note:  
//The createobject command is more efficient if you use the IID  
//instead of the interface name.
```

date

The date function converts a string type into a datetime type using a format that you specify. This function allows you to include static characters such as a slash (/), which gives you access to full date support.

Common use

The date function is commonly used in SQL maps to convert a date that is stored in the database as a string type into a datetime variable or field. It is also used in print forms. Because the document name or ref data can only be updated with a standard rule against a string field, a date field has to be defined as string then converted if necessary.

Syntax

```
Datetime = date("format",string);
```

where:

- datetime = datetime variable (month specified as 1-12)
- format = desired date format
- string = string variable

Example

```
datetime d;  
d = date(2012,4,6);  
d = date(2012,4,6,12,0);  
d = date("%y/%m/%d","12/4/6");  
d = date("%y/%m/%d",#strdate);
```

Format specifiers

Table 31. Format specifiers

Format Specifier	Description
%8	ISO-8601 date format YYYYMMDDTHHMMSS.mmmZ Four-digit year, two-digit month, two-digit day, T (time) indicator, two-digit hour, two-digit minutes, two-digit seconds in Universal Time (also called Zulu Time or Greenwich Mean Time), Z (Zulu time) indicator (example: 20031209T123000.000Z) Note: This date format cannot be combined with any other format specifier.
%a	Abbreviated weekday name
%A	Full weekday name
%b	Abbreviated month name
%B	Full month name
%d	Day of the month as a decimal number (01 – 31)
%D	ISO-8601 date format (date component only) YYYYMMDDZ This date format cannot be combined with any other format specifier.
%H	Hour in 24-hour format (00 – 23)
%I	Hour in 12-hour format (01– 12)
%j	Day of the year as a decimal number (001 – 366)
%m	Month as a decimal number (01 – 12)
%M	Minute as a decimal number (00 – 59)
%S	Second as a decimal number (00 – 59)
%U	Week of the year as a decimal number, with Sunday as the first day of the week (00 – 51)
%w	Weekday as a decimal number (0 – 6, with Sunday as "0")
%W	Week of the year as a decimal number, with Monday as the first day of the week (00 – 51)
%y	Year without the century as a decimal number (00 – 99)

Table 31. Format specifiers (continued)

Format Specifier	Description
%Y	Year with the century as a decimal number
%%	Percent sign

delete

The delete function deletes a specified iteration of a repeating record or group.

Common use

The delete function is often incorrectly used at field level or On End of the group to try to delete the current iteration. The translator cannot delete the current iteration. Because of this, many people choose to copy the iterations they do want to a temp group instead.

Syntax

```
delete(GROUPNAME[iteration]);
```

where:

- iteration = the occurrence of the group that you want to delete

Example

```
delete($ILD[2]);  
//Deletes the second occurrence of the ILD group.
```

deleteobject

The deleteobject function enables you to delete an instance of an ActiveX Automation Server. An object must be deleted before the end of the map that uses it. It is more efficient to delete the object immediately on completion, although the Sterling Gentran:Server translator will delete the object automatically at the end of the map. Also, if you assign one object to another one, both copies of the object must be deleted for that object to be properly unloaded.

Syntax

```
deleteobject(object);
```

Example

```
object ob;  
ob = createobject("InternetExplorer.Application");  
deleteobject(ob);  
//Deletes the instance of the object.
```

empty

The empty function sets the value of a field in internal storage to null. This function is not the same as setting the value of a field to a zero length string (" ") or to zero.

Common use

The empty function is often misused to try to empty all of the fields of a record that is not wanted. This will cause an empty iteration for the record and all

occurrences after the empty iteration will not be processed.

Syntax

```
empty($GROUP_NAME[index][index][index].#FIELD_NAME);  
//Set the value of the specified instance of the FIELDNAME field to null.
```

Example

The following example sets the value of the specified instance of the VATC element to null. You typically use this function to prevent output to the specified field.

```
empty($ILD.#VATC);
```

exec

A user exit is an extended rule that enables the map to temporarily exit translation to enhance your functionality or fulfill specific requirements that Sterling Gentran:Server does not perform during normal translation. The User Exit (exec) function invokes the execution of a batch file or program.

The translator waits until the script finishes before continuing with translation. After the script runs and returns a numeric return code, the exec function:

- Retrieves the return code
- Returns to translation
- Uses the return code in translation.

Note: You must set an integer value equal to the return value of the exec (...) call for the rule to compile.

Common use

You can use the exec function in an extended rule for a map component at any hierarchical level, including field level extended rules, if appropriate.

When you apply a user exit to a map component, subordinate map components may also be able to execute the same user exit.

The following are the extended rule data types that you can use with exec:

- INTEGER
- REAL
- STRING

You cannot use date and time data unless you process it as a string data type.

Syntax

```
nReturn = exec(string)
```

where:

- nReturn = return value
- string = string variable or literal value that represents the shell script

Example

```
integer nReturn;  
nReturn = 0;  
nReturn = exec ("c:\addrnm.sh");
```

exist

The exist function tests to determine if a field is empty (null). It returns a non-zero (true) value if there is data in a specified field in internal storage. If data is not present in the specified field, this function returns a zero (false) value. This function is typically used as a part of a condition.

There are some situations when the if exist returns a non-zero (true) value whether or not the condition is true (for example, if the field or element has a "Use Code" standard rule applied to it). You can work around this by only using if exist for date- and number-type fields. Be certain that all references to the field which is interrogated are nested within the if exist begin block.

Note: For string-type fields, use the format `if field1 = ""`.

All modes of operation are exercised by SWIFT MX and FIN maps in the exist extended rule. FIN maps reference the traditional usage (`$group.#field`) of the exist function while the MX maps reference only the group name (`$group`).

Note: The group name only usage is reserved for the XML syntax only, because of how the blocks are inserted into the map structure during the compilation process to handle the XML start and end tags in an XML document.

The exist function accepts a block reference denoted with a % prefix, as well as a group reference denoted with the \$ prefix. The group reference supports a scenario in which an element was supplied in the input file but none of its conditional children existed. Instead of using variables to test for the condition of a parent element (parentNode) you can just use the group reference (only for XML). If you wanted to check for the existence of the parentNode in this scenario, you can add a flag to the extended rule of the parentNode to determine this condition if the child fields are missing. For example:

```
integer p;  
p = exist($parentNode);
```

Common use

The exist function is more often used as !exist (not exist) to store a default value into a field if the field does not exist. The exist function is often used with segments such as SDQ where there can be multiple pairs of information for stores and quantities. You can check to make sure a pair exists before attempting to manipulate the value, for example summing the quantities. The exist function is also used to only output a qualifier if the field it's qualifying exists.

Syntax

```
if exist($GROUP_NAME[index][index][index].#FIELD_NAME) then
```

Example

```
if exist(#FIELD A) then  
#FIELD B = "EA";  
//Return a non-zero value if the condition is true (data is present in  
//the specified instance of the FIELD A field). A zero value is  
//returned if the condition is false (no data is present in the  
//specified instance of the FIELDNAME field).  
  
if !exist(#FIELD C) then  
#FIELD C = "100";  
//Populate FIELD C with the value of 100 if it does not exist already.
```

fseek

The `fseek` function moves the file pointer to a new location, which is a specified number of bytes (offset) from the designated point of origin in the file (the point of origin may be the beginning of the file or relative to either the end of the file or the current position).

To invoke the `fseek` function against the input file, the value for `current_file` is 0. To invoke the `fseek` function against the output file, the value for `current_file` is 1. The `fseek` function is typically only used in conjunction with the `ftell`, `readblock`, and `writeblock` functions.

Syntax

```
fseek(current_file,offset,origin);
```

where:

- `current_file` = 0 indicates the input file, 1 indicates the output file
- `offset` = position to which the file pointer is moved relative to the origin
- `origin` = keyword depends on the starting location:
 - `begin` = start at beginning of the file
 - `end` = start at the end of the file
 - `current` = start at the current position in the file

Example

```
string[1024]temp_buffer;
Integer Position;
Position = ftell(0);
while readblock(temp_buffer) do
begin
    if left(temp_buffer,3) = "IEA" then
        begin
            fseek(0,Position,begin);
            break;
        end
    writeblock(temp_buffer);
    Position = ftell(0);
end
//Read a segment from input file and place in temp_buffer. Look for
//"IEA" segment tag. If found, reset file pointer to where it was
//before the "IEA" segment was read. Write contents of temp_buffer
//to output file. Set "Position" = the current file pointer position.
```

ftell

The `ftell` function obtains the current position of the file pointer and returns it as an integer. To invoke the `ftell` function against the input file, the value for `current_file` is 0. To invoke the `ftell` function against the output file, the value for `current_file` is 1. The `fseek` function is typically only used in conjunction with the `fseek`, `readblock`, and `writeblock` functions.

Syntax

```
numeric_variable = ftell(current_file);
```

Example

```
string[1024]temp_buffer;
Integer Position;
Position = ftell(0);
while readblock(temp_buffer) do
```

```

begin
  if left(temp_buffer,3) = "IEA" then
    begin
      fseek(0,Position,begin);
      break;
    end
    writeblock(temp_buffer);
    Position = ftell(0);
  end
  //Read a segment from input file and place in temp_buffer. Look for
  //"IEA" segment tag. If found, reset file pointer to where it was
  //before the "IEA" segment was read. Write contents of temp_buffer
  //to output file. Set "Position" = the current file pointer position.

```

get

The get function enables you to access individual components of a datetime variable. Valid datetime components are year, month, day, hour, minute, second.

Common use

The strdate function was added to extended rules after the get function. The strdate function is used more often than the get function because it is more versatile.

Syntax

```
integer_variable = get datetime_component (datetime_variable);
```

where:

- integer_variable = integer variable
- datetime_component = individual component of the datetime variable
- datetime_variable = datetime variable of which you want to access a component part

Example

```

integer temp_days;
integer temp_hours;
datetime d;
temp_days = 0;
temp_hours = 0;
d = '12/25/2001 12:15:30';
//A fields value in the map can be assigned to the datetime variable
//"d" or a hard coded value can be assigned.

temp_days = get days (d);
temp_hours = get hours (d);
//Accesses the days from the datetime variable "d" and loads into
//variable " temp_days ". Accesses the hours from the datetime
//variable "d" and loads into variable "temp_hours".

```

getiid

The getiid function enables you to obtain the unique identifier for an interface, by using the string-character name of the interface to return the globally unique identifier that is used by software to run the interface.

Syntax

The following command creates an instance of the default interface of an ActiveX Automation Server.

```
string_variable = getiid("ProgID");
```

The following command looks up the InterfaceID (IID) of an interface.

```
string_variable = getiid("ProgID", "Interface_Name or {Interface ID}");
```

Example

```
object ob;  
string[50] iid;  
iid = getiid("InternetExplorer.Application", "IWebBrowser2");  
ob = createobject("InternetExplorer.Application", iid);  
ob.Visible = 1;  
//Displays the Internet Explorer on the desktop by setting a  
//property value in an ActiveX Automation Server.
```

if ... then ... else

The if, then, and else keywords allow the use of conditional logic. Sterling Gentran:Server uses conditional logic to test conditions and then, depending on the results of the test, perform different operations.

Conditions can be nested to any level. You can use the if/then keywords to run one or more statements conditionally. The condition is typically a comparison, but it can be any expression that concludes with a numeric value. Sterling Gentran:Server interprets the value as either true or false. The system interprets a zero value as false and a nonzero value as true.

Sterling Gentran:Server evaluates the if/then condition, and if it is true, the system runs all the statements that follow the then keyword. If the condition is false, none of the statements following then are run.

You can use the else keyword in conjunction with if/then to define several blocks of statements, one of which is run. Sterling Gentran:Server tests the first if/then condition. If the condition is false, the system proceeds to test each sequential condition until it finds one that is true. The system runs the corresponding block of statements for the true condition. If none of the if/then conditions are true, the system runs the statements following the else keyword.

The begin/end keywords enclose a group of statements that form the body of an if/then/else statement. You can use the if/then/else keywords to run one or more statements conditionally. If you include more than one statement in the body of an if/then statement, you must surround the statements with the begin/end keywords. If you use only a single statement, you can omit the begin/end.

Note: Do not end conditions with a semicolon (;) – this terminating syntax is necessary for statements only.

Example

```
if condition then  
begin  
    statement1;  
    statement2;  
end  
else condition then  
begin  
    statement3;  
    statement4;  
end
```

index

The index function enables you to determine which instance of a particular loop the translator is currently accessing.

Common use

More often than not, variables are used to keep track of the current loop count instead of the function. The syntax states that you specify an integer variable for the parameter, but a constant is normally used instead of a variable. The index function can be used to determine if you are at a specific iteration of a group such as:

```
If index(2) = 1 then  
...
```

Or

```
If index(1) = 10 then  
...
```

The first example will check to see if the translator is at the first child group for the current parent. The second example will check to see if the translator is at the 10th iteration of the Parent group.

The Index function can also be used when using indexes to reference a field such as:

```
$SUB_GROUP[index(1)][index(2)][x].#FIELD = "TEXT";
```

This use would normally be used when you need to reference a repeating subgroup that is at the same level or a different branch of the group where the rule is written.

Syntax

```
index(integer_variable);
```

where:

- integer_variable = integer variable that indicates the hierarchical level for which you want to determine the loop count

Example

```
integer x;  
x = index(1);
```

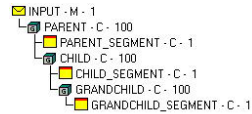
```
//This will populate x with the current loop count / iteration for the  
//outer most parent group, from where the rule is written.  
//The group will be located off of the root level of the map.
```

```
x = index(2);
```

```
//This will populate x with the current loop count / iteration for the  
//first child group, from where the rule is written.  
//The group will be a child to the parent group off the root level.
```

```
x = index(3);
```

```
//This will populate x with the current loop count / iteration for the  
//first grandchild group, from where the rule is written.  
//The group will be a grand child to the parent group off the root  
//level, and a child to the first child group.
```

insert

The insert function allows information in the database tables to be updated.

Syntax

```
insert into tablename [ (fieldlist) ] [ (valuelist) ];
```

where:

- tablename = one of the following:
 - DivisionLookup
 - PartnerLookup
 - DivisionLocation
 - PartnerLocation
 - DivisionXref
 - PartnerXref
- fieldlist = (fieldname [, fieldname])
- fieldname = name of one of the fields in the table
- valuelist = (String [, String])

Notes:

- The fieldlist lists one or more fieldnames to which data is to be added.
- The fieldnames can be listed in any order.
- The valuelist must be in the same order as the field list.

Example

```
updateStatus = update PartnerLookup
set Description = "Lookup Update Test",Text1="Text1Updated",Text2="Text2Updated",
Text3="Text3Updated",Text4="Text4Updated"
where TableName = "PartLkp" and Item = "1";

if updateStatus = 1 then
begin
  messagebox("Update PartnerLookup: Record Not Found,Attempting Insert...",1);
  insertStatus = insert into PartnerLookup( PartnerKEY, TableName,Item,Description,
    Text1,Text2,Text3,Text4)
  values ("PETZONE","PartLkp","1","Lookup Insert Test","Text1","Text2","Text3","Text4");
  if insertStatus = 0 then
  begin
    messagebox("Insert PartnerLookup: Failed",1);
  end
end

if updateStatus = 2 then
begin
  messagebox("Update PartnerLookup Failed with OtherError",1);
end//
Example assumes that a lookup table named "PartLkp" has been created.
```

left

The left function extracts a specified number of character from the left side of a string variable or field and returns the result as a string.

Common use

The left function is used when you only need the first part of a string. If you only want the first five digits of a zipcode, use the following example:

```
#TEMP_ZIP = left(#ZIP_CODE,5);
```

The left function is also commonly used in conjunction with the len function, to "drop" characters from the end of a string. If you want to drop the last three characters of a string, use the following example:

```
#TEMP = left(#FIELD,len(#FIELD) - 3);
```

The right, left, and len functions can all be used together. If you want to remove 0s from the beginning of an ID, but there is not a set number of 0s, use the following example:

```
while left(#ID,1) = "0" do  
#ID = right(#ID,len(#ID)-1);
```

Syntax

```
string_variable = left(string_variable,num_char);
```

where:

- string_variable = A variable defined as type string.
- num_char = integer variable

Note: You may not use an ActiveX property as the first parameter of the left function because the length of the property is unknown prior to compilation.

Example

```
string [25]name;  
string [5]temp_variable;  
name = "Acme Shipping Company"  
temp_variable = left(name,4);  
// "temp_variable" would contain "Acme"
```

len

The len function is a numerical function that counts and returns the number of characters in a string. The numerical functions enable you to convert one data type to another.

Common use

The len function is most often used inline with other functions, such as left and right. It is also used within a while/do loop to pad a string to a specific length. If you need to add 0s to the front of a string to make the string 10 characters, use the following example:

```
while len(#field) < 10 Do  
#field = "0" + #field;
```

Syntax

```
number_char = len(string);
```

where:

- num_char = integer variable
- string = The string you wish to evaluate.

Example

```
integer a;  
a = 0;  
a = len("hello");  
// "a" contains the value 5
```

messagebox

The messagebox function enables you to display a message box for which you have designated the format and content. You can specify the number and type of buttons on the message box, the message icon (for example, hand, question mark, exclamation point, or asterisk), and the message displayed. You can also issue a return value based on the chosen action.

Common use

The messagebox function is used to help debug a map. Messages placed throughout a map can help determine where a hung map is hanging. Multiple messages can be combined into one string to avoid confusion about the results. Instead of receiving three messageboxes with the values of three fields, you can combine them all into one messagebox, with labels. For example:

```
String[100] msg;  
  
msg = "field1: " + #field1 + " field2: " + #field2 + " field3: " + #field3;  
messagebox(msg,0);
```

This will output one string with the values for the three fields in line:

field1: value field2: value field3: value

Note: Only String values can be displayed in a message box.

Syntax

```
messagebox("message",defined_number);
```

where:

- message = message string
- defined_number = defined number of the desired buttons plus the defined number of the desired icon (if used)

The defined numbers for the button and icon types are as follows:

- 0 = OK button only
- 1 = OK and Cancel buttons
- 4 = Yes and No buttons
- 16 = Icon Hand
- 32 = Icon Question Mark
- 48 = Icon Exclamation Point
- 64 = Icon Asterisk

The message box return values are as follows:

- 1 = OK selected
- 2 = Cancel selected
- 6 = Yes selected
- 7 No selected

Example

```
if messagebox("Do you really want to delete this object?",36) = 6
begin
.
.
.
end
//Displays a message box with the given string as the message, Yes
//and No buttons (4) and a question mark icon (32). The number and
//type of buttons (4) plus the icon (32) equals the defined_number(36).
//If the user clicks the Yes button (return value "6"), the
//statements in the begin/end loop are executed.
```

Defined_ numbers

Table 32. defined_numbers for the button and icon types

Defined_Number	Button or Icon Types
0	OK button only
1	OK and Cancel buttons
4	Yes and No buttons
16	Icon Hand
32	Icon Question Mark
48	Icon Exclamation Point
64	Icon Asterisk

Message box return values

Table 33. Message box return values

Return Value	Action Selected
1	OK selected
2	Cancel selected
6	Yes selected
7	No selected

mid

The mid function extracts from a specified position in a string, either to the end of the string or for a specified number of characters and returns the resultant string. This function is zero-based.

Common use

The mid function is often used with the strstr function. The strstr function will return the position of a specific character, then the mid can be used to return a substring from that character.

Because the mid function is zero-based, it is often used incorrectly, with the resultant string off by one character. An easy way to determine the correct starting position is to envision a cursor and count how many times you need to move it to get to the starting position you want. See the example below.

Syntax

```
string_variable = mid(string_variable,start_pos,num_char)
```

where:

- `string_variable` = The variable containing the string you want to extract.
- `start_pos` = The starting position in the string of characters (integer).
- `num_char` = The number of characters from the starting position (integer).

Note: You may not use an ActiveX property as the first parameter of the mid function because the length of the property is unknown prior to compilation.

Example

```
string [25]name;  
string [10]temp_variable;  
name = "Acme Shipping Company"  
temp_variable = mid(name,5,8);  
//The map will read 8 characters in the string starting with  
//the sixth character. It is essentially ignoring the first  
//five characters, so "temp_variable" will contain "Shipping".
```

ntoa

The ntoa function is a numerical function that converts real numbers into strings. The numerical functions enable you to convert one data type to another.

Common use

The ntoa function is often used when you cannot change the data type for a field, such as when you are writing to a database. The ntoa function is also used to assist in debugging. For example, because you cannot use numeric fields in a messagebox, you must convert the value to a string first.

```
real b;  
string[20] s, msg;  
b = 5.5;  
ntoa(b, s);  
msg = "b: " + s;  
messagebox(msg,0);  
//The messagebox output will contain "b: 5.5"
```

Syntax

```
string = ntoa(real,string);
```

where:

- `real` = The real number variable you wish to convert.
- `string` = The name of the string in which you want to store the converted number.

Example

```
real b;  
string[20] s;  
b = 5.5;  
ntoa(b,s);  
//The variable "s" contains the string "5.5".
```

param

The param function is used to read the value of the PARAM(n) variable. The extended rule allows you to reference values that have been passed into the translator via the command line.

Introduction

The -u switch can be used during the invocation of the translator to pass values in so they can be referenced by an extended rule. Typically, the param extended rule is not used by maps that are running within the Sterling Gentran:Server environment because Sterling Gentran:Server does not use the -u switch when invoking the translator. However, if you are invoking the translator to perform translation outside of Sterling Gentran:Server (tx32 -f <inputfile> <templatefile> <outputfile> <reportfile>), you have the option to pass values into the translator using the -u switch and reference those values using the param extended rule.

Syntax

```
param(integer_PARAM_number);
```

where:

- integer_PARAM_number = the number of the pre-defined variable

Example

For example, if you invoke TX32.EXE in the following manner:

```
tx32.exe -f input.txt 850.tpl output.txt out.rpt -u "PurchaseOrderNumber" -u 12345  
-u 500.25
```

Then you could write an extended rule in the \GENSRVNT\Tutorial\Pet_850.map that looks like the following:

```
string [32] strDescription;  
string [32] strPONbr;  
string [32] strTotalCost;  
real nTotalCost;  
  
strDescription = param(0);  
strPONbr = param(1);  
strTotalCost = param(2);  
nTotalCost = aton(strTotalCost);  
  
if nTotalCost > 200 then  
begin  
    messagebox("Get approval from boss",0);  
end
```

You can run this rule from any scope in your map (for example, pre-session, post-session, group onbegin, field).

queryobject

The queryobject function is used to request a different interface on an existing object.

Syntax

```
object2 = queryobject(object1, "{IID}");
```

where:

- object2 = is the result that contains the requested interface to the object
- IID = is the interface identifier of the requested interface
- object1 = an existing object

Example

```
object ob, ob2;  
ob = createobject("InternetExplorer.Application");  
ob2 = queryobject(ob, "{EAB22AC1-30C1-11CF-A7EB-0000C05BAE0B}");  
//Uses the Interface ID of the desired interface to obtain another  
//(different) interface of the existing object (object1).
```

readblock

The readblock function reads a block of data (segment or record) from the input file and places it into the argument of a string variable.

The readblock and writeblock functions are used in conjunction with each other to pass a block of data from the input file to the output file without compliance checking or testing for proper EDI syntax. Together these functions provide a more efficient alternative of using "wildcard" segments, which are typically implemented in build and break maps.

The readblock and writeblock functions are also used in conjunction with the Document Extraction service to specify the beginning and end of each document in a batch of documents, so that each document can be extracted individually. See the Example 2, below.

The readblock function returns a zero value if it does not read any data. However, if readblock returns a zero value, you should not assume the translator has reached the end of the file. If the data file has a number of new lines embedded in it, the readblock function returns a zero for each new line. If you want to know for certain when the end of the file is reached, use the eof function.

Notes:

- Readblock, writeblock, and unreadblock are supported only for positional and EDI files.
- You may not use an ActiveX property as the first parameter of the readblock function because the length of the property is unknown prior to compilation.
- See the *IBM Sterling Gentran:Server for Microsoft Windows XML User Guide* for special considerations when using this function with XML data.

Syntax

```
readblock(string_variable);
```

Examples

Example 1

```

String[1024] buffer;

readblock(buffer);
writeblock(buffer);

while readblock(buffer) do
begin
    if left(buffer,3) = "HDR" then
        begin
            unreadblock();
            break;
        end
    writeblock(buffer);
end
//Read a block from the input file and place it in buffer. Look for
//a "HDR" record tag. If found, reset the file pointer to where it was
//before the "HDR" record was found. Write contents of the buffer to
//the output file.

```

Example 2

```

string[250] buffer;
string[3] match;
integer match_len;
integer eofInput;
// set these next two variables
match = "SUM"; // the tag of the last record in the document
match_len = 3; // the length of the tag
// read the block we're on and write it
readblock(buffer);
writeblock(buffer);
eofInput = eof(0); // check if we are at the end of the input document
// keep reading and writing records until the end of the document
while !eofInput do
begin
    if readblock(buffer) then
        begin
            writeblock(buffer);
            if left(buffer, match_len) = match then
                //write the document, not new lines and continues to process documents
                begin
                    break;
                end
            end
        end
    eofInput = eof(0);
    // check if we are at the end of the input document
end

```

readbytes

The readbytes function reads a number of bytes from the input file. This function is used in conjunction with the writebytes function. Used together, the readbytes and writebytes function provide an efficient method of passing data through a map if the data does not need to be compliance checked or altered in any way.

Readbytes is similar to the readblock function, but readblock only works with entire blocks (for example, an entire segment or record), and readbytes works with any quantity of data, whether it is smaller or larger than a block.

The readbytes function uses two parameters, first the string variable into which the data being read will be stored, and second the number of bytes to read.

Syntax

```
readbytes(read_from_buffer, num_bytes);
```

where:

- `read_from_buffer` = string variable into which the data being read will be stored
- `num_bytes` = integer value representing the number of bytes to read from the input file.

Note: The `readbytes` function returns the number of bytes it was actually able to read.

Example

```
string [1024] tempBuffer;  
while readbytes(tempBuffer,1024) do  
begin writebytes(tempBuffer,1024);  
End  
//Read 1024 bytes from input file and place in string variable  
//named tempBuffer.  
  
writebytes("^OD^OA",2);  
//Appends a CRLF to the end of the output file.
```

right

The `right` function extracts a specified number of characters from the right side of a string variable or field.

Common use

The `right` function is used when you only need the last part of a string. If you only want the last four digits of a social security number, use the following example:

```
#TEMP_SS = right(#SOCIAL,4);
```

The `right` function is also commonly used in conjunction with the `len` function, to "drop" characters from the beginning of a string. If you want to drop the first three characters of a string, use the following example:

```
#TEMP = right(#FIELD,len(#FIELD) - 3);
```

The `right`, `left`, and `len` functions can all be used together. For example, if you want to remove 0s from the end of an ID, but there is not a set number of 0s, use the following example:

```
while right(#ID,1) = "0" do  
#ID = left(#ID,len(#ID)-1);
```

Syntax

```
string_variable = right(string_variable,num_char)
```

where:

- `string_variable` = The name of the string of characters you wish to manipulate.
- `num_char` = The number of characters to count from the right side of a string.

Note: You may not use an ActiveX property as the first parameter of the `right` function because the length of the property is unknown prior to compilation.

Example

```
string [25]name;  
string [10]temp_variable;  
name = "Acme Shipping Company"  
temp_variable = right(name,7);  
// "temp_variable" would contain "Company"
```

select

The select function allows information to be retrieved from the database tables. Only the tables and fields available in the select standard rule are available for the select extended rule.

Common use

The select function is often used as an extended rule instead of a standard rule when you only want to run it based on other criteria. For example, if you want to pull a value from Process Data if a field was not included in the data, use the following example:

```
If !exist(#Sender) then  
    Select xpathresult into #Sender from processdata where xpath = "\sender\text()";
```

It is also commonly written as an extended rule when there are multiple select statements to be performed because the standard rule only allows one per field.

Syntax

In the command syntax, expression and receiverlist can be string fields, string variables, or string literals. It is important to note that the table and field names for the select extended rule are slightly different than those depicted in the standard rule.

```
select fieldname into receiverlist from tablename where key = expression  
[and key = expression];
```

Example

```
string[50] var;  
select xpathresult into var from processdata where xpath="example";
```

set

The set function enables you to define individual components of a datetime variable. Valid datetime components are year, month, day, hour, minute, second.

Common use

The date function was added to extended rules after the set function. The date function is used more often than set because it is more versatile.

Note: Setting an integer value higher than the logical limit for the component will increase the corresponding related component. For example, if you set a value of 14 for months, it will increase the year by 1 and use the value 2 for the months. If you use the value 80 for minutes, it will increase the hours by 1 and use 20 for the minutes.

Syntax

```
set datetime_component (datetime_variable, integer_variable);
```

where:

- `datetime_component` = The individual component of the datetime variable.
- `datetime_variable` = The datetime variable of which you want to access a component part.
- `integer_variable` = An integer variable

Example

```
integer a, b, c;  
datetime d;  
a = 5;  
b = 3;  
c = 11;  
set months (d,a);  
set days (d,b);  
set hours (d,c);  
//Defines the months of the datetime variable "d" from variable "a"  
//Defines the days of the datetime variable "d" from variable "b".  
//Defines the hours of the datetime variable "d" from variable "c".
```

strdate

The `strdate` function converts a datetime type into a string using a format that you specify. This function allows you to include static characters such as a slash (/), which gives you access to full date support.

Common use

The `strdate` function is often used when you cannot change the data type for a field, such as when you are writing to a database. The `strdate` function is also used to assist in debugging. Because you cannot use a date field in a messagebox, you must convert the value to a string first.

```
string[20] s, msg;  
  
strdate(#datefield, "%m/%d/%Y", s);  
msg = "Date: " + s;  
messagebox(msg, 0);  
//The messagebox output will contain "Date: value"
```

The `strdate` function is also used to determine shipping days of the week, and adjust accordingly. For example, if you do not ship on Sundays, you can check if `%w` returns a 0 and if so, add a day to make it Monday.

```
String[10] shipday;  
  
strdate(#ship_date, "%w", shipday);  
if shipday = "0" then  
    #ship_date = #ship_date << days(1);
```

Syntax

```
strdate(datetime, "format", string);
```

where:

- `datetime` = datetime variable (month specified as 1 - 12)
- `format` = desired date format (see Format specifiers, below)
- `string` = string variable

Example

```
datetime d;  
string[8] s;  
d = date(2012,4,6);  
s="";
```

```
strdate(d,"%y/%m/%d",s);
```

//Converts a datetime variable into an eight character string in the
//format "year/month/day", in this case 2012/4/6.

Format specifiers

Table 34. Format specifiers

Format Specifier	Description
%8	ISO-8601 date format YYYYMMDDTHHMMSS.mmmZ Four-digit year, two-digit month, two-digit day, T (time) indicator, two-digit hour, two-digit minutes, two-digit seconds in Universal Time (also called Zulu Time or Greenwich Mean Time), Z (Zulu time) indicator (example: 20031209T123000.000Z) Note: This date format cannot be combined with any other format specifier.
%a	Abbreviated weekday name
%A	Full weekday name
%b	Abbreviated month name
%B	Full month name
%d	Day of the month as a decimal number (01 – 31)
%H	Hour in 24-hour format (00 – 23)
%I	Hour in 12-hour format (01– 12)
%j	Day of the year as a decimal number (001 – 366)
%m	Month as a decimal number (01 – 12)
%M	Minute as a decimal number (00 – 59)
%S	Second as a decimal number (00 – 59)
%U	Week of the year as a decimal number, with Sunday as the first day of the week (00 – 51)
%w	Weekday as a decimal number (0 – 6, with Sunday as "0")
%W	Week of the year as a decimal number, with Monday as the first day of the week (00 – 51)
%y	Year without the century as a decimal number (00 – 99)
%Y	Year with the century as a decimal number
%%	Percent sign

strstr

The strstr function finds a substring inside a string. This function returns the position of the first instance of the designated substring within the specified string. If this function does not find the specified substring in the string, it returns a value of -1. This function is zero-based.

Common use

The `strstr` function is often used with the `mid` function to return a substring. For example, if you wanted to extract a 10-digit PO number that is listed after a slash, use the following example:

```
String[10] po_number;  
po_number = mid(#PONUM, strstr(#PONUM, "/"), 10);
```

If you do not know the length of the substring, the `strstr` function can also be used to determine how many characters are to the right of the character, by subtracting the position returned by `strstr` from the length using `len`.

```
String[20] po_number;  
po_number = mid(#PONUM, strstr(#PONUM, "/"), (len(#PONUM) - strstr(#PONUM, "/")));
```

Syntax

```
integer = strstr("string", "substring");
```

where:

- `integer` = integer variable
- `string` = the string to evaluate
- `substring` = the part of the string you are interested in

Examples

```
integer d;  
d = 0;  
d = strstr("mississippi", "is");  
//Finds the first instance of the substring "is" inside the string  
//"mississippi" and returns the position of that first substring.
```

To use this function to enable a purchase order number to be processed differently depending on its format (for example, if the third position of the purchase order number is numeric do "X," otherwise do "Y"), use the following example:

```
integer position;  
string [1] PONumChar2;  
PONumChar2=mid(#PONumber, 1, 1);  
position=strstr("0123456789", PONumChar2);  
  
//This function finds a substring within the string. So if the second  
//position of purchase order number is not equal to -1 then it is  
//numeric and "X" should be executed. Otherwise, the second position is  
//not numeric and "Y" should be executed.
```

unreadblock

The `unreadblock` function provides a method of moving the input file-pointer back one block (a block of data is equivalent to one EDI segment or one positional record). This function unreads the block of data that was just processed by the `readblock` function.

Note: The `unreadblock` function works only once and only for the most recent `readblock`. It can only be used in conjunction with the `readblock` function. The `unreadblock` function will only unread the most recent block of data processed. If you use `unreadblock` more than once, you will not be able to point to any earlier `readblocks`.

The `unreadblock` function is provided as an alternative to the `fseek` and `ftell` functions, and is the preferred method of moving the file pointer back one block of

data. The `unreadblock` function allows the translator to correctly track the number of bytes read and number of segments read during the translation process by moving the file-pointer back and decrementing the segment and byte counts accordingly.

The `unreadblock` function is commonly used with the `readblock` and `writeblock` functions to pass blocks of data in bulk from the input file to the output file. This is useful for maps that are designed to envelope data.

`Readblock`, `writeblock`, and `unreadblock` are supported only for positional and EDI files.

Common use

The `unreadblock` function is often used in the extended rule for document extraction maps.

If the tag that is being used within the `if` statement is the header tag, then an `unreadblock` is performed so that the header record will remain with the remainder of the unprocessed document. A `break` is issued after the `unreadblock` to exit the `while` loop, and the `writeblock` comes after the `IF` statement so that the header record is not written out.

If the tag that is being used within the `if` statement is a trailer tag, then an `unreadblock` is not used for the extended rule. Otherwise, the trailer record would be included as the first record in the next document to be processed.

Syntax

```
unreadblock();
```

Example

```
String[1024] buffer;

readblock(buffer);
writeblock(buffer);

while readblock(buffer) do
begin
  if left(buffer,3) = "HDR" then
  begin
    unreadblock();
    break;
  end
  writeblock(buffer);
end
//Read a block from the input file and place it in buffer. Look for
//a "HDR" record tag. If found, reset the file pointer to where it was
//before the "HDR" record was found. Write contents of the buffer to
//the output file.
```

update

The `update` function allows information in the database tables to be updated. This function is similar to the `Update` standard rule, except that it provides more flexibility.

Only the tables and fields available in the `Update` standard rule are available for the `update` extended rule. In the command syntax expression can be a string field,

string variable, or string literal. It is important to note that the table and field names for the update extended rule are slightly different than those depicted in the standard rule.

Table 35. Support for the update function

If the map/form is of type ...	Then the update function is ...
Export	Only valid on the input side of the map.
Import	Valid on the input or output side of the map.
Break (Interchange, Group, or Transaction Set)	Only valid on the input side of the map.
Build (Interchange, Group, or Transaction Set)	Only valid on the input side of the map.
Print	Not supported.
Screen entry	Not supported.

The update function also enables you to update process data with a string, instead of using the messagebox function.

Note: For the Transaction Register, the updates do not go directly to the database; they are kept in memory until the eventual select, and then they are checked against the database and inserted if necessary.

Common use

The update function is often used as an extended rule instead of a standard rule when you only want to run it based on other criteria. For example, if you want to update a value to Process Data if a quantity is over 100:

```
If #QTY > 100 then
  Update ProcessData set XPathResult = "LARGE ORDER" where XPath = "MSG";
```

It is also commonly written as an extended rule, when there are multiple update statements to be performed, since the standard rule only allows one per field.

Syntax

Syntax 1

```
update tablename set fieldname = expression [fieldname = expression] where
  key = expression [and key = expression];
```

Note: If you are updating multiple fields, each field = expression term should be separated by a comma.

Syntax 2

Updating process data with a string

```
update ProcessData set XPathResult = <some string>
  where Xpath = <location in process data>;
```

Examples

Syntax 1

```
update processdata set xpathresult="hello world" where xpath="example";
```

Example 2

Updating process data with a string

```
update ProcessData set XPathResult = #Sender
where XPath = "SenderID";
```

Additional Information

Also see “select and update Options” on page 156 for information about database tables and the associated field names that are available when using the select and update extended rules.

while ... do

The while ... do keywords run a statement repeatedly until the specified termination condition evaluates to zero. The system tests the terminating condition before each iteration of the loop, so a while loop executes zero or more times depending on the value of the termination expression.

The begin/end keywords enclose a group of statements that form the body of a while/do loop. You can use the begin/end keywords to run one or more statements conditionally. If you include more than one statement in the body of a while/do loop, you must surround the statements with the begin/end keywords. If you use only a single statement, you can omit the begin/end.

Note: Do not end conditions with a semicolon (;) – this terminating syntax is necessary for statements only.

Common use

The while/do function is often used to initialize arrays, add or remove characters from a string, and to loop through iterations of a repeating structure. For more in-depth examples, see the "Using While Do Loops In Extended Rules" white paper.

Syntax

```
while condition do
```

Example

```
integer i;

while i < 10 do
begin
    i = i + 1;
    if (i = 8) then
        continue;
    if (i = 9) then
        break;
end
//While "i" is less than ten, run the loop. If "i" is equal to or
//greater than ten, terminate the loop.
```

winexec

The winexec function enables you to execute another program while running the translator.

This program is executed asynchronously. You specify the program and determine how you want the program window displayed. You can also return an error code, if desired. If the error code is greater than 32, the program ran without errors. If the error code is less than 32, the program did not run because of an error. If the error code is "0," the system is out of memory. If the error code is "2," you didn't specify a file name. The error code is not the return value from the program you executed.

Notes:

- If you specify a program on another machine or in another domain, you must have the appropriate permission to access the specified folder.
- If translation is executing from an unattended process control command, the user ID under which that service is running must have the appropriate permission to access the specified file.

Syntax

```
winexec("program",window_display)
```

where:

- program = executable program name string (if necessary, including UNC or direct file path)
- window_display = number that indicates how you want the program window displayed (see Window display numbers, below)

Example

```
winexec("program.exe", 3)
```

```
//Exits Gentrans:Server and executes the "program.exe" program asynchronously.  
//The system displays the program window maximized (3).
```

Window display numbers

The window_display numbers that control the appearance of the program window are as follows (you must use the number to indicate how you want the program window displayed, not the window_display value):

Table 36. Window display numbers

Number	Window_Display	Definition
0	SW_HIDE	Hides the window and activates another window.
1	SW_SHOWNORMAL	Activates and displays a window. If the window is minimized or maximized, it is restored to the original size and position. This flag should be specified when displaying a window for the first time.
1	SW_NORMAL	Activates and displays a window in the original size and position.
2	SW_SHOWMINIMIZED	Activates the window and displays it as a minimized window.
3	SW_SHOWMAXIMIZED	Activates the window and displays it as a maximized window.
3	SW_MAXIMIZE	Maximizes the window.

Table 36. Window display numbers (continued)

Number	Window_Display	Definition
4	SW_SHOWNOACTIVATE	Displays the window in its most recent size and position, but does not activate it (the current active window remains active).
5	SW_SHOW	Activates the window and displays it in its current size and position.
6	SW_MINIMIZE	Minimizes the window.
7	SW_SHOWMINNOACTIVE	Displays the window as a minimized window without activating it (the current active window remains active).
8	SW_SHOWNA	Displays the window in its most recent size and position without activating it (the current active window remains active).
9	SW_RESTORE	Activates and displays the window. If the window was minimized or maximized, it is restored to its original size and position.
10	SW_SHOWDEFAULT	Activates the window and allows Microsoft Windows to determine the size and position.

writeblock

The writeblock function writes the data contains in the argument of a string variable to the output file.

The readblock and writeblock functions are used in conjunction with each other to pass a block of data from the input file to the output file without compliance checking or testing for proper EDI syntax. Together these functions provide a more efficient alternative of using "wildcard" segments, which are typically implemented in build and break maps.

The readblock and writeblock functions are also used in conjunction with the Document Extraction service, to specify the beginning and end of each document in a batch of documents, so that each document can be extracted individually.

The readblock, writeblock, and unreadblock functions are supported only for positional and EDI files.

See the "XML Build and Break Maps" appendix in the *IBM Sterling Gentran:Server for Microsoft Windows XML User Guide* for special considerations when using this function with XML data.

Common use

The writeblock function is primarily used for document extraction maps as well as build and break maps.

Syntax

```
writeblock(string_variable);
```

Examples

Example 1

```
String[1024] buffer;

readblock(buffer);
writeblock(buffer);

while readblock(buffer) do
begin
    if left(buffer,3) = "HDR" then
        begin
            unreadblock();
            break;
        end
        writeblock(buffer);
    end
//Read a block from the input file and place it in buffer. Look for
//a "HDR" record tag. If found, reset the file pointer to where it was
//before the "HDR" record was found. Write contents of the buffer to
//the output file.
```

Example 1

```
string[250] buffer;
string[3] match;
integer match_len;
// set these next two variables
match = "SUM"; // the tag of the last record in the document
match_len = 3; // the length of the tag
// read the block we're on and write it
readblock(buffer);
writeblock(buffer);
// keep reading and writing records until the end of the document
while readblock(buffer) do
begin
    writeblock(buffer);
    if left(buffer, match_len) = match then
        begin
            break;
        end
    end
end
```

writebytes

The writebytes function writes a specified number of bytes to the output file. This function is used in conjunction with the readbytes function. Used together, the readbytes and writebytes function provide an efficient method of passing data through a map if the data does not need to be compliance checked or altered in any way.

Note: The system does not append a segment terminator to the end of the string value written to the output file.

Writebytes is similar to the writeblock function, but writeblock only works with entire blocks (for example, an entire segment or record), and writebytes works with any quantity of data, whether it is smaller or larger than a block.

The writebytes function uses two parameters, first the string variable containing the data to be written to the output file, and second the number of bytes to write.

Syntax

```
writebytes(write_to_buffer, num_bytes);
```

where:

- write_to_buffer = string variable containing the data to be written to the output file
- num_bytes = integer value representing the number of bytes to read from the write_to_buffer variable and write to the output file

Note: If hex values are used in constants, use \0x instead of ^.

Example

```
string [1024] temp_buffer;  
while readbytes(temp_buffer,1024) do  
begin  
  writebytes(temp_buffer,1024);  
End  
writebytes("^0D^0A",2);  
  
//Read 1024 bytes from input file and place in string variable Snamed temp_buffer.  
//Appends a CRLF after the while loop finishes executing.
```

select and update Options

This topic contains the database table names and the associated field names that are available when using the select and update extended rules. Additional keys are also available for certain tables. Where applicable, a description of the key follows the field name.

Notes

When you use extended rules to reference a Sterling Gentran:Server database table, the syntax used is different from the actual table name. The appropriate referencing syntax is outlined for each table in this section.

For example, to refer to the Document_tb in an extended rule, you would reference Document.

See “select” on page 146 and “update” on page 150 for more information about the functions.

Division

These are the field names that are available when using the select or update extended rules to reference the division partner on the Partner Table (Partner_tb).

Refer to this table as Division.

- PARTNERNAME
- EDICODE
- APPLICATIONPARTNERKEY

DivisionLocation

These are the field names that are available when using the select or update extended rules to reference a division location on the Location Table (Location_tb).

Refer to this table as DivisionLocation.

- CONTACTNAME
- NAME
- ADDRESS1
- ADDRESS2
- ADDRESS3
- CITY
- STATE
- ZIP
- COUNTRY
- TELEPHONE
- PRIMARYREFCODE
- SECONDARYREFCODE
- FAX

DivisionLookup

These are the field names that are available when using the select or update extended rules to reference a division lookup on the Lookup Table (Lookup_tb).

Refer to this table as DivisionLookup.

- ITEM
- DESCRIPTION
- TEXT1
- TEXT2
- TEXT3
- TEXT4

This is an additional key that is available for the DivisionLookup table.

- TABLENAME

DivisionXref

These are the field names that are available when using the select or update extended rules to reference a division cross-reference on the Cross-reference Table (CrossReference_tb).

Refer to this table as DivisionXref.

- MYITEM
- PARTNERITEM
- DESCRIPTION
- TEXT1
- TEXT2
- TEXT3
- TEXT4

This is an additional key that is available for the DivisionXref table.

- TABLENAME

Document

These are the field names that are available when using the select or update extended rules on the Document Table (Document_tb).

This sub-table can only be accessed from the following map types; Transaction Break/Build, Import, Export, Print, Screen, Group Build, and Interchange Build.

The information that is accessed will be the information of the last document that was processed at the table level accessed by the rules. This must be taken into consideration when using rules that access these tables.

Refer to this table as Document.

- TESTMODE
- AGENCY
- VERSION
- TRANSACTIONSETID
- RELEASE
- DOCUMENTTYPE
- REFERENCEDATA
- DOCUMENTNAME
- APPFIELD1
- APPFIELD2
- APPFIELD3
- APPFIELD4
- APPFIELD5
- APPFIELD6
- DOCUMENTPARTNERKEY
- CONTROLNUMBER
- PARTNERKEY
- TABLEKEY (Select only)

GenericEnvelopeSegment

These are the field names that are available when using the select extended rule on the Generic Envelope Segment Table (GenericEnvelopeSegment_tb).

Refer to this table as GenericEnvelopeSegment.

- CONTROLNUMBER
- SUBCOUNT
- FIELD1
- FIELD2
- FIELD3
- FIELD4
- FIELD5
- FIELD6
- FIELD7
- FIELD8
- FIELD9

- FIELD10
- FIELD11
- FIELD12
- FIELD13
- FIELD14
- FIELD15
- FIELD16
- FIELD17
- FIELD18
- FIELD19
- FIELD20
- FIELD21
- FIELD22
- FIELD23
- FIELD24
- FIELD25
- FIELD26
- FIELD27
- FIELD28
- FIELD29
- FIELD30
- FIELD31
- FIELD32
- FIELD33
- FIELD34
- FIELD35
- FIELD36
- FIELD37
- FIELD38
- FIELD39
- FIELD40

Group

These are the field names that are available when using the select or update extended rules on the Group Table (Group_tb).

Can only be accessed from the following map types: Group Break or Build, Transaction Break, Export, and Interchange Build.

The information that is accessed will be the information of the last group that was processed at the table level accessed by the rules. This must be taken into consideration when using rules that access these tables.

Refer to this table as Group.

- TESTMODE
- CONTROLNUMBER
- FUNCTIONALGROUPID

- VERSION
- APPFIELD1
- APPFIELD2
- APPFIELD3
- APPFIELD4
- APPFIELD5
- APPFIELD6
- AGENCY
- TABLEKEY (Select only)

Interchange

These are the field names that are available when using the select or update extended rules on the Interchange Table (Interchange_tb).

This sub-table can only be accessed from the following map types: Interchange Break or Build, Group Break, Transaction Break, and Export.

The information that is accessed will be the information of the last interchange that was processed at the table level accessed by the rules. This must be taken into consideration when using rules that access these tables.

Refer to this table as Interchange.

- TESTMODE
- CONTROLNUMBER
- VERSION
- APPFIELD1
- APPFIELD2
- APPFIELD3
- APPFIELD4
- APPFIELD5
- APPFIELD6
- AGENCY
- PARTNERKEY
- TABLEKEY (Select only)

Partner

These are the field names that are available when using the select or update extended rules for a non-division partner on the Partner Table (Partner_tb).

Refer to this table as Partner.

- PARTNERNAME
- EDICODE
- APPLICATIONPARTNERKEY

These are the additional keys that are available for the Partner table.

- PARTNERKEY
- ALTERNATEKEY (refers to Application Key)

- APPLICATIONPARTNERKEY (refers to Application Key)

PartnerLocation

These are the field names that are available when using the select or update extended rules for a non-division location on the Location Table (Location_tb).

Refer to this table as PartnerLocation.

- CONTACTNAME
- NAME
- ADDRESS1
- ADDRESS2
- ADDRESS3
- CITY
- STATE
- ZIP
- COUNTRY
- TELEPHONE
- PRIMARYREFCODE
- SECONDARYREFCODE
- FAX

PartnerLookup

These are the field names that are available when using the select or update extended rules for a non-division lookup on the Lookup Table (Lookup_tb).

Refer to this table as PartnerLookup.

- ITEM
- DESCRIPTION
- TEXT1
- TEXT2
- TEXT3
- TEXT4

This is an additional key that is available for the PartnerLookup table.

- TABLENAME

PartnerXref

These are the field names that are available when using the select or update extended rules for a non-division cross-reference on the Cross-reference Table (CrossReference_tb).

Refer to this table as PartnerXref.

- MYITEM
- PARTNERITEM
- DESCRIPTION
- TEXT1
- TEXT2

- TEXT3
- TEXT4

This is an additional key that is available for the PartnerXref table.

- TABLENAME

Chapter 7. User Exits

ActiveX and User Exit Functions

createobject function

The createobject function enables you to create an instance of an ActiveX Automation Server.

```
object = createobject("ProgID");
```

ProgID is the programmatic identifier. An example of a ProgID is:
"InternetExplorer.Application"

deleteobject function

The deleteobject function enables you to delete an instance of an ActiveX Automation Server. An object must be deleted before the end of the map that uses it. It is more efficient to delete the object immediately on completion (using the DELETEOBJECT command), although the Sterling Gentran:Server translator will delete the object automatically at the end of the map. Also, if you assign one object to another one, both copies of the object must be deleted for that object to be properly unloaded.

```
deleteobject(object);
```

getiid function

The getiid function enables you to obtain the unique identifier for an interface, by using the string-character name of the interface to return the globally unique identifier that is used by software to run the interface.

```
string_variable = getiid("ProgID", "InterfaceID");
```

InterfaceID (IID) is the interface identifier. An example of an IID is:
"IWebBrowser2"

queryobject function

The queryobject function is used to request a different interface on an existing object.

```
object2 = queryobject(object1, "{IID}");
```

Note: You may not use extended rules that compare two ActiveX properties or method results, or any combination of properties or method results. This type of comparison is invalid because property and method types are unknown prior to compilation and thus it is not possible to generate the correct comparison code.

About ActiveX Technology

Sterling Gentran:Server supports additional extended rule functionality to allow you to use Microsoft's ActiveX Data Objects (ADO) from within extended rules, as well as providing enhancements to user exit support.

The information in the following topics assumes that you:

- Are familiar with the use of ActiveX Automation Servers and languages such as Visual Basic.
- Know the Translator Programming Language (TPL) constructs for creating, manipulating, and deleting ActiveX objects.
- Know the ProgID of the ActiveX object and all its exposed interfaces.
- Understand how and when extended rules are invoked and their scope.

User exits are an advanced feature of Sterling Gentran:Server that should only be used with the above prerequisites.

Restrictions

Sterling Gentran:Server extended rules selectively support ActiveX technology.

Sterling Gentran:Server extended rules support:

- ActiveX Automation Servers
- Some ActiveX Controls, but only those that work as Automation Servers

Sterling Gentran:Server does not support:

- ActiveX controls that are not ActiveX Automation Servers. Such ActiveX controls must be hosted in a graphical user interface (GUI) display, and therefore cannot be used from extended rules
- ActiveX Arrays (for example, the VT_ARRAY data type modifier)
- References (for example, the VT_BYREF data type modifier) except for output parameters in method calls. In this instance, references are only valid for the duration of the method call.
- Sterling Gentran:Server does not read registry entries or type libraries when compiling extended rules to verify the accuracy of programmatic identifiers (ProgIDs) or interfaces.
- Extended rules that compare two ActiveX properties or method results, or any combination of properties or method results. This type of comparison is invalid because property and method types are unknown prior to compilation and thus it is not possible to generate the correct comparison code.

Definition of ActiveX Terminology

ActiveX is a term that encompasses a set of rules that define how applications should share information. ActiveX grew out of technologies developed by Microsoft, specifically Object Linking and Embedding (OLE) and Component Object Model (COM).

An ActiveX automation server is an ActiveX component (a .DLL or .EXE program) that can expose part of its functionality, specifically properties and methods, via the IDispatch interface to another program on the system.

Some ActiveX controls function as automation servers.

The IDispatch interface is a standard COM interface. Automation Servers typically expose their methods and properties through this interface.

A method is an action or function that is performed by an object (for example, a calculation or a search).

A property is a characteristic or parameter of an object (for example, type, size, or creation date).

ActiveX controls are a specifically defined method of implementing ActiveX technologies. Basically an ActiveX control is a software component that executes common tasks and can integrate into the user interface of an application that provides the necessary ActiveX control host functionality.

The ActiveX control specification enables you to build component software that interacts with your application and Sterling Gentran:Server. ActiveX controls require a user interface, so they are not appropriate for translation user exits. These controls can be developed in a variety of programming languages, including Visual Basic.

About User Exits

User exits allow you to enhance your functionality or fulfill specific requirements that Sterling Gentran:Server does not perform during normal translation.

Sterling Gentran:Server supports user exits in all types of translation objects through extended rules that enable the use of Microsoft's ActiveX technology. This feature allows you to use extended rules to invoke custom functionality that was created as an ActiveX Automation Server.

Specifically, you can:

- create objects
- delete objects
- query objects
- access properties
- invoke methods

Note: You can create your own custom functionality as an ActiveX Automation Server, or you can use third party Automation Servers.

Sterling Gentran:Server supports ActiveX indexed properties, specifically these types of index:

- variant
- numeric
- string

The index support allows you to use these syntaxes (if they are supported by your Automation server):

```
n = ob.property[1];  
n = ob.property["Count"];  
n = ob.property[ob.method()];
```

Sterling Gentran:Server also supports chaining of ActiveX method calls and properties, which simplifies extended rules for Automation servers with moderately complex object models (for example, ADO).

The following is an example of chaining statements:

```
recordset.fields.item["MessageId"].value
```

User exits may be used in the following manner:

- To access your database table to perform cross-reference or lookups instead of using the Sterling Gentran:Server tables.
- To perform complex pricing calculations (for example, involving multiple customers, where they are located, and where the product is sold).

Data types supported

Table 37. Extended rule data types and the corresponding ActiveX data types

Extended Rule Data Type	ActiveX Data Type
INTEGER	VT_14
REAL	VT_R8
DATETIME	VT_DATE
STRING	VT_BSTR
OBJECT	VT_DISPATCH

Data type conversion

The Sterling Gentran:Server translator automatically converts the extended rule data type to the ActiveX data type, whenever possible. If the conversion cannot be performed, a type mismatch error is written to the Audit Log and the extended rule is immediately terminated. An error is also written to the translator report, if one is used by the operation. Since an export operation does not generate a translator report, any conversion errors during export translation are only written to the Audit Log.

Objects

ActiveX automation servers use the datatype OBJECT, which consists of two elements: properties (a defined set of data) and methods. A method is a function that is defined by the interface of the object.

All objects that are automation servers provide the default interface IDispatch, which exposes the internal functions of an application to Sterling Gentran:Server. You can also expose other interfaces. For example, Internet Explorer exposes the interface IWebBrowser2.

Syntax

The object must be declared and then created via the CREATEOBJECT command, by pointing to the object's IDispatch interface. Then, the object's methods may be invoked using the syntax `object.method(parameters)`.

The user may specify the ProgID and an optional InterfaceID (IID). If you just specify the ProgID, Sterling Gentran:Server uses the default IDispatch interface. The IID is a machine-readable identifier that uniquely identifies the interface of an object (for example, {00020300-00B000-C00004005300}). Interfaces can also be identified by their human-readable name (for example, IDispatch).

Note: Each IID must be enclosed in braces {}.

Location

The location of the user exit in the map depends on what the user exit needs to do and what you want the scope of the rule to be. Typically, you might:

- declare an object in the On Begin rule of the Input side of the map (Loop Level Extended Rules dialog box of Input positional file),
- create and use the object in the On Begin rule of the Input side of the map (Loop Level Extended Rules dialog box of Input positional file), and
- delete the object in the On End rule of the Output side (Loop Level Extended Rules dialog box of Output EDI file).

Note: The sample above is intended as an example only. You can declare, create, use, and delete an object in an extended rule for a map component at any hierarchical level, including field level extended rules, if appropriate.

See *How to Define Extended Rules* for more information on creating extended rules for various map components.

Examples of User Exits

Defining object variable

This example defines a variable that represents an ActiveX Automation Server:
object ob;

Creating a default interface

This example creates an instance of the default interface of the Internet Explorer ActiveX Automation Server:

```
object ob;  
ob = createobject("InternetExplorer.Application");  
//Creates an instance of the default interface of an ActiveX Automation Server.
```

Creating a specific interface

This example creates an instance of the Internet Explorer ActiveX Automation Server and requests a specific interface:

```
object ob;  
ob = createobject("InternetExplorer.Application",  
    "{EAB22AC1-30C1-11CR-A7EB-000C05BAE0B}");  
//Creates an instance of a specific interface of an ActiveX Automation Server.  
//In this case, the IID is known (specified in braces).  
//Note: The createobject command is more efficient if you use the IID  
//instead of the interface name.
```

This example creates an instance of the Internet Explorer ActiveX Automation Server, where the interface identifier of the desired interface is unknown, but the name of the interface is IWebBrowser:

```
object ob;  
string[50] iid;  
iid = getiid("InternetExplorer.Application", "IWebBrowser2");  
ob = createobject("InternetExplorer.Application", iid);  
//Creates an instance of a specific interface of an ActiveX Automation Server.  
//In this case, the IID is not known, so the ProgID and name of the interface  
//(IWebBrowser) are specified and Gentrans:Server looks up the IID. The IID is  
//loaded into the string variable "iid" and then that value is used to create  
//the object.
```

Deleting an object

An object must be deleted before the end of the map that uses it. It is more efficient to delete the object immediately when you no longer need it, although the Sterling Gentran:Server translator deletes the object automatically at the end of the map. Also, if you assign one object to another one, both copies of the object must be deleted for that object to be properly unloaded.

This example deletes the object "ob" that was used in the previous examples:

```
deleteobject(ob);
```

Getting a property value

This example obtains a property value from an ActiveX Automation Server:

```
object ob;
string[50] iid;
iid = getiid("InternetExplorer.Application", "IWebBrowser2");
ob = createobject("InternetExplorer.Application", iid);
IF ob.Visible = 1 THEN
BEGIN
//The property value is accessed.
END
//Test to see if the system displays the Internet Explorer.
```

Setting a property value

This example sets a property value in an ActiveX Automation Server:

```
object ob;
string[50] iid;
iid = getiid("InternetExplorer.Application", "IWebBrowser2");
ob = createobject("InternetExplorer.Application", iid);
ob.Visible = 1;
//The property value is set. Display Internet Explorer on the desktop.
```

Passing parameters to a method

To pass parameters to a method, use the syntax `objectname.methodname(parameters)`. The Sterling Gentran:Server translator automatically converts the extended rule data type of the property to the ActiveX data type, whenever possible. If there is no directly corresponding type in the extended rules, the conversion is performed as well as possible by the operating system.

Sterling Gentran:Server relies on default data types. For example, Sterling Gentran:Server converts VT_CURRENCY (which is an unknown data type to the translator programming language) to a real or numeric data type. If the conversion cannot be performed, a type mismatch error is written to the Translator Report and the extended rule is immediately terminated.

This example navigates to a web page by passing the URL to the Navigate method of Internet Explorer:

```
object ob;
string[50] iid;
iid = getiid("InternetExplorer.Application", "IWebBrowser2");
ob = createobject("InternetExplorer.Application", iid);
ob.Navigate("www.sterlingcommerce.com");
//Navigates to the IBM web page by passing the URL to the Navigate
//method of Internet Explorer. Note: URL must be enclosed in quotation marks.
```


Returning values in output parameters

To return values in output parameters, use the syntax
`objectname.methodname(InputParameter, OUT OutputParameter)`.

Note:

- Input and output parameters may occur anywhere in the parameter list.
- Output parameters must be preceded by the OUT keyword.
- When an extended rule variable is used as an output parameter, the ActiveX datatype must match exactly.

See Data types supported for a list of ActiveX data types and the analogous extended rule data type for each.

This example decrypts data in a field:

```
object ob;  
ob = createobject("YourCompany.DecryptionUserExit");  
ob.Decrypt("EncryptionKey", OUT #Field_Name);
```

Testing successful object creation

This example tests the value of an object against zero to determine if it was correctly created:

```
object ob;  
ob = createobject("YourCompany.DecryptionUserExit");  
IF ob = 0 THEN  
BEGIN  
//Object not created properly; perform task X.  
END  
//Test to see if the object was created successful. If it was not  
//created, perform the specified task.
```

Obtaining another interface of an existing object

This example uses the IID of an object to obtain a different interface of an existing object (object1):

```
object ob, ob2;  
ob = createobject("InternetExplorer.Application");  
ob2 = queryobject(ob, "{EAB22AC1-30C1-11CF-A7EB-0000C05BAE0B}");
```

If the IID of the desired interface is unknown, use the `getiid` function to determine the correct IID and then use the `queryobject` function, as this example demonstrates:

```
object ob, ob2;  
string[50] iid;  
ob = createobject("InternetExplorer.Application");  
iid = getiid("InternetExplorer.Application", "IWebBrowser2");  
ob2 = queryobject(ob, iid);
```

Accessing a database

This example uses a user exit to cross-reference a UPC code with a value in a database to return an SKU number. This example adds the user exit to an invoice (import) map (refer to the Sterling Gentran:Server tutorial folder, PET_810.MAP for an example). The user exit code is implemented by increments in the map, with each section added to the logical map component.

On Begin Extended Rule

Type the following code into the On Begin extended rule of the INPUT positional file:

```
//Declare the object
object ob;
//Create an instance of the Visual Basic Active X Automation Server
ob = CreateObject("SCUPCSKU.SKUResolve");
//To create an instance of the C++ ActiveX Automation Server
//use the following command instead:
//ob = CreateObject("UpcSku.Application");

if ob = 0 then
begin
MessageBox("Create Object failed",0);
end
```

INVDETAIL.UPCCODE Field Extended Rule

Type the following code into the UPCCODE field extended rule on the Input side of the map (INVDETAIL record):

```
string [100] msg;
msg = "UPC Code = ";
concat(msg,#UPCCODE,12);
//Call the automation server
#UPCCODE = ob.ResolveSKU(#UPCCODE);
concat(msg, ", SKU Code = ",13);
concat(msg,#UPCCODE,len(#UPCCODE));
//Display message box to user listing UPC code and SKU matches
MessageBox(msg,0);
```

On End Extended Rule

Type the following code into the On End extended rule of the INPUT positional file:

```
//Delete the object
deleteobject(ob);
```

Note: See Examples of Automation Servers for sample automation servers (written in Visual Basic and C++) that access a Microsoft Access Database to cross-reference a UPC code with an SKU code.

Examples of Automation Servers

This topic contains sample automation servers that access a Microsoft Access Database to cross-reference a UPC code with an SKU code. The first example is written in the Visual Basic programming language and the second one in C++.

Notes

Both of the following automation servers could be used with the "Accessing a database" user exit extended rules on Accessing a database.

Visual Basic Automation Server

This sample automation server was written in the Visual Basic programming language. It accesses a Microsoft Access Database to cross-reference a UPC code with an SKU code. The syntax is: ResolveSKU(UPCCode as String). Note that after

compiling the Automation Server (SCUPCSKU.EXE), it was registered using REGSRV32.EXE because it was not self-registering.

```
//This is a Visual Basic Active X EXE project. All code shown here resides in
//the SKUResolve class. (Class 1 was renamed to SKUResolve). To access the
//database, the following reference was included in the project: Microsoft
//DAO 2.5/3.5Compatibility Library.
//The following line forces the compiler to make sure all variables are declared
//prior to use within the program.
Option Explicit
//This function receives a UPC code as it's only parameter and cross-references it
//with a value in the database to get the corresponding SKU code.
Public Function ResolveSKU(UPCCode As String) As String
    Dim DB As Database, RS As Recordset
    Set DB = DBEngine.Workspaces(0).OpenDatabase("c:\GENSRVNT utorial\upcsku.mdb")
    Set RS = DB.OpenRecordset("UPCSKU", dbOpenDynaset)
    //Make sure there are no trailing spaces on the parameter passed to this function.
    UPCCode = Trim(UPCCode)
    //Since the database is case-sensitive make sure that the parameter is all
    //capital letters.
    UPCCode = UCase(UPCCode)
    RS.MoveFirst
    //Loop until we find the UPC code or until the end of the recordset is found.
    While Not RS.EOF
        //If we find the UPC code, return the 'SKU code associated with it.
        If RS("UPCCode") = UPCCode Then
            ResolveSKU = RS("SKUCode")
            Exit Function
        End If
        RS.MoveNext
    Wend
    //No matching UPC code was found for the parameter passed to the function.
    //Return a value stating that there was no SKU found.
    ResolveSKU = "No Record Found"
End Function
```

C++ Automation Server

This sample automation server was written in the C++ programming language. It performs exactly the same function as the previous Visual Basic sample user exit—accessing a Microsoft Access Database to cross-reference a UPC code with an SKU code. Note that after compiling the Automation Server (UPCSKU.DLL), it was registered using REGSRV32.EXE because it was not self-registering.

```
// This is a MFC AppWizard (dll) project with the Automation option selected.
//All code shown here resides in the CUpcSkuMain class, which is derived from
//CCmdTarget. When the class was created in ClassWizard, the ProgID
//"UpcSku.Application" was specified in the "Createable by type ID:" field.
// This ProgID is different than the one used for the VB example so that both
// Automation servers can co-exist in the registry.

//To access the database, a class (CUpcSkuDaoRecordset) derived from CDaoRecordset
//is needed. DAO and Snapshot options were selected and the location of the database
// (Upcsku.mdb) was specified during the creation of the class in ClassWizard.
//Include the header file for the derived recordset class.
#include "UpcSkuDaoRecordset.h"
// This function is called when the last reference for this automation object
//is released.

void CUpcSkuMain::OnFinalRelease()
{
    CCmdTarget::OnFinalRelease();

    // Since we used DAO to access the database,
    // terminate DAO now so the DLL can unload.
```

```

        if (AfxOleCanExitApp())
            AfxDaoTerm();
    }
    // This function receives a UPC code as it's only parameter and cross-references it
    //with a value in the database to get the corresponding SKU code.

BSTR CUpcSkuMain::ResolveSKU(LPCTSTR UPCCode)
{
    CString strResult;
    CString sUPCCode = UPCCode;

    CDaoDatabase DB;
    DB.Open("C:\\GENSRVNT\\Tutorial\\Upcsku.mdb", FALSE, TRUE);

    CUpcSkuDaoRecordset RS(&DB); RS.Open();

    // Make sure there are no trailing spaces on the parameter passed to this function.
    sUPCCode.TrimRight();
    // Since the database is case-sensitive make sure that the
    //parameter has all capitalized letters in it.

    sUPCCode.MakeUpper();
    RS.MoveFirst();

    // Loop until we find the UPC code or until the end of the recordset is found.
    while ( !RS.IsEOF() )
    {
        // If we find the UPC code, return the SKU code associated with it.

        if ( RS.m_UPCCode == sUPCCode )
        {
            strResult = RS.m_SKUCode;
            RS.Close();
            DB.Close();
            return strResult.AllocSysString();
        }
        RS.MoveNext();
    }

    // No matching UPC code was found for the parameter passed to the function.
    //Return a value stating that there was no SKU found.

    strResult = "No Record Found";

    RS.Close();
    DB.Close();

    return strResult.AllocSysString();
}

```

Creating a User Exit

Before you begin

The ActiveX Automation Server must be installed on the same machine as the Sterling Gentran:Server translator (where you are using the translation object that contains the user exit).

About this task

Use this procedure to create a user exit.

Procedure

1. Start the Sterling Gentran:Server Application Integration subsystem.
2. Open the map in which you want to apply a user exit.
3. Determine where the user exit will be located in the map (for example, session rule or extended rule).

Note: When you apply a user exit to a map component, subordinate map components may also be able to execute the same user exit.

4. Construct the user exit.
 - See How to Define Extended Rules for more information on creating extended rules for various map components.
 - See the topics in the Alphabetic Language Reference section for more information about the createobject, deleteobject, and getiid commands.

Chapter 8. GentranEx.DLL

About GentranEx.DLL

GentranEx.DLL is an in-process automation server that extends the flexibility of the Sterling Gentran:Server MAPPER.EXE (Application Integration) and TX32.EXE (translator) programs by using COM technology and the IDispatch interface.

GentranEx.DLL enables you to perform several functions that exceed the scope of normal Sterling Gentran:Server operation, including:

- accessing any GentranDatabase table
- gathering information from a translation session and writing that information to a log file
- extending the functionality of Application Integration extended rules

GentranEx.DLL is automatically installed during the Sterling Gentran:Server installation process.

The information in the following topics assumes that you are familiar with the following:

- Sterling Gentran:Server
- the Application Integration subsystem
- your operating system
- automation servers
- how and when extended rules are invoked and their scope
- how to create extended rules

About Database Access

GentranEx.DLL allow access to any GentranDatabase table through the TSql statements UPDATES and INSERTS. The GentranEx.DLL uses ODBC and the Sterling Gentran:Server ODBC DSN (typically GentranDatabase) to connect to the database.

Note: The TSql statement must be generated by the user and passed to the Execute method. To support a connection to an Oracle database the TSql statements must enclose Tablenames and Columnames in square brackets. These square brackets are automatically converted to double quotes before the statement is executed.

Implementing Database Access

Prog ID

"GentranEx.DispatchDBAccess"

Interface ID

{860AAE85-7DA8-11D2-ABDB-00C04FF3971C}

Table 38. Access Methods

Method	Description
short ExecuteSQL(BSTR IpszSQLString)	Returns "1" for success and "-1" for failure. Errors are written to the Application Event Viewer.
BSTR GetLastError()	Returns a string containing the error of the most recent attempt to invoke the ExecuteSQL method. If the last attempt was successful this call returns a null string.
BSTR GetLastSQL()	Returns a string containing the most recently executed SQL statement using the Δ ExecuteSQL (...) function.

Properties: None.

Extended rule example

This is an example of an extended rule using GentranEx.DLL to access the GentranDatabase.

```
Object obDBAccess;
Integer result;
String [64] szPartnerKEY;
String [1024] szMsg;

obDBAccess = CreateObject ("GentranEx.DispatchDBAccess");
szTSQL = "Update [interchange_tb] Set [AppField6]=""+"
#STATUS+
"" where [PartnerKEY]=""+"
szPartnerKEY+
"" and [Direction]=1 and [ControlNumber]=""+"#REF+"";
result = obDBAccess.ExecuteSQL(szTSQL);
if result = -1 then
    Begin
        szMsg = obDBAccess.GetLastError();
        messagebox(szMsg,0);
        szMsg = obDBAccess.GetLastSQL();
        messagebox(szMsg,0);
    End

deleteobject(obDBAccess);
```

About Debugging

GentranEx.DLL enables you to gather information from any translation session executed on the machine on which GentranEx.DLL is installed, and write that information to a log file. This function does not require any interaction from the user and thus is an efficient alternative to using the MessageBox function when debugging map.

Each entry made to the log file contains the following information, which provides clarity and readability to the log:

- date and time stamp
- name of the log owner
- source of the message
- the message

The name of the log owner, the source of the message, and the message are all supplied by the user via the `OpenLog()` and `WriteLog()` functions.

Implementing Debugging

Prog ID

"GentranEx.TraceLog"

Interface ID

{7DA17F77-8090-11D2-ABE4-00C04FF3971C}

Table 39. Debugging Methods

Method	Description
short <code>OpenLog(BSTR PathFilename, BSTR Owner, short WriteMode)</code>	<ul style="list-style-type: none"> • Returns "1" for success, "-1" for failure, and "-2" if the file is already open. • Argument 1 is the path and filename of the log file to be created. • Argument 2 is the owner of the log. This value is written to the log file and is included to make the log easier to read. • Argument 3 is the writemode (for example, 1 = append (this is the default) and 2 = truncate). • The "Owner" column in the generated log file is 20 characters wide — therefore Argument 2 is truncated at that value (20) when written to the log file.
short <code>Open()</code>	Opens a trace log file (named "GentranTrace.log") in the GENSrvNT home directory. The file is opened in Append mode and is assigned an Owner name of "GentranEx".
short <code>WriteLog(BSTR Source, BSTR Message)</code>	<ul style="list-style-type: none"> • Returns "1" for success and "-1" for failure. • Argument 1 is the source of the message that is written. • Argument 2 is the message text. Both Arguments 1 and 2 are written in separate columns in the log file for clarity. • The "Source" column in the log file is 20 characters wide and thus Argument 1 is truncated at that value (20). • The "Message" column in the log file is 256 characters wide and thus Argument 2 of this function is truncated at that value when written to the log file.
BSTR <code>GetLastError()</code>	Returns a string containing the error of the most recent attempt to open the log file.
short <code>CloseLog()</code>	<p>Closes the log file and returns "1" for success or "-1" for failure.</p> <p>Note: The file should always be closed before your trace object goes out of scope.</p>

Properties: None.

Extended rule example

This is an example of an extended rule using GentranEx.DLL for debugging.

```

Object obTrace;
Integer result;
String [32] szPartnerKEY;

obTrace = CreateObject ("GentranEx.TraceLog");
result = obTrace.OpenLog("D:\GENSRVNT\TraceTest.txt","ICLANA",2);
if result = -1 then
    Begin
        obTrace.WriteLog("Start of Log", "Trace Test");
        obTrace.WriteLog("Converted TUN#",szPartnerKEY);
    End
obTrace.CloseLog();
deleteobject(obTrace);

```

About Rules Extensions

GentranEx.DLL provides methods and properties that extend the functionality of the Application Integration extended rules logic. For example, there are methods that allow the conversion of strings from uppercase to lowercase, which allows greater control over string comparison logic. Also, there are functions available that return the current date and/or time. GentranEx.DLL also provides a property that can be referenced to determine whether the automation server is running in a non-interactive environment.

Implementing Rules Extensions

Prog ID

"GentranEx.Rules"

Interface ID

{387C0F34-82F5-11D2-ABEA-00C04FF3971C}

Table 40. Rules Extension Methods

Method	Description
DATE GetDate()	Returns the current date. The assignment operator (=) can be used to set any date/time variable or date/time type field equal to the value returned by this method.
DATE GetTime()	Returns the current time. The assignment operator (=) can be used to set any date/time variable or date/time type field equal to the value returned by this method.
BSTR ToUpper(BSTR szMixedCase)	<ul style="list-style-type: none"> • Uses the string to be converted as its argument. • Returns the original string converted to uppercase. Any string variable, string field, or string literal can be supplied as the first argument.
BSTR ToLower(BSTR szMixedCase)	<ul style="list-style-type: none"> • Uses the string to be converted as its argument. • Returns the original string converted to lowercase. Any string variable, string field, or string literal can be supplied as the first argument.

Properties

```
short InteractiveState
```

Extended rule example

This is an example of an extended rule using GentranEx.DLL to exceed normal extended rule logic.

```
Object obRules;
String [32] szLowerCaseStr;
String [32] szUpperCaseStr;
String [32] szResultString;
date d;

//Initialize values of string variables
szLowerCaseStr = "abcdefg";
szUpperCaseStr = "WXYZ";

//Create the Rules object
obRules = CreateObject("GentranEx.Rules");

//Test string conversion methods and display results
szResultString = obRules.ToUpper(szLowerCaseStr);
messagebox(szResultString,0);
szResultString = obRules.ToLower(szUpperCaseStr);
messagebox(szResultString,0);

//Test date methods
d = obRules.GetDate();
#TimeField = obRules.GetTime();

//Test whether running interactively
if obRules.InteractiveState = 1 then
    messagebox("We're running on the desktop",0);
else
    Begin
        //use the TraceLog object to write to a log file since we're
        //running non-interactively
    End

deleteobject(obRules);
```

Chapter 9. Translator Command Line Interface

Command Line Syntax

Note:

- If you do not specify the XML format parameter, the report is generated in binary form by default.
- This interface is currently used by all applications that invoke the services of the TX32.EXE translator.

The syntax of the command line is:

```
tx32 [-n] [-q]
      [-a server]
      [-e eventid]
      [-i ipckey [hwnd] ]
      [-f infile templatefile outfile report file]
      [-u param -u param ...]
      [-t tracefile]
      [-x XMLformattedreport]
```

Syntax parameters

This table describes the command line syntax parameters: .

Table 41. Syntax parameters

Parameter	Description	
-n	Do not move document to InDrawer after print or export	
-q	Quiet mode (does not display message boxes or status dialog box)	
-a	Server (name of audit server to which TX32.EXE connects)	
-e	EventID (identifier of an audit event to which this translator belongs)	
-i	Inter-process communication (IPC) mode	
	Subparameter	Description
	ipckey	Key to ipcfile (translator will use the IpcMsg folder and add .IPC)
	hwnd	Number representation of the parent applications's main window handle
-f	Simple-translator mode	
	Subparameter	Description
	infile	Path to file to be translated
	templatefile	Path to translation object to be used
	outfile	Path to file to write translated output
	reportfile	Path to create translator report
-u	User-defined parameters accessible from extended rules	
	Subparameter	Description
	param	Parameter value

Table 41. Syntax parameters (continued)

Parameter	Description	
-t	Trace mode	
	Subparameter	Description
	tracefile	Path to trace the file to be used
-x	XMLformattedreport	<p>Generates the translator report in XML format</p> <p>Notes:</p> <ul style="list-style-type: none"> • If you do not specify this parameter, the report is generated in binary form by default. • Translator reports in XML format are not supported with maps that have CII or CII Positional specified as either the input or output format. This parameter will be ignored if specified with CII or CII Positional maps, and the translator report will be generated in binary form.

Chapter 10. Error Messages

About Error Messages

The informational messages are dependent on the context of the program and are intended to be self-explanatory.

The Sterling Gentran:Server error messages and other informational messages are displayed in one of the following dialog boxes.

- Compile Errors dialog box when you compile a translation object
- Error section of the Extended Rules dialog boxes when you compile an extended rule containing errors before compiling the translation object and when you commit an erroneous action in Sterling Gentran:Server

The error messages are described in the following topics, along with the actions you can take to correct the problems.

Compile Error Messages

The Compile Error Messages are displayed in the Compile Errors dialog box if you compile a translation object with errors. Error messages are also displayed in the Error section of an Extended Rule dialog box if you compile the extended rule containing errors before compiling the translation object. After you correct the cause of the errors, click Compile again to ensure that the rule is error-free.

Messages

The compile error messages are listed by the four- or five-digit message number and the error message text. The error definitions contain the actions that you can take to correct the problem (if appropriate) and a description that includes possible causes of the error.

Table 42. Compile Error Messages

Msg ID	Message Text	Explanation	Your Action
1000	expected ' '	The rule does not have the required "." between a group name and a field name.	Insert a "." between the group name and field name.
1001	no statement to compile	The rule is does not contain any statements.	Add a statement or statements to the rule.
1002	unexpected end of program	The rule was not complete.	Finish the rule.
1003	expected ' , '	The rule does not have the required "," between parameters.	Insert a "," between parameters.
1004	expected ' ; '	A statement was not terminated properly.	Terminate the statement in an appropriate manner.
1006	no statements to compile	The body of an IF/ELSE or WHILE condition was empty.	Complete the body of the unfinished condition.
1007	syntax error ...	The map component contains a syntax error.	Correct the syntax error.

Table 42. Compile Error Messages (continued)

Msg ID	Message Text	Explanation	Your Action
1008	expected '#'	The rule does not have the required '#' before a field name.	Insert a '#' before the field name.
2000	group ... undefined	The rule references a group that does not exist.	Change the reference to an existing group or delete the reference.
2001	... is not a member of ...	The rule references a field that does not belong to the specified group.	Change the reference to an existing field in the specified group.
2002	insufficient indices to access group ...	The rule does not give the full addressing for a group.	Complete the addressing for the group.
2003	too many indices to access group...	The rule uses too many addresses for the group.	Address the group correctly.
2004	... has not been defined	The rule references an undefined variable.	Define the variable in the declarations section.
2005	out of temporary variables	The rule could not be compiled because some expressions are too complex.	Simplify the expressions and compile the rule again.
2008	field type unknown	The compiler was unable to determine the type of a field.	Verify that a Data Type is selected for this field.
2009	cannot reference a local field with no current group	The rule (probably pre- or post-session) references a local field, but is not associated with any group.	Reference the field using the proper addressing.
2010	instances of '%1' are transient and cannot be accessed	The rule references an output group using full addressing. This form of addressing is only appropriate for input groups.	Reference the output group with the proper address.
2011	no field specified	The rule omits a field reference.	Add the field reference to the rule.
2012	group ... is promoted and cannot be accessed in this manner	You attempted to access a promoted group in a COUNT or DELETE rule.	Do not COUNT or DELETE a promoted group.
2100	... is not an array variable	The rule uses array indexing for a variable that is not an array.	Use the proper indexing for the variable.
2101	...: array index required	The rule uses an array variable without using the necessary array indexing.	Add the necessary indexing to the array variable.
2102	...: array overflow	The rule uses an invalid array index.	Use the proper array index for the array variable.
2103	only one wildcard index is permitted	The rule uses more than one wildcard index.	Only one wildcard index is permitted per rule.
2104	a wildcard index must be specified	The rule does not specify a wildcard index when required.	Add a wildcard index where necessary.
2200	expected a string	A required string or string variable is not supplied.	Add the required string or string variable.
2201	string overflow	A string overflow occurred.	Verify that the size of a target string is equal to or greater than the source string.
3000	array size expected in declaration of ...	Invalid array declaration.	
3001	declaration of ... missing ']	Invalid array declaration.	
3002	string size expected	Invalid string declaration.	

Table 42. Compile Error Messages (continued)

Msg ID	Message Text	Explanation	Your Action
3003	string size missing ']'	Invalid string declaration.	
3004	variable name expected	Invalid variable declaration.	
3005	... already defined	Two variables with the same name are defined at the same scope.	Rename one of the two variables.
3006	expected an accumulator number, found ...	Invalid accumulator reference.	
3007	... is not a valid accumulator number	Invalid accumulator reference.	
4000	expected a numeric expression, found ...	You specified something other than the expected numeric expression.	Specify the correct numeric expression.
4001	expected a term, found ...	You specified something other than the expected term.	Specify the correct term.
4002	expected '+' or '-'	You specified something other than the expected "+" or "-".	Specify "+" or "-".
4003	expected '*' or '/'	You specified something other than the expected "*" or "/".	Specify "*" or "/".
4004	expected ')'	You specified something other than the expected ")".	Specify ")".
4005	expected a factor, found ...	Invalid numeric expression.	
4006	expected '('	You specified something other than the expected "(".	Specify "(".
4007	... is of incorrect type	The specified expression is of an incorrect type.	Specify the correct type for the expression.
4008	expected a relational operator	You specified something other than the expected relational operator.	Specify the correct relational operator.
4009	missing argument	A required parameter was omitted.	Add the required parameter.
4010	assignment expected	The assignment operator was omitted from an assignment statement.	Add the correct assignment operator to the statement.
4011	operator ... requires two arguments	Only one parameter was supplied for a binary operator.	Supply a second parameter for the binary operator.
4012	converting real to integer may lose significant digits	You are converting a real number to an integer.	Verify that losing decimal places is acceptable in this instance.
4013	expression too complex, use sub-expressions	A mathematical expression contains too many terms.	Group some of the terms of the mathematical expression in parentheses.
4014	expected an integer, found ...	The compiler expected to find an integer instead of [...].	Supply the correct value for [...].
4015	no valid condition specified	You did not specify a valid IF or WHILE condition.	Supply a valid condition in the IF or WHILE statement.
4100	expected a date, found ...	A date is required but not supplied.	Specify a date.
4101	expected a date modifier, found ...	A date modifier was required but not supplied.	Specify a date modifier.
4102	... is not a date	Invalid date expression.	
5000	THEN expected	The compiler expected a THEN condition.	

Table 42. Compile Error Messages (continued)

Msg ID	Message Text	Explanation	Your Action
5001	DO expected	The compiler expected a DO condition.	
5002	END expected	The compiler expected a END condition.	
5004	FROM expected	The compiler expected a FROM condition.	
5005	INTO expected	The compiler expected a INTO condition.	
5006	END without BEGIN	An END statement was found without a corresponding BEGIN statement.	Insert a BEGIN statement in the correct location.
5016	misplaced 'BREAK'	The compiler found BREAK outside of a loop.	Remove the BREAK or place it inside a loop.
5017	misplaced 'CONTINUE'	The compiler found CONTINUE outside of a loop.	Remove the CONTINUE or place it inside a loop.
5018	too many parameters	Too many parameters were supplied for a function.	Remove the unnecessary parameters.
5019	too few parameters	Too few parameters were supplied for a function.	Add the necessary parameters.
6000	expected a database table name	The compiler expected a database table name.	Insert a database table name in the necessary location.
6001	expected a database column name	The compiler expected a database column name.	Insert a database column name in the necessary location.
6002	too many column receivers specified	The number of columns specified did not match the number of receivers in a SELECT statement.	Correlate the number of columns specified in the SELECT statement to the number of receivers.
6003	too few column receivers specified	The number of columns specified did not match the number of receivers in a SELECT statement.	Correlate the number of columns specified in the SELECT statement to the number of receivers.
6004	WHERE expected	The compiler expected a WHERE statement.	
6005	expected a database key	An invalid database key was specified in a WHERE statement.	Specify the correct database key in the WHERE statement.
6006	WHERE not allowed with this database table	A WHERE statement is not necessary to access the specified database table.	Remove the WHERE statement.
6007	invalid key combination	The combination of keys specified in the WHERE statement is not a valid unique compound key to the table.	Specify a valid combination of keys in the WHERE statement.
7000	... is not a valid seek type	You used an invalid seek type in FSEEK.	Use BEGIN, END, or CURRENT.
20001	Record ..., Field ... : date field missing date format	The specified field does not have a date format.	Edit the field and choose a date format.
20003	Field ... : constant used in standard rule does not exist	The standard rule for the specified field uses an invalid constant.	Correct the standard rule or create the constant.
20004	Field ... : code list used in standard rule does not exist	The standard rule for the specified field uses an invalid code list.	Correct the standard rule or create the code list.

Table 42. Compile Error Messages (continued)

Msg ID	Message Text	Explanation	Your Action
20005	Field ... : the qualifier field specified in a use constant standard rule is invalid	The standard rule for the specified field uses an invalid qualifier.	Correct the standard rule.
20006	Field ... : the field specified to store the code description in a use code standard rule is invalid	The standard rule for the specified field designates an invalid field for the description.	Correct the standard rule.
20007	Record ... : the specified key field ... : uses an undefined constant	The key field for the specified record uses an invalid constant.	Correct the key field.
20008	Record ... : the specified key field ... : uses an undefined code list	The key field for the specified record uses an invalid code list.	Correct the key field.
20010	Record ... : the specified key field ... : is inactive	The key field for the specified record is inactive.	Activate the key field.
20700	only one binary data and one binary length field are permitted	You have more than one binary data and/or more than one binary length element in one segment.	Remove the additional binary data and/or binary length elements from the segment.
20701	binary length must precede binary data	The binary length element must be sequenced before the binary data element in the segment so the translator knows how much data to expect.	Move the binary length element to before the binary data element.
20702	incomplete binary data	You marked a segment as binary, but did not include either a binary length element or a binary data element (or both).	Add a binary length element and/or a binary data element to the segment.
20703	group ... has no active child objects	You tried to compile the translation object, but the specified group is empty.	Activate at least one child object in the group.
20704	Element ..., Attribute ...: enumerated attribute declared without accompanying standard rule	The specified XML attribute is configured to use an enumeration but there is no code list standard rule that defines the allowed values.	Define a use code list standard rule for the specified attribute.
20705	Element ..., Attribute ...: code list used in enumerated attribute does not exist.	The code list for the specified enumerated XML attribute does not exist.	Define the code list.
20706	Element ..., Attribute ...: default value is not valid ...	The specified default value is not in the code list for this XML attribute.	Ensure that the specified default value is in the code list.
20707	Codelist ..., Attribute ...: value used in enumerated attribute code list does not match XML NMTOKEN production	A value in the enumeration code list is not valid for XML.	Verify that all the value specified in the code list are valid (legal) for an XML attribute.
20708	Entity ... : Entity value is invalid	The value of the specified entity is not valid (legal) in XML.	Correct the entity value.
20709	... : Illegal use of reference character	An entity reference was not terminated properly.	Correct the entity reference.

Table 42. Compile Error Messages (continued)

Msg ID	Message Text	Explanation	Your Action
20710	... : Malformed character reference encountered	An XML character reference was not terminated properly.	Correct the character reference.
20711	... : Invalid character referenced	An XML character reference is invalid.	Correct the character reference.
20712	... : Reference to undefined entity encountered	The referenced entity is not defined.	Define the correct entity.
20713	... : Circular entity references encountered	An entity references an entity that in turn references the original entity.	Do not reference the original entity in a circular manner.
20714	...: default does not match the attribute type	The default value is the wrong type for the attribute.	Either correct the type of the attribute or the default value.
20715	...: Contains illegal character ('<')	The default value contains the illegal character '<'.	Correct the default value.

Sterling Gentran:Server Error Messages

The Sterling Gentran:Server error messages are displayed when you commit an erroneous action. When the system displays an error message, you must acknowledge the message by clicking OK and then taking the appropriate action.

Messages

The error messages are listed alphabetically below by the first letter of the error message text. The error definitions contain the actions that you can take to correct the problem (if appropriate) and a description that includes possible causes of the error.

Table 43. Sterling Gentran:Server Error Messages

Message Text	Explanation
A code list entry must have a code value	You attempted to add a code without an associated code value.
A code list must have an element id	You attempted to create a code list without an element ID.
A code list for this element already exists. You must use another element id or delete the original code list first	You attempted to create a code list with the same element ID as an already existing code list.
A condition field is required	You established a conditional relationship that requires a condition field, but did not specify a condition field.
A data type is required.	You did not specify the Data Type of a field.
A field must have a name that is unique within its parent group	You tried to give a field a name that is already in use by another field within the same group (not necessarily the same record).
A group must have a unique name	You tried to give a group a name that is already used by another group or record.
A group or record with a maximum usage of 1 cannot be split or promoted	You attempted to split or promote a single-occurrence group or record. This is not a valid action.

Table 43. Sterling Gentran:Server Error Messages (continued)

Message Text	Explanation
A key field has been selected but no key value has been specified.	You established a key field without specifying a key value.
A Memory Exception has occurred.	A system error occurred.
A name must be entered	You did not specify a name for an object.
A problem occurred while attempting to close loop This is probably due to incorrect standards.	An error occurred when reading from the standards.
A record must have a unique name	You tried to give a record a name that is already used by another group or record.
A Resource Exception has occurred.	A system error occurred.
A serious error was encountered whilst accessing the clipboard and the action was abandoned	The system aborted a cut, copy, or paste operation because a serious error occurred.
A system error was encountered while compiling the translation object	While compiling a translation object, a system-related problem, such as lack of memory, prevented the compile from completing.
A TDF could not be generated due to a lack of available memory. Either restart Microsoft Windows or close down other applications, then try again	While generating a TDF, the system ran out of memory.
An accumulator must be chosen for this entry	You did not specify the accumulator to be used in a Use Accum standard rule.
An invalid usage count has been entered	You entered an invalid usage count for a record or group.
No actions have been chosen for this accumulator entry	You selected an accumulator but did not specify any actions for it in a Use Accum standard rule.
No alternate accumulator has been selected	In a Use Accum standard rule, you attempted to create an accumulator entry that used an alternate accumulator, but did not specify which alternate accumulator should be used.
No character ranges have been specified for this token.	You did not specify the characters allowed for a syntax token.
No fields have been used in this relationship	You established a conditional relationship but did not specify the fields involved.
No more accumulator entries can be created	You attempted to create more than six accumulator entries in a Use Accum standard rule.
No more field mappings can be created	You attempted to create more than eight field mappings for a Select standard rule.
The constant ID must be unique.	You entered a literal constant identifier in the ID field that is already used in an existing constant.

Table 43. Sterling Gentran:Server Error Messages (continued)

Message Text	Explanation
The contents of the clipboard cannot be pasted into the translation object at this point	Either the Clipboard does not contain copied or cut data, or you are trying to paste XML information into a positional file format or paste position information into an XML file format.
The file format cannot be deleted	You tried to delete a file format object.
The Maximum Usage must be greater than zero and not less than the Minimum Usage.	You entered an invalid maximum usage count.
The number of entries to be split must be greater than zero and less than the maximum number of entries in the original	You entered an invalid number in the Split dialog box. The number to be split must be greater than zero and less than the maximum number of entries in the original.
The translation object could not be compiled	The translation object is not ready to be compiled.
The token code must be unique.	You attempted to create a syntax token with the same token identifier as an existing syntax token.
These fields cannot be linked because the input field is deeper than the output field. The translator would be unable to address the input field.	You are attempting to create an invalid link. A valid link involves an Input field and an Output field that are at the same level.
This code already exists. You must use another code or delete the original code first	You attempted to add a code to a code list that already contained that code.
This field cannot be linked to another field	Due to an unknown error, Sterling Gentran:Server was unable to begin creating the link.
This file is not a valid Gentran translation object	You attempted to open (load) a file that is not a translation object.
You have entered an invalid character or character code in Record Delimiter 1	You entered an invalid Record Delimiter 1 on the Positional File Format Properties dialog box.
You have entered an invalid character or character code in Record Delimiter 2	You entered an invalid Record Delimiter 2 on the Positional File Format Properties dialog box.
You have entered an invalid decimal separator or character code.	You entered an invalid decimal separator.
You have entered an invalid element delimiter character or character code	You entered an invalid element delimiter.
You have entered an invalid Pad character or character code	You entered an invalid pad character for a positional field.
You have entered an invalid release character or character code	You entered an invalid release character.
You have entered an invalid sub-element delimiter character or character code	You entered an invalid sub-element delimiter.
You have entered an invalid tag delimiter character or character code	You entered an invalid tag delimiter.

Table 43. Sterling Gentran:Server Error Messages (continued)

Message Text	Explanation
You must enter a description of the translation object.	You attempted to save the translation object without entering the translation object description on the Translation Object Details dialog box.
You must enter a subtable name.	For a Select or Update standard rule, the selected table requires you to specify a table name in the Sub Table field.
You must enter a valid character range.	You entered an invalid character range for a syntax token.
You must enter a valid syntax token.	You did not enter a valid token identifier in the Token field when creating or editing a syntax token.
You must enter a value for this constant.	You did not specify a value for the constant in the Value field when creating or editing a syntax token.
You must enter an ID.	You did not specify a literal constant identifier for the constant in the ID field when creating or editing a syntax token.
You must enter the name of the author of this translation object	You attempted to save the translation object without entering the name of the translation object author on the Translation Object Details dialog box.
You must select a constant type.	You did not select a constant type from the Type list when creating or editing a constant.
You need to specify Input and Output File Types for the new translation object	You clicked OK on the Create New Translation Object dialog box without selecting both the Input Format Type and Output Format Type.

About Translator Report Error Messages

The Document and Interchange Translator Report error messages are displayed on the Document Translator Report and the Interchange Translator Report under the Message Number and Message columns.

The Message Number column on the translator report contains a prefix (INF, EDI, or POS), a dash ("-"), and a four digit number that identifies the error. The prefixes are described in the following table

Prefix	Description
INF	Used only with information messages, which are not defined in this chapter because they are intended to be self-explanatory.
EDI	Used with all the messages listed below that are not informational. It is used if the error is related to an EDI file.
POS	Used with all the messages listed below that are not informational. It is used if the error is related to a positional flat file.

The Message column on the translator report contains the actual error message text.

Translator Error Messages

The translator report error messages are listed below by the last three digits of the message number and the error message text.

Table 44. Translator Error Messages

Msg ID	Message Text	Explanation	Your Action
100	Mandatory Element Missing	An element that the translation object designated as "Mandatory" was not created in an Outbound document or was not received in an Inbound document.	<p>Use the Segment/Record ID, Sequence, and Element fields on the Translator Report to determine which mandatory element in the document is missing.</p> <p>Outbound</p> <p>If the document was entered using the Document Editor, open the document and complete the missing field. If you imported the document into your system, delete the document and then import that document after the import file has been corrected. See the <i>IBM Sterling Gentran:Server for Microsoft Windows User Guide</i> for more information.</p> <p>Inbound</p> <p>Contact your trading partner and determine what action you should take.</p>
110	Incorrect Element Format	An element was entered or received with an incorrect format. Some examples of incorrect format are: a numeric field that contains non-numeric characters, and a field that exceeds the maximum length or is less than the minimum length (as defined in the standard) and invalid dates.	<p>Use the Segment/Record ID, Sequence, and Element fields on the Translator Report to determine which element in the document is invalid.</p> <p>Outbound</p> <p>Correct the data source.</p> <p>Inbound</p> <p>Contact your trading partner and determine what action you should take.</p>
120	Too Many Components in Composite	A composite element in a document you received has more component elements (sub-elements) than allowed by the standard.	<p>Use the Segment/Record ID, Sequence, and Element fields on the Translator Report to determine which element in the document is invalid.</p> <p>Outbound</p> <p>If the document was entered using the Document Editor, open the document and correct the invalid field. If you imported the document into your system, delete the document, correct the data, and then import that document again. See the <i>IBM Sterling Gentran:Server for Microsoft Windows User Guide</i> for more information.</p> <p>Inbound</p> <p>Contact your trading partner and determine what action you should take.</p>

Table 44. Translator Error Messages (continued)

Msg ID	Message Text	Explanation	Your Action
130	Invalid Conditional Relationship	A conditional relationship in the document is not valid.	<p>Use the translator report to determine where in the document the error occurred.</p> <p>Outbound</p> <p>If the document was entered using the Document Editor, open the document and correct the conditional relationship. If you imported the document into your system, delete the document and then import that document again.</p> <p>Inbound</p> <p>Contact your trading partner and determine what action you should take.</p>
140	Implicit Rule Failure	A validation rule set up against this field failed in the translator. Typically, this occurs when the Exclusive flag is set for a standard rule, and the field value does not match the data table.	<p>Check the data value that you received against the valid data that is allowed for the field.</p>
200	Mandatory Component Missing	A component (sub-element) of a composite element that the translation object designated as "Mandatory" was not created in an Outbound document or not received in an Inbound document.	<p>Use the Segment/Record ID, Sequence, Element, and Composite fields on the Translator Report to determine which mandatory component in the document is missing.</p> <p>Outbound</p> <p>If the document was entered using the Document Editor, open the document and complete the missing field. If you imported the document into your system, delete the document and then import that document again.</p> <p>Inbound</p> <p>Contact your trading partner and determine what action you should take.</p>

Table 44. Translator Error Messages (continued)

Msg ID	Message Text	Explanation	Your Action
210	Incorrect Component Format	A component (sub-element) of a composite element that the translation object designated as "Mandatory" was entered (Outbound) or received (Inbound) with an incorrect format. Some examples of incorrect format are: a numeric field that contains non-numeric characters, and a field that exceeds the maximum length or is less than the minimum length (as defined in the standard).	<p>Use the Segment/Record ID, Sequence, Element, and Composite fields on the Translator Report to determine which element in the document is invalid.</p> <p>Outbound</p> <p>If the document was entered using the Document Editor, open the document and correct the invalid field. If you imported the document into your system, delete the document, correct the data, and then import that document again.</p> <p>Inbound</p> <p>Contact your trading partner and determine what action you should take.</p>
220	Component Delimiter	A component delimiter was encountered instead of the expected element.	Contact either/or your trading partner or the translation object creator and determine what action you should take.
300	Mandatory Segment	<p>A segment that the translation object designated as "Mandatory" was not created in an Outbound document or was not received in an Inbound document.</p> <p>This error can be generated in a variety of circumstances. The most common is that the input data sequence does not correspond to the data sequence defined in the translation object used to translate the data. If this is the case, the information provided with the message may indicate a segment in the data.</p>	<p>Use the Segment/Record ID field on the Translator Report to determine which mandatory segment in the document is missing.</p> <p>Outbound</p> <p>If the document was entered using the Document Editor, open the document and key data into the fields that are necessary to generate the segment. If you imported the document into your system, delete the document, add the data that is necessary to generate the segment, and then import that document again.</p> <p>Inbound</p> <p>Contact your trading partner and determine what action you should take.</p>
310	Invalid Loop Start/End Structure	An invalid Loop Start/Loop End was found in an Inbound document.	Use the information in the translator report to determine which LS/LE pairing is invalid. Contact your trading partner and determine what action you should take.
315	Invalid Segment or Record Structure	A segment (in an EDI file) or a record (in a positional flat file) in an inbound file did not match what the translation object was expecting.	From viewing the information in the translator report and viewing the "Raw EDI" interchange, determine which segment or record is invalid. Contact your trading partner to determine what action you should take. See the <i>IBM Sterling Gentran:Server for Microsoft Windows User Guide</i> for more information.

Table 44. Translator Error Messages (continued)

Msg ID	Message Text	Explanation	Your Action
405	Unknown Partner	An Interchange was received but the system cannot determine which partner sent it.	From viewing the information in the translator report and viewing the "Raw EDI" interchange, determine which partner sent you the interchange. If the partner is not listed on your system, create the partner and a relationship, and attach the interchange to that partner. If the partner already exists on your system, attach the interchange to that partner and then determine why the system did not automatically identify the partner.
410	Header/Trailer Control Numbers do not match	The control numbers on the header and trailer do not match, as specified by the standard.	Check the "Raw EDI" view to determine which control numbers are in the EDI file, and then contact your trading partner to determine what action you should take. See the <i>IBM Sterling Gentran:Server for Microsoft Windows User Guide</i> for more information.
415	Control Total Incorrect	The EDI control total in the Segment Identified field of the translator report does not equal the value that was calculated by the Compliance Checker.	Check the "Raw EDI" view to determine what the control total should be, and then contact your trading partner to determine what action you should take. See the <i>IBM Sterling Gentran:Server for Microsoft Windows User Guide</i> for more information.
420	Unknown Relationship	A document was received but the Partner Profile for that partner does not include a corresponding Inbound Relationship.	From the viewing the information in the translator report and viewing the "Raw EDI" interchange, determine which relationship the document requires and create that inbound relationship for the partner. See the <i>IBM Sterling Gentran:Server for Microsoft Windows User Guide</i> for more information.
460	Invalid Test Mode Flag	The partner relationship was found but the test mode flag in the data did not match the test mode defined in the inbound partner relationship.	Change the test mode of the inbound partner relationship to match the test mode of the data. See the <i>IBM Sterling Gentran:Server for Microsoft Windows User Guide</i> for more information. Then, ask your trading partner to change the test mode of the data they are sending to match that defined by the inbound relationship.

User Level Error Messages

User-level type errors can occur when extended rules are used in the map.

The user-level error messages are listed below in numerical order based on their Message ID. The table also includes an explanation of the error and an action you can take to correct the error.

Table 45. User-level Error Messages

Msg ID	Message Text	Explanation	Your Action
800	Batch File Header Section Missing	The system cannot find a header section record within the batch file.	Contact your Trading Partner.
801	Unexpected Header Segment ID	A value other than "00" is in the header section record segment Identifier field (701).	Contact your Trading Partner.

Table 45. User-level Error Messages (continued)

Msg ID	Message Text	Explanation	Your Action
802	Unexpected Transmission Type	A transmission value other than T (Transaction), R (Response), or E (Error) is in the transmission type (880-K6) field of the header section record.	Contact your Trading Partner.
803	Sender ID Missing	The system cannot find a value in the sender ID (880-K1) field in the Header Section Record.	Contact your Trading Partner.
804	Invalid Batch Number	The batch number field (806-5C) contains a format other than the YYDDD format, where: <ul style="list-style-type: none"> • YY=Year • DDD=Julian date 	Contact your Trading Partner.
805	Unexpected File Type	A value other than P (Production) or T (Test) is in the File Type field (702).	Contact your Trading Partner.
806	Receiver ID Missing	The system cannot find a value in the receiver ID field (880-K7) of the header section.	Contact your Trading Partner.
807	Batch File Detail Data Record Missing	The system cannot find a Detail Data Record within the batch file.	Contact your Trading Partner.
808	Unexpected Data Segment ID	A value other than "G1" is in the segment identifier field of the Detail Data Record (701).	Contact your Trading Partner.
809	Transaction Reference Number Missing	The system cannot find a value in the transaction reference number field of the Detail Data Record (880-K5).	Contact your Trading Partner.
810	Batch File Trailer Record Missing	The system cannot find the trailer record within the batch file.	Contact your Trading Partner.
811	Invalid Trailer Segment ID	A value other than "99" is in the segment identifier field (701) of the trailer record.	Contact your Trading Partner.
812	Unexpected Batch Number	The batch number field (806-5C) of the trailer record does not match the batch number field (806-5C) of the header record.	Contact your Trading Partner.
813	Invalid Record Count	The record count field (751) of the trailer record contains an invalid number.	Contact your Trading Partner.
814	Processed DDRs Mismatch Record Count	The actual count of Detail Data Records is different from the value in the record count field (751) of the trailer record.	Contact your Trading Partner.
815	Duplicate Segment Within Transaction	A segment occurs more than one time within the transaction.	Contact your Trading Partner.

Table 45. User-level Error Messages (continued)

Msg ID	Message Text	Explanation	Your Action
816	Unexpected Transaction Code	<p>A value other than a value listed below is in the transaction code field (103-A3) of the request transaction header segment or response header segment.</p> <ul style="list-style-type: none"> • E1 = Eligibility Verification • B1 = Billing • B2 = Reversal • P1 = Re-bill • P2 = P.A. Request & Billing • P3 = P.A. Inquiry • P4 = P.A. Request Only • N1 = Information Reporting • N2 = Information Reporting Reversal • N3 = Information Reporting Re-bill • C1 = Controlled Substance Reporting • C2 = Controlled Substance Reporting Reversal • C3 = Controlled Substance Reporting Re-bill 	Contact your Trading Partner.
817	Invalid Transaction Count	<p>A value other than a value listed below is in the transaction count field (109-A9) of the request transaction header segment or the response transaction header segment.</p> <ul style="list-style-type: none"> • Blank = not specified • 1 = One Occurrence (Default) • 2 = Two Occurrence (Except Transaction Code E and P) • 3 = Three Occurrence (Except Transaction Code E and P) • 4 = Four Occurrence (Except Transaction Code E and P) 	Contact your Trading Partner.
818	Processed Transactions Mismatch Transaction Count	The actual count of the transactions is different from the value in the transaction count field (109-A9) of the response header segment.	Contact your Trading Partner.
819	Unexpected Header Response Status Code	A value other than A (Accepted) or R (Rejected) is in the header response status field (501-F1) of the response header segment.	Contact your Trading Partner.

Table 45. User-level Error Messages (continued)

Msg ID	Message Text	Explanation	Your Action
820	Unexpected Transaction Response Status Code	<p>A value other than a value listed below is in the transaction response status field (112-AN).</p> <ul style="list-style-type: none"> • A = Approved • C = Captured • D = Duplicate of Paid • F= PA Deferred • P = Paid • Q = Duplicate of Capture • R = Rejected • S = Duplicate of Approved 	Contact your Trading Partner.

Core Translator Error Messages

Core translator type errors can occur when you are receiving or sending data.

The core translator error messages are listed below in numerical order based on their Message ID. The table also includes an explanation of the error and an action you can take to correct the error.

Table 46. Core Translator Error Messages

Msg ID	Message Text	Explanation	Your Action
830	Null Character	The data file contains binary data.	Contact your Trading Partner.
833	Unexpected End of File	A field of a specified length is incomplete.	Contact your Trading Partner.
835	Unexpected Start of Text	The record does not contain End of Text character.	Contact your Trading Partner.
836	Unexpected End of Text	The record does not contain Start of Text character.	Contact your Trading Partner.
837	Missing Stream Separator1	The system is reading non-streamed data with a template from a map where "stream segments" box is unchecked.	Contact your Trading Partner.
838	Missing Stream Separator2	The system reading non-streamed data with a template from a map where "stream segments" box is unchecked.	Contact your Trading Partner.
839	Unexpected Separator	A group separator, segment separator, field separator, STX, or ETX is present in the data.	Contact your Trading Partner.
840	Unexpected Field Separator	Based on the map, two field separators occur consecutively or a different separator is expected.	Contact your Trading Partner.
841	Unexpected Group Separator	Based on the map, a group separator occurs when none was specified.	Contact your Trading Partner.
842	Missing Group Separator	Based on the map, a group separator is missing.	Contact your Trading Partner.

Table 46. Core Translator Error Messages (continued)

Msg ID	Message Text	Explanation	Your Action
843	Unexpected Stream Separator1	Based on the map, a stream separator1 is occurs when none was specified.	Contact your Trading Partner.
845	Positional Tag Defined Beyond EOS	The position of the transaction code (Positional Record Tag) is specified to start beyond the end of the segment.	Change the value of the positional segment under the NCPDP Properties tag to start before the end of the segment.
855	Delimited Segment Has No Fields	The delimited segment must have at least the mandatory fields.	Contact your Trading Partner.
856	Missing Delimited FS After SS	The delimited field separator is not present after the segment separator.	Contact your Trading Partner.
857	Field ID Data Does Not Match Map Field ID	The field ID data is different from the defined delimited field ID tag.	Contact your Trading Partner.
858	Unexpected Char(s) in Delimited Field ID	A separator was found in the delimited field ID data.	Contact your Trading Partner.
859	Null Character in Delimited Field ID	The delimited field ID contains binary data.	Contact your Trading Partner.
860	Incomplete Delimited Field ID	The field ID data is shorter than expected.	Contact your Trading Partner.
861	Missing Delimited Field Data	No data is found after a delimited field ID.	Contact your Trading Partner.
862	Null Character in Delimited Field Data	The delimited field data contains binary data.	Contact your Trading Partner.
865	Segment ID Data Does Not Match Map Segment ID	The segment ID data is different from the defined delimited segment ID field.	Contact your Trading Partner.
866	Unexpected Char(s) in Delimited Segment ID	A separator was found in the delimited segment ID.	Contact your Trading Partner.
867	Null Character in Delimited Segment ID	The delimited segment ID contains binary data.	Contact your Trading Partner.
868	Incomplete Delimited Segment ID	The delimited segment ID is shorter than expected.	Contact your Trading Partner.
869	Field ID Size Differs From Global Field ID Size	The computed size of the segment/field ID is different from the value of the global field ID length.	Contact your Trading Partner.
885	No Data after STX in Batch Segment	The batch record does not contain any data.	Contact your Trading Partner.
886	Unexpected STX in Batch Segment	The batch record does not contain an ETX character.	Contact your Trading Partner.
887	Missing ETX in Batch Segment	The batch record does not contain an ETX character.	Contact your Trading Partner.

Core Translator Error Messages: Sending Batch Data Using the NCPDP Data Record Field

Core translator errors can also occur when you are sending batch data using the NCPDP Data Record field to your Trading Partner. Many times the error occurs because the Batch File string is missing a portion of the Batch File or because a value in the string is incorrect.

To prevent errors from occurring verify that the Batch File string follows the detail data record (DDR) format:

<separator>filename<separator>StartOffset<separator>EndOffset<separator>

The core translator error messages are listed below in numerical order based on their Message ID. The table also includes an explanation of the error and an action you can take to correct the error.

Table 47. Core Translator Error Messages: Sending Batch Data Using the NCPDP Data Record Field

Msg ID	Message Text	Explanation	Your Action
888	Could Not Open NCPDP Data File	The NCPDP data file either does not exist or it is not readable.	Verify that the file you are selecting is the correct filename or supply the information that is missing from the string field.
890	Missing Separator in DDR Info String	A separator is missing from the DDR string field.	Insert a ";" in the DDR string field.
891	Missing Filename in DDR Info String	The filename is not in the DDR string field.	Add the filename to the DDR.
892	Invalid Start byte Offset in DDR Info String	The system found non-numeric data at start of set.	Add a valid start of set (numeric) value.
893	Invalid End Byte Offset in DDR Info String	The system found non-numeric data at end of set.	Add a valid end of set (numeric) value or remove the data if data up to end of file should be processed.
894	Missing Start/End Byte Offset in DDR Info String	The Start/End Byte Offset is not in the DDR string field.	Add the Start/End Byte Offset to the DDR string field or remove extra separators.
895	Extra Data in DDR Info String	The DDR string contains extra data.	Remove the extra data from the DDR string.
896	Total Bytes Read Mismatch Total Bytes to Read	The actual number of bytes read is less than the number of bytes specified in the set.	Verify the Start/End Byte Offsets or remove the end byte offset if data up to end of file should be processed.
897	Start Byte Offset is Beyond End of File	The position of the Start Byte Offset is specified to start after the end of the file.	Change the start position of the Start Byte Offset.
898	Start Byte Offset is Beyond End Byte Offset	The position of the Start Byte Offset starts after the position of the End Byte Offset.	Change the position of the Start Byte Offset to start before the End Byte Offset, or change the position of the End Byte Offset to start after the Start Byte Offset.

Notices

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

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

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

IBM Director of Licensing

IBM Corporation

North Castle Drive

Armonk, NY 10504-1785

U.S.A.

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

Intellectual Property Licensing

Legal and Intellectual Property Law

IBM Japan Ltd.

19-21, Nihonbashi-Hakozakicho, Chuo-ku

Tokyo 103-8510, Japan

The following paragraph does not apply to the United Kingdom or any other country where such provisions are inconsistent with local law:

INTERNATIONAL BUSINESS MACHINES CORPORATION PROVIDES THIS PUBLICATION "AS IS" WITHOUT WARRANTY OF ANY KIND, EITHER EXPRESS OR IMPLIED, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. Some states do not allow disclaimer of express or implied warranties in certain transactions, therefore, this statement may not apply to you.

This information could include technical inaccuracies or typographical errors. Changes are periodically made to the information herein; these changes will be

incorporated in new editions of the publication. IBM may make improvements and/or changes in the product(s) and/or the program(s) described in this publication at any time without notice.

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

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

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

IBM Corporation

J46A/G4

555 Bailey Avenue

San Jose, CA 95141-1003

U.S.A.

Such information may be available, subject to appropriate terms and conditions, including in some cases, payment of a fee.

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

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

Information concerning non-IBM products was obtained from the suppliers of those products, their published announcements or other publicly available sources. IBM has not tested those products and cannot confirm the accuracy of performance, compatibility or any other claims related to non-IBM products. Questions on the capabilities of non-IBM products should be addressed to the suppliers of those products.

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

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

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

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

COPYRIGHT LICENSE:

This information contains sample application programs in source language, which illustrate programming techniques on various operating platforms. You may copy, modify, and distribute these sample programs in any form without payment to IBM, for the purposes of developing, using, marketing or distributing application programs conforming to the application programming interface for the operating platform for which the sample programs are written. These examples have not been thoroughly tested under all conditions. IBM, therefore, cannot guarantee or imply reliability, serviceability, or function of these programs. The sample programs are provided "AS IS", without warranty of any kind. IBM shall not be liable for any damages arising out of your use of the sample programs.

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

© IBM 2024. Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. 2024.

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

Trademarks

IBM, the IBM logo, and `ibm.com`[®] are trademarks or registered trademarks of International Business Machines Corp., registered in many jurisdictions worldwide. Other product and service names might be trademarks of IBM or other companies. A current list of IBM trademarks is available on the Web at "Copyright and trademark information" at <http://www.ibm.com/legal/copytrade.shtml>.

Adobe, the Adobe logo, PostScript, and the PostScript logo are either registered trademarks or trademarks of Adobe Systems Incorporated in the United States, and/or other countries.

IT Infrastructure Library is a registered trademark of the Central Computer and Telecommunications Agency which is now part of the Office of Government Commerce.

Intel, Intel logo, Intel Inside, Intel Inside logo, Intel Centrino, Intel Centrino logo, Celeron, Intel Xeon, Intel SpeedStep, Itanium, and Pentium are trademarks or registered trademarks of Intel Corporation or its subsidiaries in the United States and other countries.

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

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

ITIL is a registered trademark, and a registered community trademark of the Office of Government Commerce, and is registered in the U.S. Patent and Trademark Office.

UNIX is a registered trademark of The Open Group in the United States and other countries.

Java™ and all Java-based trademarks and logos are trademarks or registered trademarks of Oracle and/or its affiliates.

Cell Broadband Engine is a trademark of Sony Computer Entertainment, Inc. in the United States, other countries, or both and is used under license therefrom.

Linear Tape-Open, LTO, the LTO Logo, Ultrium and the Ultrium Logo are trademarks of HP, IBM Corp. and Quantum in the U.S. and other countries.

Connect Control Center®, Connect:Direct®, Connect:Enterprise®, Gentran®, Gentran®:Basic®, Gentran:Control®, Gentran:Director®, Gentran:Plus®, Gentran:Realtime®, Gentran:Server®, Gentran:Viewpoint®, Sterling Commerce™, Sterling Information Broker®, and Sterling Integrator® are trademarks or registered trademarks of Sterling Commerce®, Inc., an IBM Company.

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

Index

Special characters

<< operator 116

A

accumulators 75, 78, 80, 81, 82, 83, 113

activation 25, 26

ActiveX

automation server 163

controls 163

functions 163

restrictions 163

alphabetic language reference

atoi 120

atol 120

auditlog 121

begin, end 122

break 122

cerror 123

concat 127

continue 127

count 128

createobject 128

date 128

delete 130

deleteobject 130

empty 130

exec 131

exist 132

fseek 133

ftell 133

functions 132

get 134

getid 134

if, then, else 135

index 136

insert 137

left 138

len 138

messagebox 139

mid 140

ntoa 141

param 142

queryobject 143

readblock 143

readbytes 144

right 145

select 146

set 146

strdate 147

strstr 149

unreadblock 149

update 150

while, do 152

winexec 153

writeblock 154

writebytes 155

analysis

application file format 17

EDI file 17

analysis (*continued*)

overview 17

reconciling files 18

application file analysis 17

application file format 5

Application Integration overview 1

arrays 113

assignment statement 115

atoi function 120

atoi syntax 119

atol function 120

atol syntax 119

auditlog function 121

auto-increment map version

customizing 11

automation servers

C++ 170

examples 170

Visual Basic 170

B

batch data

core translator errors 200

begin, end keywords 122

binary segments 55

break keyword 122

C

C++ automation server 170

cerror function 123

CII files

modifying 44

overview 44

code item descriptions 90

code list table 84, 86, 87, 88, 89, 90

code table 84

colors

customizing 8

commands

extended rules 111

compiling

errors 183

composite

EDI file format 6

composites

properties 41

concat function 127

concat syntax 118

conditional logic

if, then, else keywords 135

while, do 152

conditions

defining 43

modifying 43

confirmation messages

customizing 14

constants

defining 73

constants (*continued*)

deleting 73

mapping 74

overview 71

using 72

continue keyword 127

copy function 28

count function 128

create sub function 28

createobject function 128, 163

creating groups or records 28

cross-reference table

select function 65

customizing the display 8

cut function 28

D

data

export 101, 102

formatting 47

importing 91

sending or receiving errors 198, 200

data formats

numbers 53

data types

string 48

user exits 165

database access 175

date and time

Use System Variable function 70

date format

options 12

setting default 12

date function 128

date/time type

using 54

datetime

expression 116

syntax 116

deactivation 26

debugging 176

decimal point

changing the default 29

use of comma 29

declarations

extended rules 105

delete function 130

deleteobject function 130, 163

delimiters

EDI delimiters 36

display options 8

Division table 156

DivisionLocation table 156

DivisionLookup table 156

DivisionXref table 156

document name 69

Document table 156

document translator

errors 191

E

- EDI associations 20
- EDI delimiters 36
- EDI file analysis 17
- EDI file format 6
- EDI files
 - activating 25, 26
 - composite properties 41
 - copy, cut, and paste 28
 - deactivating 26
 - element properties 41
 - loop segments 39, 40, 41
 - modifying group properties 37
 - modifying segment properties 38
 - overview 36
 - promoting 26
 - splitting 27
- element
 - EDI file format 6
- elements
 - composites 41
 - properties 41
- empty function 130
- equalize the map sides 11
- error messages
 - application errors 188
 - compile errors 183
 - core translator errors 198, 200
 - extended rules 195
 - overview 183
 - receiving data 198, 200
 - sending batch data 200
 - sending data 198, 200
 - Sterling Gentran:Server 188
 - translation objects 183
 - translator errors 192
 - translator report 191
 - user errors 188
 - user-level errors 195
- exec function 131
- exist function 132
- export map
 - overview 101
- export process
 - inbound process before export 101
 - overview 101
 - setting up 102
 - supplementary envelope 102
- expressions
 - datetime 116
- extended price 82, 83
- extended rules
 - ActiveX 163
 - creating a user exit 172
 - declarations and initialization 105
 - defining 109
 - errors 195
 - GentranEx.DLL 175, 176, 178
 - map component rule 110
 - overview 105
 - processing 107
 - session rules 109
 - statements 106
 - syntax 111, 112, 113
 - user exit 165

F

- field
 - application file format 5
- fields
 - creating 34
 - in a group 113
 - in internal storage 113
- file definitions 24, 25
- file properties
 - CII files 44
- fixed-format file
 - creating a group 45
- fixed-format files
 - creating a group 34
 - creating fields 34
 - creating subsequent records 30
 - using 28
- fonts
 - customizing 9
- formatting data
 - date/time type 54
 - number type 52
 - overview 47
 - string type 48
- fseek function 133
- ftell function 133
- functions
 - atoi 120
 - atol 120
 - auditlog 121
 - error 123
 - concat 127
 - count 128
 - createobject 128, 163
 - date 128
 - delete 130
 - deleteobject 130, 163
 - empty 130
 - exec 131
 - fseek 133
 - ftell 133
 - functions
 - queryobject 163
 - get 134
 - getid 134, 163
 - index 136
 - insert 137
 - left 138
 - len 138
 - messagebox 139
 - mid 140
 - ntoa 141
 - param 142
 - queryobject 143
 - readblock 143
 - readbytes 144
 - right 145
 - select 146
 - select and update options 156
 - set 146
 - strdate 147
 - strstr 149
 - unreadblock 149
 - update 150
 - winexec 153
 - writeblock 154
 - writebytes 155

G

- GenericEnvelopeSegment table 156
- GentranEx.DLL
 - examples 175, 176, 178
- get function 116, 134
- getid function 134, 163
- global display options
 - customizing 8
- group
 - application file format 5
 - creating 45
 - EDI file format 6
- Group table 156
- groups
 - creating 34
 - in internal storage 113
 - modifying 37
 - promoting 26
 - splitting 27

H

- hash totals 80
- header records 30

I

- if, then, else keywords 117, 118, 135
- import
 - process 91
 - system configuration 91
 - system import map 93
 - system import translation object 93
- import process
 - overview 91
- in-process automation servers 175
- inbound translation process 1
- index function 136
- initialization
 - extended rules 105
- insert function 28, 137
- Interchange table 156
- interchange translator
 - errors 191

K

- keywords
 - begin, end 122
 - break 122
 - continue 127
 - extended rules 111
 - if, then, else 117, 118, 135
 - while, do 152

L

- left function 138
- left syntax 118
- len function 138
- len syntax 119
- line items 78
- link display
 - customizing 10
- linking 55

- links
 - creating 55
- literal constants
 - defining 73
 - deleting 73
 - mapping 74
 - overview 71
- location tables
 - select function 67
- lookup tables
 - select function 67
- Loop Count function 75
- loop end
 - defining inbound 40
 - defining outbound 41
- loop segments 39, 40, 41
- loop start
 - defining inbound 39
 - defining outbound 40

M

- map component rules
 - extended rules 110
- mapping
 - auto-register 57
 - compile 57
 - compile using command line 58
 - description 1
 - input side 56
 - output side 56
 - overview 15
 - report 58
- maps
 - building 24
 - components 25, 26
 - creating 18
 - details 20, 24
 - equalizing sides 11
 - incrementing 11
 - version number 11
- messagebox function 139
- mid function 140
- mid syntax 118

N

- navigation 7
- ntoa function 141
- ntoa syntax 119
- number formats 53
- number types
 - using 52

O

- operators
 - extended rules 112
- outbound translation process 3

P

- param function 142
- partner definition
 - select function 64

- Partner table 156
- PartnerLocation table 156
- PartnerLookup table 156
- PartnerXref table 156
- paste function 28
- post-session rules
 - extended rules 107, 109
- pre-session rules
 - extended rules 107, 109

Q

- qualifiers
 - generating 74
- qualifying relationship 72
- quantity invoiced 81
- queryobject function 143, 163

R

- readblock function 143
- readbytes function 144
- record
 - application file format 5
- record delimiters
 - resetting 29
- records
 - creating subsequent 30
 - creating the first 30
 - header 30
 - temporary records 31, 32
- reference data 69, 70
- relational conditions
 - defining 43
 - modifying 43
- repeating elements 113
- repeating segments
 - promoting 26
 - splitting 27
- right function 145
- right syntax 118
- rules extensions 178
- running total 82, 83

S

- segment
 - EDI file format 6
- segments
 - binary 55
 - loop segments 39, 40, 41
 - modifying 38
 - promoting 26
 - splitting 27
- select function 61, 146
 - cross-reference table 65
 - examples 63
 - location tables 67
 - lookup tables 67
 - options 156
 - partner definition 64
 - table access examples 63
- Select function
 - supplementary envelope 102
- session rules
 - extended rules 109

- set function 116, 146
- simple mapping 55
- standard rules 61
 - Loop Count function 75
 - select function 61, 64, 65, 67
 - update function 68, 69, 70
 - Use Accumulator function 75, 78, 80, 81, 82, 83
 - Use Code function 84, 86, 87, 88, 89, 90
 - Use Constant function 71, 72
 - Use System Variable function 70

statements

- assignment 115
- extended rules 106
- Sterling Gentrans:Server errors 188
- strdate function 147
- strdate syntax 118
- string constants 113
- string types 48
- strstr function 149
- strstr syntax 118
- symbols
 - extended rules 113

syntax

- command line interface 181
- extended rules 111, 112, 113

syntax tokens

- character range 51
- creating for Eastern European languages 50
- creating for Western European languages 49
- deleting 51
- deleting a character range 51
- using 52

system configuration

- modifying 91
- system import map
 - alternate key 97
 - application alias value field 98
 - application ID field 98
 - compiling 99
 - creating 93
 - document 95
 - partner key 94, 97
 - release field 96
 - six-field key 94
 - standard field 95
 - test/production field 96
 - transaction set 95
 - version field 95
- system import translation object
 - compiling 99
 - creating 93

T

- temporary records 31, 32
- TFDs
 - CII files 44
 - creating the first 46
- time syntax 116
- translation objects
 - details 20, 24
 - errors 183

- translation process
 - inbound 1
 - outbound 3
- translator
 - errors 192

U

- unreadblock function 149
- update function 68, 150
 - document name 69
 - document name and reference data 69
 - options 156
 - reference data 70
- Use Accumulator function 75, 78, 80, 81, 82, 83
- Use Code function 84, 86, 87, 88, 89, 90
- Use Constant function 71, 72
 - generating qualifiers 74
 - qualifying relationship 72
- Use System Variable function
 - system date and time 70
- user exits
 - access a database 167
 - create an interface 167
 - creating 172
 - define object variable 167
 - delete an object 167
 - description 165
 - examples 167
 - exec function 131
 - get a property value 167
 - pass parameters 167
 - return parameters 167
 - set a property value 167
- user interface overview 7

V

- validating data 89
- value total 81
- variables
 - extended rules 105
- Visual Basic automation server 170

W

- while, do keywords 152
- winexec function 153
- writeblock function 154
- writebytes function 155



Product Number: 5725-D09

Printed in USA