

IBM Sterling Gentran:Server for Windows



# Forms Integration User Guide

*Version 5.3.1*



IBM Sterling Gentran:Server for Windows



# Forms Integration User Guide

*Version 5.3.1*

**Note**

Before using this information and the product it supports, read the information in "Notices" on page 169.

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

© **Copyright IBM Corporation 1996, 2012.**

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

---

# Contents

## Chapter 1. Form Integration Basics . . . . 1

About Forms Integration . . . . .	1
Forms Integration User Interface. . . . .	1
EDI File Format Window . . . . .	3
Layout Window . . . . .	5
The Form Building Process . . . . .	6

## Chapter 2. Form Design . . . . . 7

Preparation and Analysis . . . . .	7
About Global Defaults . . . . .	7
Preferences dialog box - Tree tab. . . . .	8
Preferences dialog box - Standard Formats tab . . . . .	9
Preferences dialog box - Positional Default tab. . . . .	10
Preferences dialog box - Files tab . . . . .	11
Preferences dialog box--Layout tab . . . . .	12
Preferences dialog box--Confirmations tab . . . . .	13
Preferences dialog box--Version tab . . . . .	14
Customizing Global Display Options . . . . .	14
Customizing Global Colors . . . . .	15
Customizing Global Fonts . . . . .	16
Options for Customizing the Display. . . . .	16
Setting the Default Date/Time Format . . . . .	17
Rearranging the Order of Date/Time Formats . . . . .	17
Adding a Date/Time Format . . . . .	17
Deleting a Date/Time Format . . . . .	18
About Creating Forms. . . . .	18
Creating a Form . . . . .	18
Defining Form Details . . . . .	20
Activating Form Components . . . . .	20

## Chapter 3. Form Component Modification . . . . . 23

About EDI File Formats . . . . .	23
Promoting Groups and Repeating Segments . . . . .	23
Splitting Groups and Repeating Segments . . . . .	23
Using Copy, Cut, and Paste . . . . .	24
Verifying EDI Delimiters . . . . .	25
Saving a File Definition . . . . .	25
Creating a Group . . . . .	26
Modifying Group Properties. . . . .	26
Modifying Segment Properties . . . . .	27
Defining a Loop Start . . . . .	28
Defining a Loop End . . . . .	29
About Elements . . . . .	29
Number Formats . . . . .	29
Formatting Numbers . . . . .	31
Date/Time Formats. . . . .	31
Formatting Dates and Times. . . . .	33
Formatting Strings . . . . .	34
Creating and Editing Syntax Tokens for Western European Languages . . . . .	34
Creating and Editing Syntax Tokens for East Asian Languages. . . . .	36
Deleting Syntax Tokens . . . . .	36
Deleting a Character Range . . . . .	37

Using Syntax Tokens . . . . .	37
Modifying Element Properties . . . . .	38
Storing EDI Code Value Descriptions. . . . .	39
Defining and Modifying Relational Conditions . . . . .	39

## Chapter 4. Form Completion . . . . . 43

Compiling a Translation Object. . . . .	43
The Print Function . . . . .	43
Completing the Form . . . . .	44

## Chapter 5. Standard Rules . . . . . 45

About Standard Rules . . . . .	45
Standard Rule Tab - select Function . . . . .	45
Using Select in a Form . . . . .	47
Using Information from the Partner Definition. . . . .	48
Using Information from Location Tables. . . . .	49
Using Information from Lookup Tables . . . . .	49
Using Information from Cross-Reference Tables . . . . .	50
The Update Function . . . . .	51
Standard Rule Tab--Update Function . . . . .	51
Using Update in a Form . . . . .	52
The Use System Variable Function. . . . .	52
Standard Rule Tab--Use System Variable Function . . . . .	52
Using the System Date and Time . . . . .	52
The Use Constant Function . . . . .	53
Using Constants in a Form . . . . .	53
About Literal Constants and Qualifying Relationships . . . . .	53
Defining and Editing Literal Constants . . . . .	54
Deleting Literal Constants . . . . .	55
Mapping Literal Constants . . . . .	55
Generating Qualifiers . . . . .	55
Standard Rule Tab - Use Accumulator Function . . . . .	56
Using an Accumulator in a Form . . . . .	59
Standard Rule Tab - Loop Count Function . . . . .	59
Using the Loop Count Function . . . . .	59
Standard Rule Tab - Use Code Function . . . . .	60
Using Use Code in a Form . . . . .	62
Code List Tables. . . . .	62
Defining and Modifying a Code List . . . . .	63
Deleting a Code List . . . . .	64
Deleting a Code List Entry . . . . .	64
Importing a Code List. . . . .	64
Exporting a Code List . . . . .	65
Loading a Code List Table from the Standard . . . . .	65
Copying and Pasting Code Lists . . . . .	66
Validating Data Against Code List Tables . . . . .	67
Loading Code Item Descriptions . . . . .	67

## Chapter 6. Extended Rules . . . . . 69

About Extended Rules. . . . .	69
Declarations and Initialization . . . . .	69
Statements. . . . .	70
When Extended Rules are Processed . . . . .	71
How to Define Extended Rules . . . . .	72

Defining a Session Rule . . . . .	73
Defining a Form Component Extended Rule . . . . .	73
Extended Rule Syntax . . . . .	74
Keywords and Commands . . . . .	74
Operators . . . . .	76
Symbols . . . . .	76
Extended Rule Functions . . . . .	79
About the Extended Rule Functions . . . . .	79
atoi . . . . .	83
aton . . . . .	84
auditlog . . . . .	84
begin ... end . . . . .	85
break . . . . .	86
cerror . . . . .	86
concat . . . . .	90
continue . . . . .	91
count . . . . .	91
createobject . . . . .	91
date . . . . .	92
delete . . . . .	93
deleteobject . . . . .	93
empty . . . . .	94
exec . . . . .	94
exist . . . . .	95
fseek . . . . .	96
ftell . . . . .	97
get . . . . .	97
getiid . . . . .	98
if ... then ... else . . . . .	98
index . . . . .	99
insert . . . . .	100
left . . . . .	101
len . . . . .	102
messagebox . . . . .	102
mid . . . . .	104
ntoa . . . . .	104
param . . . . .	105
queryobject . . . . .	106
readblock . . . . .	106
readbytes . . . . .	108
right . . . . .	108
select . . . . .	109
set . . . . .	109
strdate . . . . .	110
strstr . . . . .	112
unreadblock . . . . .	113
update . . . . .	114
while ... do . . . . .	115
winexec . . . . .	116
writeblock . . . . .	117
writebytes . . . . .	118
select and update Options . . . . .	119

**Chapter 7. Form Customization. . . . . 127**

About Customizing Forms . . . . .	127
Customizing Field Labels . . . . .	127
Hiding Fields . . . . .	127
Setting an Initial Value for a Field . . . . .	128
Using a Constant Value for a Field . . . . .	129
Creating Definitions for a List Box . . . . .	129
Creating Help Text for Fields . . . . .	130
Creating Help Text for List Boxes. . . . .	130
Creating Help Text for Frames. . . . .	131
Formatting Entry Fields . . . . .	131
Formatting Display-Only (Non-Editable) Fields . . . . .	132

**Chapter 8. Form Formatting . . . . . 133**

Selecting One or More Fields or Field Labels . . . . .	133
Grouping Fields and Field Labels. . . . .	134
About Resizing Fields . . . . .	134
Using Size to Length . . . . .	134
Resizing a Field Manually . . . . .	135
Resizing Groups of Fields Manually . . . . .	135
Changing Grid Settings . . . . .	136
Using Alignment Lines . . . . .	136
Moving Fields and Field Labels . . . . .	137
Aligning Fields and Field Labels . . . . .	137
Spacing Fields and Field Labels . . . . .	138
Resizing and Positioning a Frame . . . . .	139
Tips for Resizing Drop-Down Combo Box . . . . .	139
Resizing Drop-Down Combo Boxes . . . . .	140
Setting Tab Sequences . . . . .	140
Adding and Positioning Static Text . . . . .	141
Modifying Frame Titles . . . . .	141
Modifying List Box Names . . . . .	142
Adding Column Headings to List Boxes . . . . .	142

**Chapter 9. User Exits . . . . . 145**

About ActiveX Technology . . . . .	145
About User Exits . . . . .	146
ActiveX and User Exit Functions . . . . .	148
Examples of User Exits . . . . .	149
Examples of Automation Servers . . . . .	152
Creating a User Exit . . . . .	154

**Chapter 10. Translator Command Line Interface . . . . . 157**

About the Command Line Interface . . . . .	157
--	-----

**Chapter 11. Error Messages . . . . . 159**

About Error Messages . . . . .	159
Compile Error Messages. . . . .	159
Sterling Gentran:Server Error Messages. . . . .	164

**Notices . . . . . 169**

---

# Chapter 1. Form Integration Basics

---

## About Forms Integration

The IBM® Sterling Gentran:Server® for Microsoft Windows Forms Integration subsystem enables you to design forms to facilitate the keying (screen entry) and printing (print) of EDI documents that you receive inbound or send outbound.

A form is a set of instructions that you define in the Forms Integration subsystem to indicate how the system should format data. When you compile the form, Sterling Gentran:Server generates a translation object. After you register the translation object with Sterling Gentran:Server and associate the translation object with a partner relationship, the system uses the translation object to format EDI documents.

This table describes the two types of forms that you can create using Forms Integration.

Form	Description
Print translation object	Organizes and formats the printout of EDI documents that are received from or sent to the trading partners for which you have established a trading relationship (inbound or outbound) that utilizes that print translation object. The print translation object enables you to view the EDI document in an easily readable format.
Screen entry translation object	Provides a standardized format for keying an EDI document into the Document Editor, for translation and transmission to your trading partners, for which you have established an outbound trading relationship that utilizes that screen entry translation object. The screen entry translation object ensures that your users key all the data necessary to create the required EDI document.

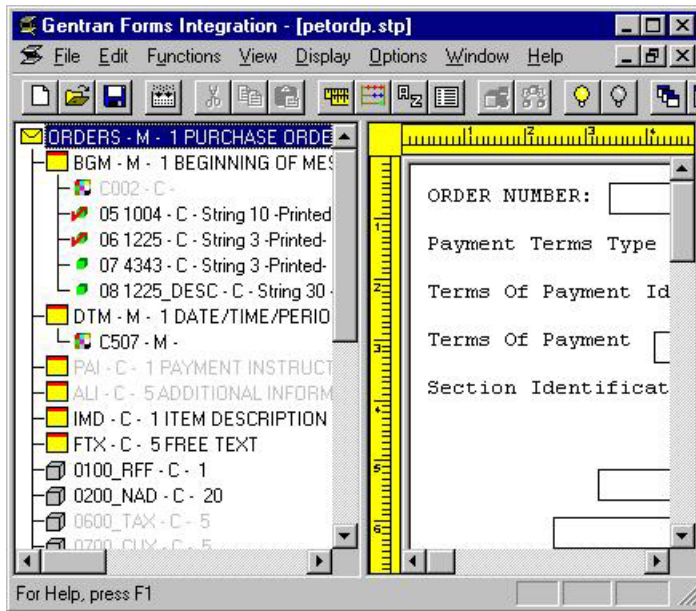
---

## Forms Integration User Interface

The Forms Integration subsystem contains two windows: The EDI File Format Window (left side of the screen) and the Layout Window (right side of the screen).

When you click a segment or group in the EDI File Format Window, the corresponding frame is displayed in the Layout Window. When you click an element in the EDI File Format Window, the corresponding layout component is highlighted in the Layout Window.

This illustration shows the Forms Integration Subsystem Main Window.



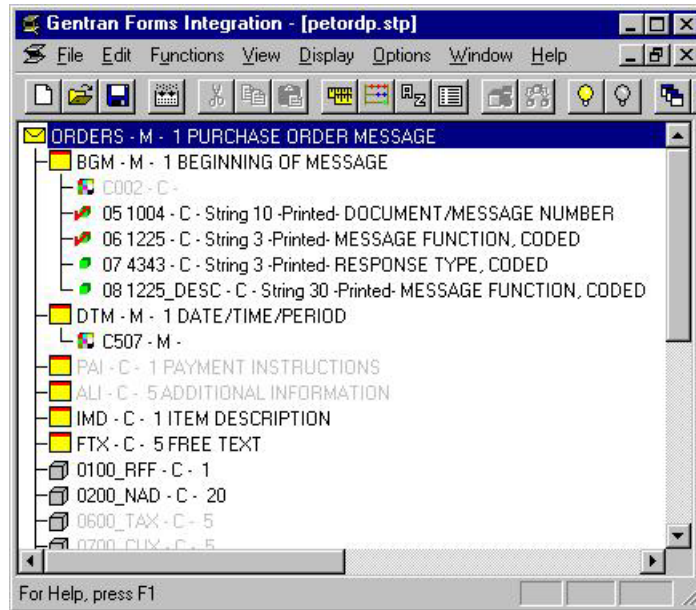
This table describes the Forms Integration Main Window parts and their functions.

Window	Description
Title Bar	Displays the title of the open form.
Main Menu Bar	Contains drop-down menus of Forms Integration commands.
Mail Toolbar	Provide short cut options to common Forms Integration components.
The EDI File Format Window (left side of the Forms Integration subsystem main window)	Contains the EDI file format.
The Layout Window (right side of the Forms Integration subsystem main window)	Contains the actual format of the translation object.







## EDI File Format Window

This illustration shows the EDI File Format Window.



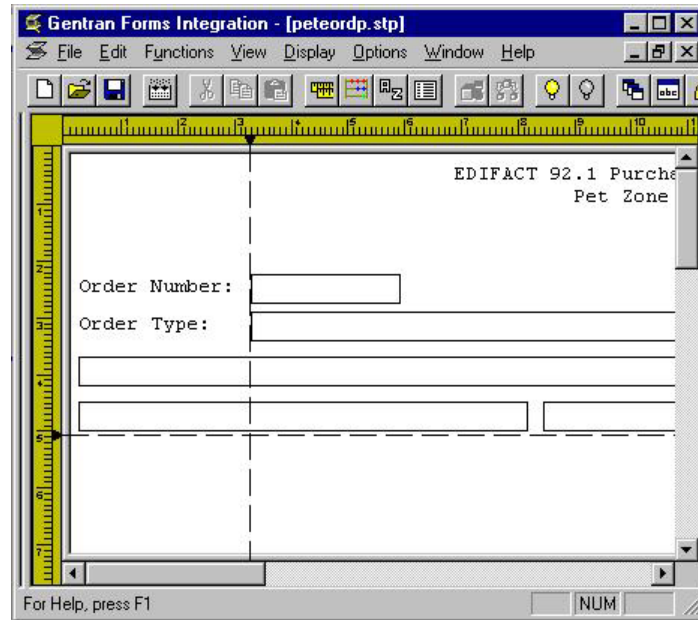
This table describes the parts of the EDI File Format window and their functions.

Icon	Description
	<p>A group 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.</p> <p>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).</p>
	<p>A 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.</p>

Icon	Description
	<p>A composite 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).</p> <p>A repeating composite is a related group of EDI subelements that have the ability to loop as a whole (occur more than once) within a particular EDI segment. To allow a composite to repeat multiple times within a segment, each occurrence of the composite must be separated by a special delimiter, known as the repeating element delimiter.</p> <p><b>For example:</b> Your delimiters are set as follows:</p> <ul style="list-style-type: none"> <li>• Segment ~</li> <li>• Element *</li> <li>• Tag *</li> <li>• Sub Element :</li> <li>• Repeating Element ^</li> </ul> <p>Then, if a BGM segment contains a repeating composite that occurs 3 times within the data followed by 1 regular element, and the composite contains 2 subelements, the segment is in the following form: BGM*SubA-1:SubB-1^SubA-2:SubB-2^SubA-3:SubB-3*Field2~</p>
	<p>An element is the smallest piece of information defined by the EDI standards. An individual element can have different meanings depending on the context. Therefore, elements are normally not considered to have useful meaning until they are combined into segments.</p> <p>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.</p> <p>A repeating element is an EDI elements with the ability to loop (occur more than once) within a particular EDI segment. To allow a single element to repeat multiple times within a segment, each element must be separated by a special delimiter, known as the repeating element delimiter. The use of this delimiter prevents the system from mistaking the repeating elements for normal elements.</p> <p><b>For example:</b> Your delimiters are set as follows:</p> <ul style="list-style-type: none"> <li>• Segment ~</li> <li>• Element *</li> <li>• Tag *</li> <li>• Sub Element :</li> <li>• Repeating Element ^</li> </ul> <p>Then, if a BGM segment contains 3 elements and the second element is a repeating element that occurs 4 times, the segment is in the following form: BGM*Field1*Field2.1^Field2.2^Field2.3^Field2.4*Field3~</p>

## Layout Window

This illustration shows the Layout Window.



This table describes the parts of the Layout window and their functions.

Part	Function
Horizontal and Vertical Rule	Enables you to manipulate the length, width, and alignment of your form.
Field Label <b>Note:</b> In the Layout Window illustration, Order Number: is an example of a field label.	Defines the field in which a user enters information.
Field	Text box associated with a field label in which a user can enter information. For print translation objects, groups, repeating segments, and elements are always formatted as fields.
Frame	Used to format the EDI file for the translation object. Each frame contains the groups, repeating segments, and elements at that level (single segments are not represented on the frame).
Alignment Lines	Vertical or horizontal grid lines that enable you to align the components of the Layout Window.

## List Formats

For screen entry translation objects, groups (with a maximum usage greater than one) and repeating segments are formatted as lists. Elements are formatted as edit boxes (fields) or lists on the frame. Elements are typically formatted as fields, but are formatted as a list if you apply a standard rule that allows a selection of multiple items. Each group and repeating segment also has a corresponding frame that contains all of the groups (with a maximum usage greater than one), repeating segments, and elements at that level.

---

## The Form Building Process

This topic defines how to build a form using the Forms Integration subsystem.

This table describes the process to build a form using Forms Integration.

Stage	Description
1	Prepare and Analyze how you want your form to look and what components your form will need.  See Preparation and Analysis for more information.
2	Set Global Defaults (first time only).  See About Global Defaults for more information.
3	Create, Save, and Name a New Form.  See the Creating a Form and Defining Form Details for more information.
4	Activate the Appropriate EDI Groups, Segments, and Elements.  Please see Activating Form Components for more information.
5	Define the Layout of the Form.  See About EDI File Formats for more information on customizing your EDI file, including setting the EDI delimiters.  See the Standard Rules and Extended Rules sections of this guide for more information.
6	Compile the Translation Object.  See Compiling a Translation Object for more information on compiling the translation object and translation object naming conventions.
7	Print the Report.  See The Print Function for more information.
8	Register the Translation Object with Sterling Gentran:Server.  See the <i>IBM Sterling Gentran:Server for Microsoft Windows for Microsoft Windows User Guide</i> for more information about registering translation objects.
9	Create the Appropriate Trading Relationship.  Establish the appropriate trading relationship in Sterling Gentran:Server for your trading partners.  See the <i>IBM Sterling Gentran:Server for Microsoft Windows for Microsoft Windows User Guide</i> for more information about creating trading relationships.
10	Validate the Translation Object.  For a screen entry translation object, create a new document in Document Editor, using that translation object. For a print translation object, print the document. Verify that the output is correct.  See the <i>IBM Sterling Gentran:Server for Microsoft Windows for Microsoft Windows User Guide</i> for more information about creating a new document in the Document Editor using that translation object.

---

## Chapter 2. Form Design

---

### Preparation and Analysis

Before you begin to create a form, work with your trading partner to:

- Determine necessary form fields and labels
- Obtain a layout of how you want the completed form to look
- Determine how your layout corresponds with the EDI standard you use
- Determine the operations that you need perform to generate the required layout

---

### About Global Defaults

The Preferences dialog box is a property sheet that enables you to set global Form Integration defaults. This table describes the customizable global preference options that you can define on the Preferences dialog box to control the look of your form display

Form display options can be set or changed at any time.

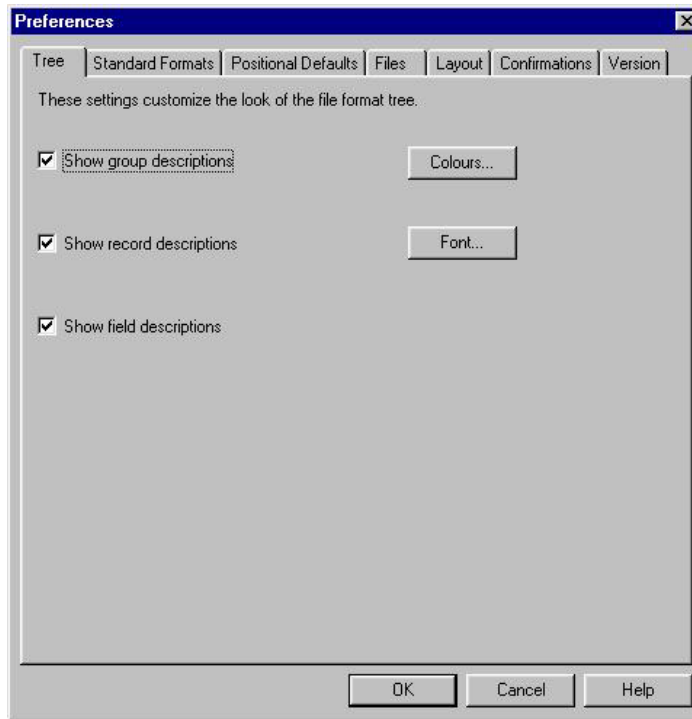
Customizable Items	Description
Tree settings (Global option)	Enables you to display group, record, and field descriptions.  You can also access: <ul style="list-style-type: none"><li>• Colors (Global option) - Enables you to select foreground and background colors to visually define form components. The use of color is optional.</li><li>• Fonts (Global option) - Enables you to globally change how fonts display. This includes the typeface, style, and size. The default font that Sterling Gentrans:Server uses is a Sans Serif 9 point.</li></ul>
Standard Formats	Displays the default date to use when elements are read from the Standards database.  Valid options are: <ul style="list-style-type: none"><li>• Six-character dates</li><li>• Eight-character dates.</li></ul>
Positional Defaults	Defines default field formats.
Files	Defines how Forms Integration works with files.
Layout	Defines the look of the layout window.
Confirmations	Controls when you receive a confirmation prompt.
Version	Controls how and when the system increments a form's version number.

### Date/Time Format

The Default Date formats dialog box, available from the Options menu, enables you to Control the default date and time formats that are used when elements are read from the standards database.

## Preferences dialog box - Tree tab

This illustration shows the Tree tab of the Preferences dialog box.

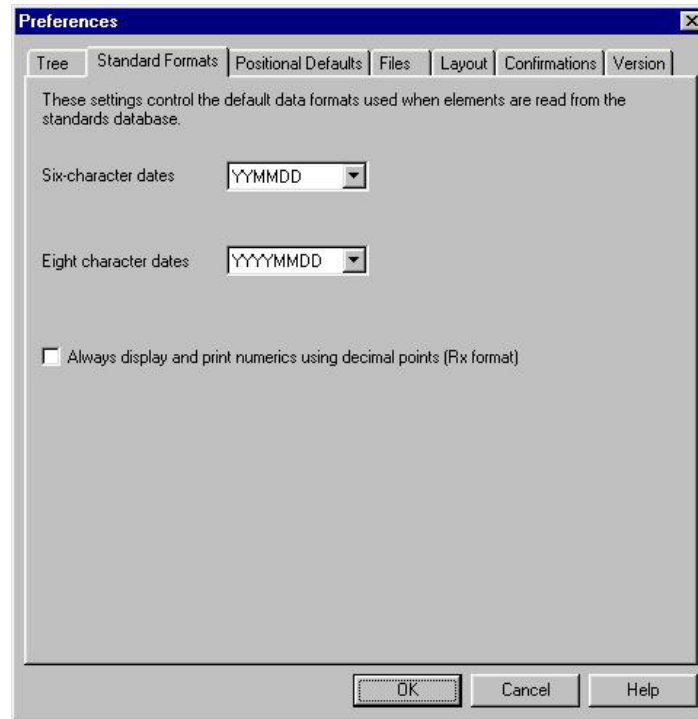


This table describes the parts and functions of the Tree tab on the Preferences dialog box.

Part	Function
Descriptions check boxes	<p>Defines what form details that you want to display in the file format tree.</p> <p>Options are:</p> <ul style="list-style-type: none"> <li>• Show group descriptions</li> <li>• Show record descriptions</li> <li>• Show field descriptions</li> </ul>
Colors	<p>Accesses the Colors dialog box, which enables you to select foreground and background colors to visually define form items and/or attributes. The use of color is optional.</p>
Fonts	<p>Accesses the Fonts dialog box, which enables you to globally change how fonts display. Customizable font options include the typeface, style, and size. The default font that Sterling Gentran:Server uses is a Sans Serif 9 point.</p>

## Preferences dialog box - Standard Formats tab

This illustration shows the Standard Formats tab of the Preferences dialog box.

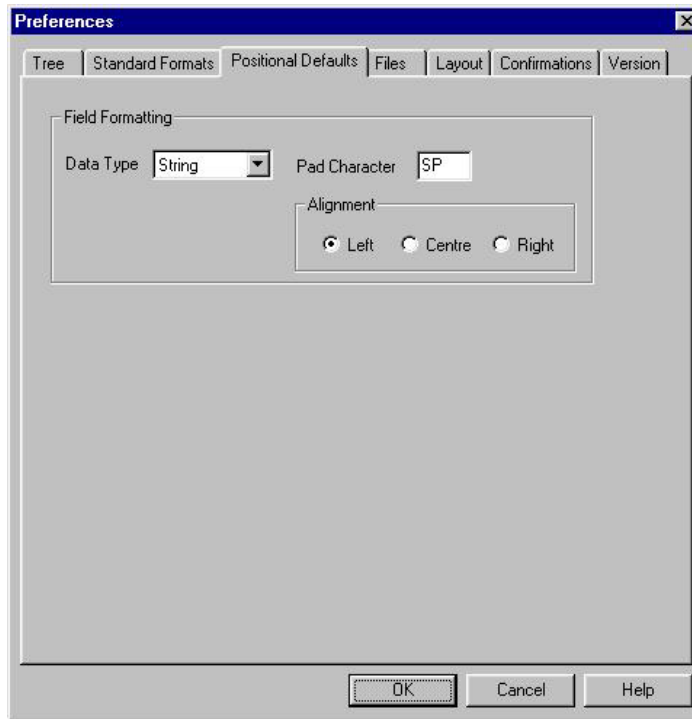


This table describes the parts and functions of the Standard Format tab on the Preferences dialog box.

Part	Function
Date lists	Controls the default date formats that are used when the element is read from the Standards database.  Customizable items include: <ul style="list-style-type: none"><li>• Six-character dates</li><li>• Eight-character dates</li></ul>
Always display and print numerics using decimal points check box	Specifies that Sterling Gentran:Server should display and print all numeric-format elements using decimal points (formatted as real). This default data format is used when elements are read from the standards database.

## Preferences dialog box - Positional Default tab

This illustration shows the Positional Defaults tab of the Preferences dialog box.



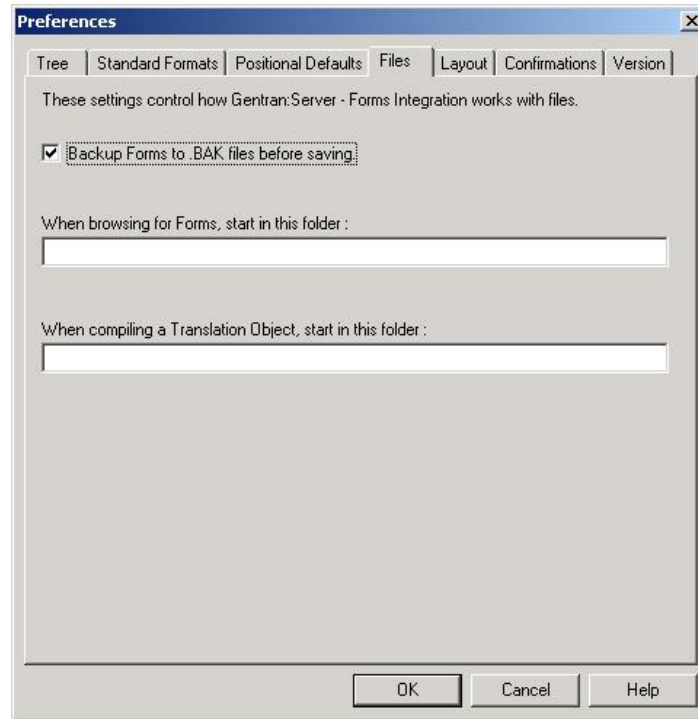
This table describes the parts and functions of the Positional Defaults tab on the Preferences dialog box.

Part	Function
Field Formatting Data Type	Describes the default data type. Valid options are: <ul style="list-style-type: none"><li>• String</li><li>• Number</li><li>• Dt/Tm</li></ul>
Pad character	Specifies whether you want characters padded by default.
Alignment	Specifies field formatting alignment. Valid options are: <ul style="list-style-type: none"><li>• Left</li><li>• Centre</li><li>• Right</li></ul>



## Preferences dialog box - Files tab

This illustration shows the Files tab of the Preferences dialog box.

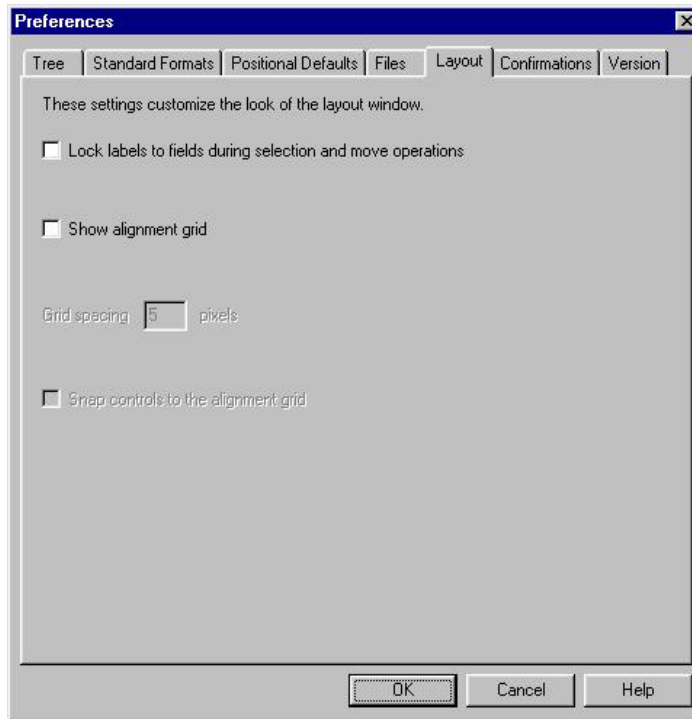


This table describes the parts and functions of the Files tab on the Preferences dialog box.

Part	Function
Backup forms to .BAK files before saving	A selected check box indicates that you want to create backup Forms Integration files.
When browsing to Forms, start at this folder	Specifies the path and folder where you want the system to begin browsing for the Forms Integration subsystem. The default location is C:\GENSRVNT\Forms.
When compiling a translation object, start at this folder	Specifies the path and folder where you want the system to start the translation object compilation process.

## Preferences dialog box--Layout tab

This illustration describes the Layout tab of the Preferences dialog box.

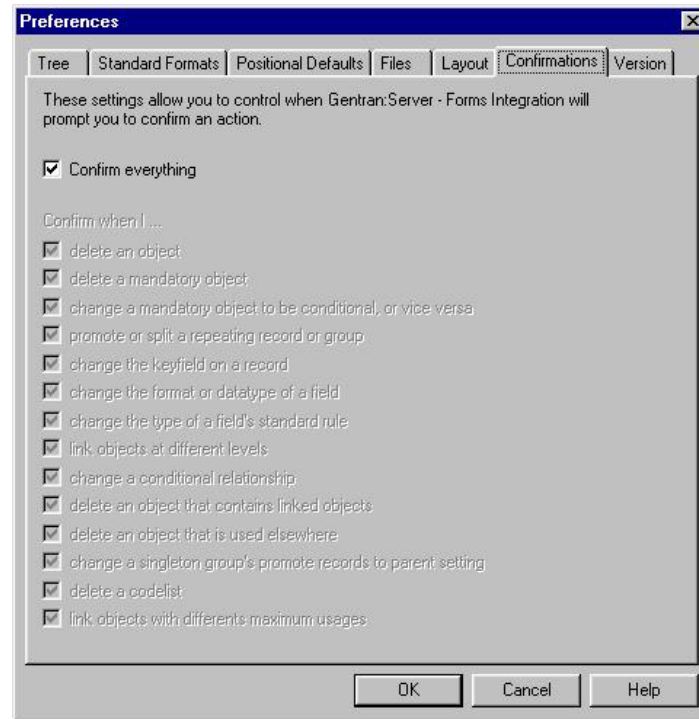


This table describes the parts and functions of the Layout tab on the Preferences dialog box.

Part	Function
Lock labels to field during selection and move operations check box	Groups the field label with the field in the Layout Window.
Show alignment grid	Displays Layout Window grid lines for alignment purposes.
Grid spacing in Pixels	Defines default grid line spacing in Pixels.
Snap controls to the alignment grid.	Aligns a field label and field on the nearest alignment grid setting.

## Preferences dialog box--Confirmations tab

This illustration describes the Confirmations tab of the Preferences dialog box.



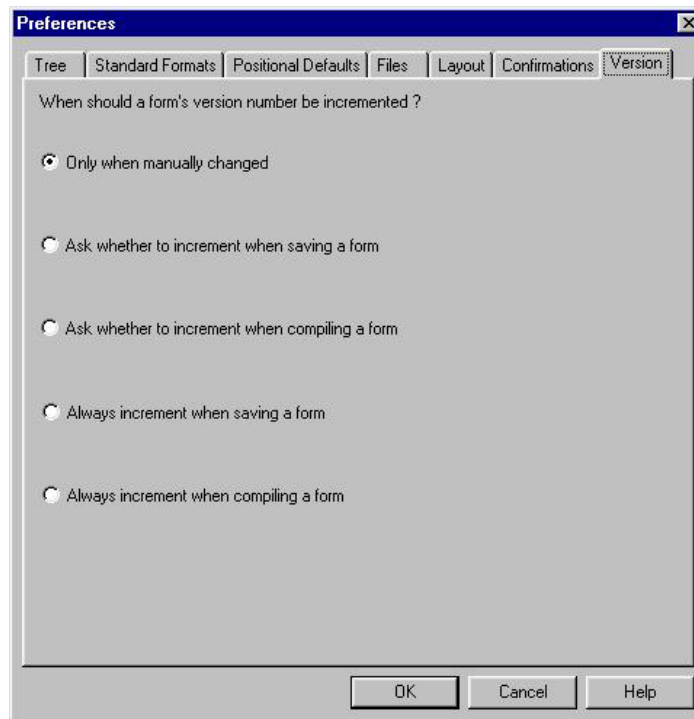
This table describes the parts and functions of the Confirmations tab on the Preferences dialog box.

Part	Function
Confirm everything	Specifies that you want to receive a prompt to confirm all actions.
Confirm when I...	Specifies specific instances in which you want to receive a prompt to confirm an action. Valid options are: <ul style="list-style-type: none"> <li>• Delete an object</li> <li>• Delete a mandatory object</li> <li>• Change a mandatory object to be conditional or vice versa</li> <li>• Promote or split a repeating record or group</li> <li>• Change the keyfield on a record</li> <li>• Change the format or datatype of a field</li> <li>• Change the type of a field's standard rule</li> <li>• Link objects at different levels</li> <li>• Change a conditional relationship</li> <li>• Delete an object that contains linked objects</li> <li>• Delete an object that is used elsewhere</li> <li>• Change a single group's promote record to parent setting</li> <li>• Delete a codelist</li> <li>• Link objects with different maximum usages.</li> </ul>

---

## Preferences dialog box--Version tab

This illustration describes the Version tab of the Preferences dialog box.



This table describes the parts and functions of the Version tab on the Preferences dialog box.

Part	Function
When should a form's version number be incremented?	Options are: <ul style="list-style-type: none"><li>• Only when manually changed - specifies that you want the system to increment the form's version number when you manually make a change.</li><li>• Ask whether to increment when saving a form - specifies that, upon saving a form, you want the system to display a prompt asking whether you want the form's version number to increment.</li><li>• Ask whether to increment when compiling a form - specifies that, upon compiling a form, you want the system to display a prompt asking whether you want the form's version number to increment.</li><li>• Always increment when saving a form - specifies that, upon saving a form, you want the system to automatically increment the version number.</li><li>• Always increment when compiling a form - specifies that, upon compiling a form, you want the system to automatically increment the version number.</li></ul>

---

## 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.

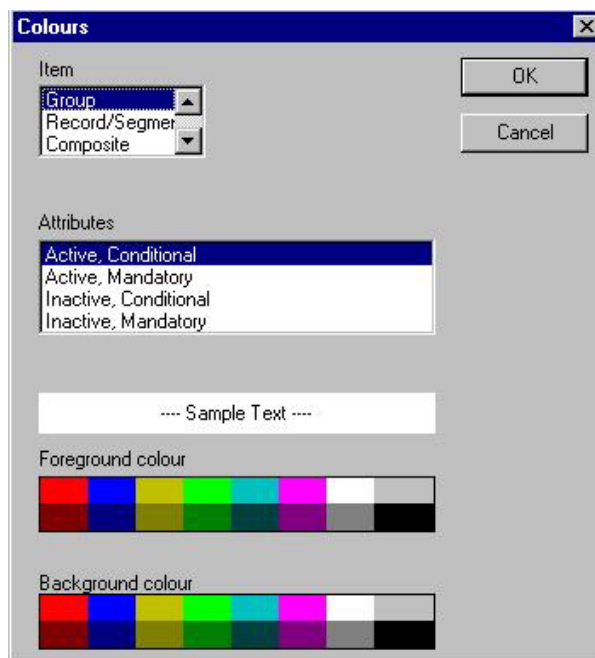
### About this task

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.

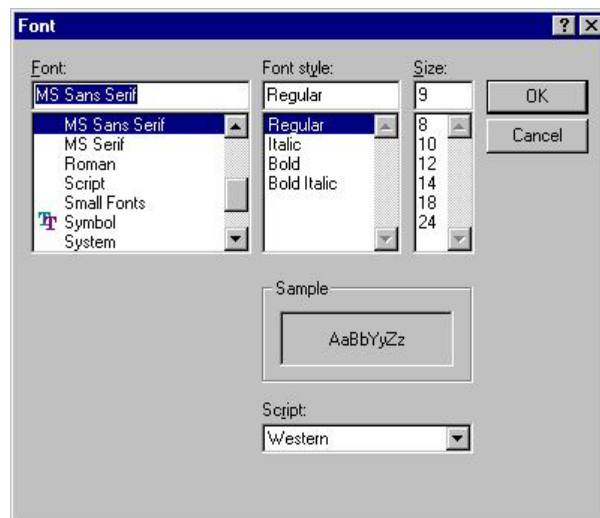
### About this task

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.

---

## Options for Customizing the Display

Sterling Gentran:Server enables you to choose how you want the EDI File Format and Layout Windows to display. These options, which are accessible from the View menu of the Main Menu bar, include:

- Maximize Tree - maximizes the EDI File Format Window.
- Maximize Layout - maximizes the Layout Window.
- Rearrange toggles the display of the EDI File Format Window and the Layout Window from left/right to above/below.

Additionally, Sterling Gentran:Server enables you to customize the measurement system that the vertical and horizontal rulers in the Layout Window use. The default measurement system is inches, but you can toggle the display to centimeters by double-clicking the yellow square where the vertical and horizontal rulers meet.

---

## 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**.

---

## Rearranging the Order of Date/Time Formats

Use this procedure to rearrange the order of Date/Time formats that appear in the Six- and Eight-character date lists on the Standard Format tab of the Preferences dialog box.

### Procedure

1. Select **Options > Date Formats**.
2. Select the Date/Time format that you want to move up or move down in the list order.
3. Press **Up** or **Down** until the date format is in the desired list position.
4. Click **OK** to save your changes and to exit the dialog box.

---

## Adding a Date/Time Format

Use this procedure to add a new Date/Time format to the Six- or Eight-character date lists on the Standard Format tab of the Preferences dialog box.

### Procedure

1. Select **Options > Date Formats**.
2. Click **New**.

3. In the Name box, enter the name of the new date format that you want to create.
4. In the Picture box, enter how you want the date format to display.
5. Click **OK** to save your changes and to exit the Date/Time Formats dialog box.

---

## Deleting a Date/Time Format

Use this procedure to delete a new Date/Time format from the Six- or Eight-character date lists on the Standard Format tab of the Preferences dialog box.

### Procedure

1. Select **Options > Date Formats**.
2. Select the Date/Time format that you want to delete.
3. Press **Delete**.

The system displays a warning dialog that specifies that you must manually correct formats for fields that use that date format that you want to delete.

4. Verify that you want to delete the date/time format, and click **Yes**.

**Note:** To recover the Date/Time format that you just deleted, click **Cancel** on the Date/Time Format dialog box.

You return to the Date/Time Format dialog box.

5. Click **OK** to save your changes and to exit the Date/Time Formats dialog box.

---

## About Creating Forms

You can create two different types of forms:

- Screen Entry - for keying documents in Document Editor
- Print - for printing documents

After you create a new form, it is important to define and save the form details. Form components include the standard, version, transaction set (document) selected, groups, segments, composites, and elements that your company requires. The specific form components that you use depend on the type of form that you create.

After you select your form components, Sterling Gentran:Server generates an EDI file format for you.

We recommend that you determine which form components you are using before generating or defining an EDI file format.

---

## Creating a Form

Use this procedure to create a new form.

### Procedure

1. Select **File > New**.

The system displays the New Form Wizard dialog.

2. In the dialog box, do the following:
  - Select the type of form.
  - Enter a name for the form.



We recommend that you use the following identifying characteristics: the partner that this form is used for, the standard, the version, the type of transaction this form uses, and the type of form.

For example, "MWT X 3030 850 Print" is the description of a print translation object used to print documents for partner MWT, for an ANSI X.12 version 003030 Purchase Order (850).

- Verify your user ID.
  - Click **Next**.
3. If you want to load the data format from a previously saved file definition, continue with Step 15. Otherwise, select **Create a new data format using this syntax**.
  4. From the list, select a syntax option.
    - If you selected Delimited EDI or CII, click **Customize** and continue with the next step.
    - If you selected Positional, click **Next** and **Finish**. You have completed this procedure.
  5. Click **Next**.
  6. Select the ODBC data source that you want to use from the list and click **Next**. The default is Sterling Gentran<sup>®</sup> Standards.
  7. Select the standards agency that you want to use from the list.
  8. Select the version of the standard from the Version list. The versions that are available depend on which standard you selected.
  9. Select the transaction set (document or message) from the Transaction list. The transactions that are available depend on which standard and version you selected.

**Note:** For TRADACOMS only, select the release number of the version that you are using from the Release list. The releases that are available depend on which standard, version, and transaction you selected.

10. Click **Next**.
11. If you are using CII syntax, select a Multi-Detail Header option.
12. Click **Finish** to load the transaction set.  
You return to the New Form Wizard.
13. Click **Next**.
14. Click **Finish** to begin editing your form.  
You have completed this procedure.
15. The following steps apply to loading the data format from a previously saved file definition.  
Select **Load the data format from a saved definition**.
16. Type the path and file name that you want to load or click **Browse** to navigate to the file.
17. The default folder is GentranNT\Bin. The default file extension is .IFD or .DDF.
18. Click **Next**.
19. Click **Finish**.

The system finishes your form and displays the EDI file format in the Sterling Gentran:Server - Forms Integration Window.

---

## Defining Form Details

Use this procedure to define form details.

### Procedure

1. Open the form and select **Edit > Details**.  
The system displays the Translation Object Details dialog box.
2. To designate this form as a system translation object (one that is used internally by the system), under the Flags section, select **System**.

**Note:** Be certain that you want to define this translation object as a system translation object. Once a translation object is registered with Sterling Gentran:Server, it cannot be deleted.

3. To designate a version number for this translation object, under Version Control, enter the version number in the Major box and the release number in the Minor box.
4. The Input and Output boxes (Agency, Version, Transaction, Release, and F Group) contain the EDI format. If required, make the desired changes.

**Note:** You can change the information in these boxes but it will not alter the content of the form. You may wish to alter these boxes if, for example, you want the form to reflect a standards version that is not loaded on your system. You can change the version on this dialog box, and then alter the form to be compliant with that version.

5. Click **OK**.  
The system displays the Save As dialog box. We recommend that you store forms in the translation object folder where Sterling Gentran:Server is installed.
6. Enter the name of the form (up to eight characters) and click **Save**. The default file extension is .STP.

---

## Activating Form Components

### About this task

When Sterling Gentran:Server generates the form, 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" (must be present) by the standard. The system does not enable you to deactivate the mandatory groups, segments, composites, and elements.

When translating data, the system does not process groups, segments, composites, and elements 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 the form.

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

### Procedure

1. Click **Activate** from the main toolbar.
2. Double-click to open each group that contains groups or segments that you want to activate.

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

3. Click each inactive group that you want to activate.

**Note:** If you activate a group, segment, composite, or element by mistake, right-click the item and select **Deactivate**.

4. Click each inactive segment that you want to activate.
5. Double-click each segment that contains composites or elements that you want to activate and click each inactive composite that you want to activate.
6. Double-click each composite that contains elements that you want to activate and click each inactive element that you want to activate.

**Note:** When you activate an element for a print translation object, the word "Printed" displays before that element's name. When you activate an element for a screen entry translation object, the words "Edit Box" or "Drop Down" display before that element's name. "Edit Box" indicates that the element is formatted as a field. "Drop Down" indicates that the element is formatted as a list because you applied a standard rule that allows a selection of multiple items.

7. Verify that all groups, segments, and composites and elements that you need are activated.
8. Click **Activate** from the main toolbar to turn activation mode off.



---

## Chapter 3. Form Component Modification

---

### About EDI File Formats

You can modify the system-generated EDI file format by modifying the form component properties and by using the promote, split, copy, cut, and paste functions. See *Creating a Form and Defining Form Details* for more information.

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

---

### 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**.

---

## 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.

### About this task

Delimiters are flags that you define to the system as separating specific EDI components. 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.

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 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 Format Properties dialog box.
2. Click the **Delimiters** tab.
3. If the required delimiters for the EDI standard you are using differ from the default Delimiter tab settings, type either the character or the hexadecimal value in the correct box.
4. Click **OK** to exit the File Format Properties dialog box.

---

## 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 forms. This provides you with a quick way to build a form.

### About this task

Use this procedure to save an individual file format definition.

### Procedure

1. Open the form.
2. Right-click the **EDI file** icon and select **Save File Definition As**.  
The system displays the Save File Definition dialog box.
3. If necessary, navigate to the folder where Sterling Gentran:Server is installed.  
The default is GENSRVNT.

4. Select the file definition from the list or type the name of the file definition in the File name box. The default file extension is .IFD or .DDF.
5. Click **Save**.

---

## Creating a Group

A group contains related segments and/or groups 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. Do one of the following:
  - To create a group that is at the same level as a selected form component, highlight the form component and select **Edit > Insert > Group**.
  - To create a group that is subordinate to a selected form component, highlight the form component and select **Edit Create Sub > Group**.  
The system displays the Name tab of the Group Properties dialog box.
2. Enter the group name.

**Note:** Do not use spaces or dashes (-) in the Name box. You can use the underscore (\_) to separate words.

3. Enter a description for the loop.
4. Do one of the following:
  - For a screen entry form, click the **Display** tab and select your display options.
  - For a print form, click the **Print** tab and select your print options.
5. Click the **Looping** tab.
6. In the Min Usage box, type the minimum number of times the loop must be repeated.  
  
**Note:** The minimum usage should be "1" or greater.
7. In the Max Usage box, type the maximum number of times the loop can be repeated.
8. If you want to specify an extended rule for this group, click the **Loop Extended Rules** tab.
9. Click **OK** to create the group.

---

## 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**.
2. On the Name tab, do one of the following:
  - To modify the group name, type the group name in the first box.  
  
**Note:** If a group occurs more than once in a 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 form.
  - To modify the group description, type the new description in the second box. This box is used to provide a brief explanation of the loop.
3. To change any display or print options, do one of the following:
  - For a screen entry form, click the **Display** tab and make your changes.
  - For a print form, click the **Print** tab and make your changes.
4. Click the **Looping** tab to access looping options.
5. In the Min Usage box, type the minimum number of times the loop must be repeated.  
  
**Note:** For a conditional loop, the minimum usage should always be "0" (zero). For a mandatory loop, the minimum usage should be "1" or greater.
6. In the Max Usage box, type the maximum number of times the loop can be repeated.
7. To specify an extended rule for this group, click the **Loop Extended Rules** tab and define the minimum and maximum usage.
8. Click **OK** to save the 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

Use this procedure to modify the properties of a segment.

## Procedure

1. Right-click the segment you want to modify and select **Properties**.
2. To change the segment name, type the segment name in the first box on the Name tab.  
  
**Note:** If a segment occurs more than once in a 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 form.
3. To change the segment description, 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. To flag this segment as containing the binary data, click the **Special** tab.  
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.

5. To use a key field, click the **Key Field** tab and make your selections. 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.
6. If you want to change any display or print options, do one of the following:
  - For a screen entry form, click the **Display** tab and make your changes.
  - For a print form, click the **Print** tab and make your changes.
7. Click the **Looping** tab.
8. In the Min Usage box, type the minimum amount of times the segment must repeat.

**Note:** If the Min Usage box contains a "0" (zero), the segment is "conditional." If the Min Usage box contains a "1" or greater, the segment is "mandatory."

9. If this segment is a looping structure, type the maximum amount of times it can repeat in the Max Usage box.
10. Select one of the following options:
  - Loop Start - This segment marks the beginning of the loop.
  - Loop End - This segment marks the end of the loop.
  - Normal - This segment is in the loop but is not the beginning or ending segment.
11. To specify an extended rule for this group, click the **Loop Extended Rules** tab and make your selections.

See Defining a Form Component Extended Rule for more information.
12. Click **OK** to save the changes to the segment.

---

## Defining a Loop Start

Certain EDI standards use Loop Start (LS) and Loop End (LE) segments. LS and LE segments specify the explicit start and end of loops.

### About this task

Use this procedure to define an LS segment for a form.

### Procedure

1. Right-click the **LS segment** and select **Properties**.
2. On the **Looping** tab, verify that the **Loop Start** option is selected.
3. Click the **Key Field** tab.
4. From the Field list, select the **Loop Identifier Code**. This list contains all the elements that are contained in the segment and allows you 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.
5. The Matching Rules section enables you to access all the literal constants and code lists currently defined for this form. Click the **Use constant** option.
6. Click **Edit** to access the Map Constants dialog box.
7. Click **New** to access the Edit Constant dialog box.
8. In the ID box, type the literal constant identifier.
9. From the Type list, select **String**. This is the category of this literal constant.

10. In the Value box, type the value of the literal constant.
11. For a screen entry form, this is the loop identifier code that you expect to receive from your trading partner.
12. Click **OK** to add the constant to the system.
13. Select the constant you created from the list. The system matches the selected constant against the loop identifier code.
14. Click **OK** to exit the Segment Properties dialog box.

---

## Defining a Loop End

Certain EDI standards use Loop Start (LS) and Loop End (LE) segments. LS and LE segments specify the explicit start and end of loops.

### About this task

Use this procedure to define an LE segment for a form.

### Procedure

1. Right-click the **LE segment** and select **Properties**.
2. On the **Looping** tab, verify that the **Loop End** option is selected.
3. Click the **Key Field** tab to access the key field options.
4. From the Field list, select the **Loop Identifier Code**. This list contains all the elements that are contained in the segment. This list 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.
5. The Matching Rules section enables you to access all the literal constants and code lists currently defined for this form. Select the constant you created for the LS segment from the Use Constant list. The system matches the selected constant against the loop identifier code.
6. Click **OK** to exit the Segment Properties dialog box.

---

## About Elements

When you define or modify an element, you must specify the type and format. Available element format options depend on which element type you select (string, number, or Date/Time).

---

## 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 1. 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.

Table 1. Real Number Format Options (continued)

Field	Description
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 2. 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.
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.

---

## Formatting Numbers

Use this procedure to format numbers in a screen entry or print form.

### Procedure

1. Double-click an existing element.  
The system displays the Element Properties dialog box.
2. Click the **Validation** tab to access validation options.
3. From the data-type list, select **Number**. This indicates that the element is a numeric or real number that can be mathematically manipulated.
4. From the form format list, select the format for the number.

**Note:** Usually, the number formats in the form format list and in the file format list are the same. However, if there is an "N" format selected in the form format list, you might want to select an "R" format in the file format list so that the decimal point in the numbers is displayed.

5. Verify that the number format selected in the file format list is the correct format that is written to the EDI file.

**Note:** A number format must be designated for number-type elements in the file format list, even for hidden fields.

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

---

## 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)

<b>Format</b>	<b>Description</b>
DDMMYYYY	Two-digit day, three-letter abbreviation of the month, four-digit year (example: 04JUL2003)
MMDDYYYY	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)
DDDY	Three-digit Julian day, last two digits of year (example: 34999)
YYYYDDD	Four-digit year, three-digit Julian day (example: 1999349)
DDDDYY	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/MM/DD	YY/MM/DD Two-digit year, separator, three-letter abbreviation of the month, separator, two-digit day (example: 99/JUL/20)
DD/MM/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/MM/DD	Four-digit year, separator, three-letter abbreviation of the month, separator, two-digit day (example: 2003/JUL/25)
DD/MM/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/DDD	Last two digits of year, separator, three-digit Julian day (example: 99/349)
DDD/YY	Three-digit Julian day, separator, last two digits of year (example: 349/99)
YYYY/DDD	Four-digit year, separator, three-digit Julian day (example: 1999/349)
DDD/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)

Format	Description
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)
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)

## Formatting Dates and Times

### About this task

Use this procedure to format dates and times in a screen entry or print form.

### Procedure

1. Double-click an existing element.  
The system displays the Element Properties dialog box (Name tab).
2. Click the **Validation** tab.
3. If the minimum allowed field length is longer than the length in the Maximum box, type a new maximum field length in the Maximum box. This value should be the maximum number of digits, including separators, that you expect for the date or time.

**Note:** The value in the Maximum box should be the same as the value in the maximum characters to be displayed box on the screen entry form Display tab or Print form Print tab.

4. Verify that **Date/Time** is selected in the data-type list, indicating that the field value must be a date or a time.

5. From the form format list, select the format for how you want the date or time to be formatted in the screen entry translation object.
6. Verify that the date or time format selected in the file format list is the correct format that is written to the EDI file.

**Note:** A Date/Time format must be designated for Date/Time-type elements in the file format list, even for hidden fields.

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

---

## Formatting Strings

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

### About this task

Use this procedure to format strings in a screen entry or print form.

### Procedure

1. Double-click an existing element.  
The system displays the Element Properties dialog box (Name tab).
2. Click the **Validation** tab.
3. From the data-type list, select **String**. This indicates that the element contains characters.
4. From the form format list, select a predefined syntax token to denote that this element must be formatted as the specified syntax token dictates.

**Note:** 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 (for example, the X syntax token). Most users of Asian or European languages and encoded character sets should use the Free Format (0x01-0xFF).

5. From the file format list, verify that the string format selected is the correct format that is written to the EDI file.

**Note:** A string format must be designated for string-type elements in the file format list, even for hidden fields.

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

---

## Creating and Editing Syntax Tokens for Western European Languages

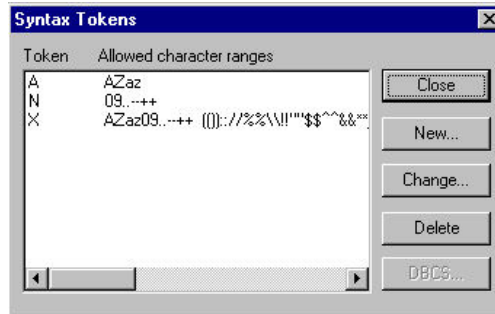
### About this task

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.

---

## 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. The name is typically the element sequence number.
3. To change the description of the element, type the new description in the second box. The description is used to provide a brief explanation of the element that allows you to differentiate it from similar elements.
4. Click the **Validation** tab.
5. To designate the element as mandatory, select the first check box.
6. To change the minimum length of the element, 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.

7. To change the maximum length of the element, type the maximum length in the Maximum box.
8. To change the data-type, select the type of the element:
  - String--alphanumeric element
  - Number--numeric or real element
  - Date/Time--date or time element
  - Bin Data--binary data
  - Bin Length--length of binary data

The binary data types are only available if you select "Binary" on the Special tab of the EDI Segment Properties dialog box. See Modifying Segment Properties for more information.
9. From the form format list, select how the data element is formatted on the form. The choices for this list depend on the type of the element you selected from the data-type list.
10. From the file format list, select how the data element is formatted in the EDI file. The choices for this list depend on the type of the element you selected from the data-type list.
11. To make changes to the display or print options, do one of the following:
  - To make changes to a screen entry form, click the **Display** tab.
  - To make changes to a print form, click the **Print** tab.
12. If the element is in a repeating group or segment and you want to change the summary list options, click the **Summary List** tab.
13. Click **OK** to save the changes to the element.

---

## Storing EDI Code Value Descriptions

Sterling Gentran:Server enables you to store code value descriptions in code storage fields. This feature is useful for elements that contain an EDI code, and for which you want to print the description of the code instead of or in addition to the code itself. You set up a standard rule to direct the system to look up an EDI code value, find the description for that value, and to display the description in another field.

### About this task

You might use this feature if, for example, you have a code list that contains given names and you want the corresponding last names loaded in the descriptive element.

Use this procedure to store an EDI code value description in a code storage field.

### Procedure

1. Double-click the element for which you need to load a code value description. The system displays the Element Properties dialog box (Name tab).
2. Click 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 will be validated against. If the Tables list is empty, you need to load a code table. See Loading a Code List Table from the Standard for more information.
5. 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.
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. For example, if the code you used is "BP," it would be useful to view the description of the code ("Buyer's Part Number"). If you selected element "0350" from the Store Field list, the description for the BP code is mapped to element "0350."
7. Click **OK** to add this standard rule 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.

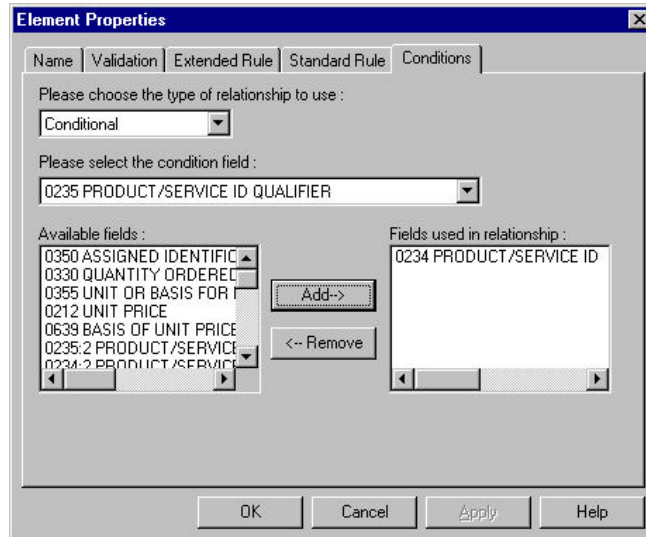
### About this task

**Important:** The system enables you to edit EDI conditional relationships, but if you do, you generates 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.



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.





---

## Chapter 4. Form Completion

---

### Compiling a Translation Object

You use the Compile function after the form is completed and saved. The Compile function compiles the form and creates a translation object. The form that you create using Sterling Gentran:Server is a source form. When that source form is compiled, the result is a compiled translation object. The translation object must be registered with the Sterling Gentran:Server system before you can use it.

#### About this task

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

#### Procedure

1. Select **File > Compile**.

The system displays the Run-Time Translation Object Name dialog.

The system defaults the name of the translation object (.TPL file) with the same filename (up to eight characters long) as you named the form (.STP file).

Preserving the same filename (with different extensions) means that the relationship between the source form and the compiled translation object will remain evident. For example, if the source form name is MWT\_850.STP, the compiled translation object name is MWT\_850.TPL.

2. Navigate to the folder where the compiled translation object is stored, if necessary, and click **Save**.

**Important:** 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 form and creates a translation object. The Compile Errors dialog displays. It indicates errors or warnings that occurred during the compile process.

3. Click **OK** to exit the Compile Errors dialog. Correct any problems and repeat Steps 1 - 3 to recompile the translation object.
4. Select **File > Save**.

The source form is saved with the Compiled on date.

**Note:** This translation object must be registered with the Sterling Gentran:Server system before you can use it.

---

### The Print Function

The Print function available from the File menu enables you to print a customized report in either hard copy or file format. Using the Print option, you can select the following options:

- Branching diagram
- Record Details
- Extended Rules
- Code Lists

---

## Completing the Form

### About this task

See the *IBM Sterling Gentran:Server for Microsoft Windows User Guide* for detailed information on completing this task.

### Procedure

1. After you have compiled your translation object, it must be registered with the Sterling Gentran:Server system before you can use it.
2. Associate this translation object with a trading partner.
3. For a screen entry translation object, create a new document in Document Editor using that translation object. For a print translation object, print a document and verify that the output is correct.

---

## Chapter 5. 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 3. 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 3. 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. <b>Note:</b> 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. <b>Note:</b> 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. <b>Notes:</b> <ul style="list-style-type: none"> <li>• A total of eight fields can be mapped using one Select rule.</li> <li>• 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.</li> </ul>

Table 4. 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 4. 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	Enables you to map from any of the fields within the current document record. <b>Notes:</b> <ul style="list-style-type: none"> <li>• This sub-table can only be accessed from the following map types: Transaction Break/Build, Import, Export, Print, Screen, Group Build, and Interchange Build.</li> <li>• 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.</li> </ul>
Generic envelope segment	Enables you to map from any of the fields within the current envelope information, regardless of the EDI standard used. <b>Note:</b> This function is normally not used unless you are familiar with the generic envelope table structure for a given standard.
Interchange	Enables you to map from any of the fields within the current interchange record. <b>Notes:</b> <ul style="list-style-type: none"> <li>• This sub-table can only be accessed from the following map types: Interchange Break or Build, Group Break, Transaction Break, and Export.</li> <li>• 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.</li> </ul>
Group	Enables you to map from any of the fields within the current group record. <b>Notes:</b> <ul style="list-style-type: none"> <li>• Can only be accessed from the following map types: Group Break or Build, Transaction Break, Export, and Interchange Build.</li> <li>• 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.</li> </ul>

## Using Select in a Form

### Procedure

1. Double-click an existing element, or create a new element.  
The system displays the Element Properties dialog box (Name tab).
2. Click the **Standard Rule** tab.
3. From the standard rule list, choose **Select**.
4. From the table drop-down list, select a table or table and key to use when selecting data.
5. If appropriate, in the Sub Table text box, enter the sub table name.

**Note:** 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.

6. To generate errors if valid table entries are not found, select the compliance error check box. The default is to not generate an error if the Select is not successful.
7. Select the element 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.

8. Choose the element to which you want to map the contents of the map from box. Each element is displayed in this list. A total of eight elements can be mapped using one Select rule.

**Note:** 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.

9. Click **OK** to save your changes.

---

## Using Information from the Partner Definition

This function enables you to load information from the Partner or Internal System Partner Definition into a screen entry or print translation object. 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.

### About this task

Use this procedure to load information from the Partner or Internal System Partner Definition into a screen entry or print translation object.

### Procedure

1. Double-click the element to which the standard rule is applied.  
The system displays the Element Properties dialog box (Name tab).
2. Click the **Standard Rule** tab.
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. Valid selections are:
  - Partner by EDI code
  - Partner by alternate code
  - Partner by partner key
  - Division
5. To generate errors if the lookup fails, select the **compliance error** check box.
6. From the map from list, select the field (Name, EDI Code, Application Code) from which you want the contents mapped.
7. From the map to list, select the element to which the information from the Partner Editor is mapped.
8. Click **OK** to add the standard rule.

---

## 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.

---

## Using Information from Cross-Reference Tables

Cross-reference tables are used to convert your values to your trading partner's values during outbound processing, or your partner's values to your values during inbound processing. Each Partner Profile and the Internal System Partner Profile can have many lookup tables associated with it. 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.

### About this task

To use information from a cross-reference (XREF) table, you must have created that table already in the Sterling Gentran:Server Partner Editor.

Use this procedure to load information from a Cross-Reference (XREF) Table into a screen entry or print translation object.

## Procedure

1. Double-click the element to which the standard rule is applied.  
The system displays the Element Properties dialog box (Name tab).
2. Click the **Standard Rule** tab.
3. From the standard rule list, choose **Select**.
4. From the table list, select the key from which the system performs the look up.  
For Cross-Reference Tables, valid selections are:
  - Partner cross-reference by my item
  - Partner cross-reference by partner
  - Division cross-reference by my item (activates Sub Table box)
  - Division cross-reference by partner item (activates Sub Table box)
5. In the sub table box, type the name of the cross-reference table.
6. To generate errors if the look up fails, select the compliance error check box.
7. From the map from list, select the field (My Item, Partner Item, Description, Text 1, Text 2, Text 3, Text 4) from which the contents are mapped.
8. From the map to list, select the element to which the information from the Partner Editor is mapped.
9. Click **OK** to add the standard rule.



---

## The Update Function

The Update function enables you to update a specific field in a document record, envelope segment, interchange, group, current partner, or document, with the contents of the element.

### Important:

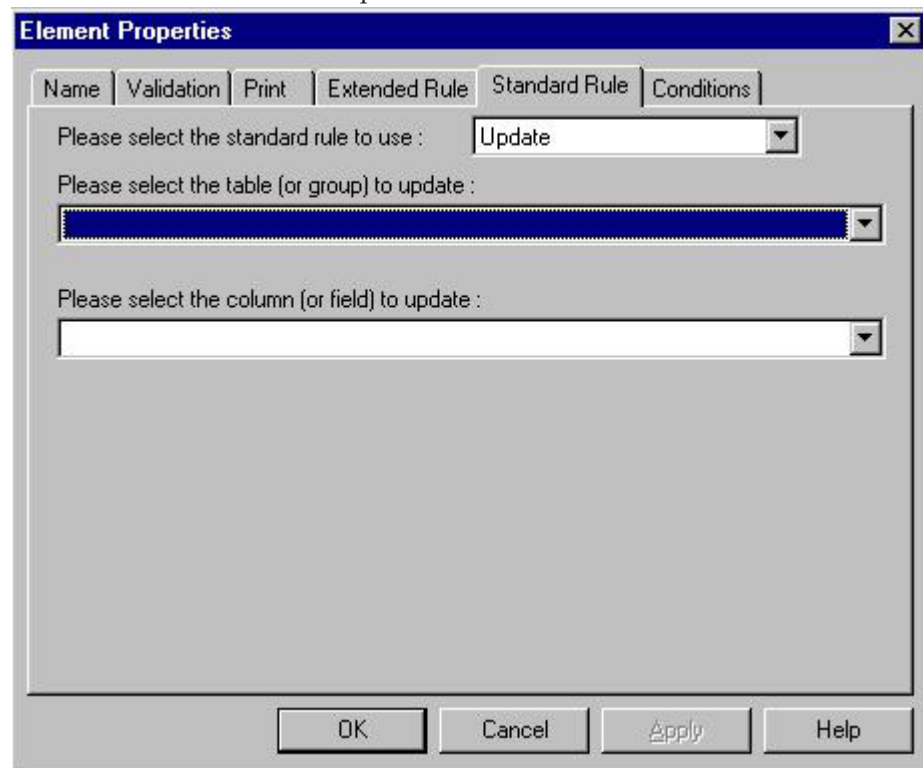
This function updates the internal Sterling Gentran:Server database tables. We recommend that you use this function only if you are sure you really want to update the internal database tables.

Typically, you only use this function if you want to update the document name and reference in the document table. Any other use of this function could have disastrous consequences.

---

## Standard Rule Tab--Update Function

This illustration shows the Update function.



This table describes the parts and functions of the Update Standard Rule.

Part	Function
Please select the table or group to update	Identifies tables or groups available for update.
Please select the column or field to update	Identifies which column or field available for update.

---

## Using Update in a Form

### Procedure

1. Double-click an existing element, or create a new element.  
The system displays the Element Properties dialog box (Name tab).
2. Click the **Standard Rule** tab.
3. From the standard rule list, choose **Update**.
4. Select the table from the table list that the system updates with the contents of the current element.
5. Select the element from the column (or field) list that the system updates with the contents of the current element. The elements that are available in this list depend on the table selected.

---

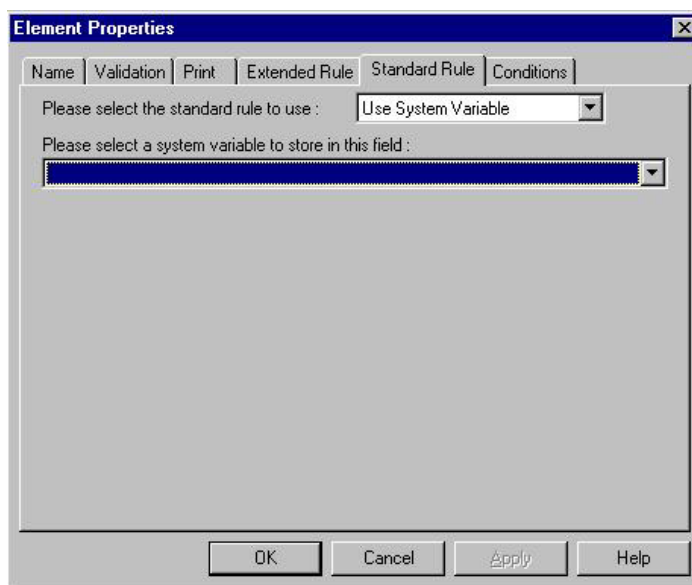
## The Use System Variable Function

The Use System Variable function enables you to set a variable that maps the current date and time to a selected element with a data type of "Date/Time."

---

## Standard Rule Tab--Use System Variable Function

This illustration shows the Use System Variable function.



This table describes the parts and functions of the Use System Variable Standard Rule.

Part	Function
Please select a system variable to store in this field list	Identifies available system variables.

---

## 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.

---

## The Use Constant Function

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

Constants are used in forms to hold information that is needed later in the form in a conditional statement. Typically, EDI qualifiers are typically generated from constants (to indicate a qualifying relationship with another element).

---

## Using Constants in a Form

### About this task

Use this procedure to use a constant in a form.

### Procedure

1. Double-click an existing element.  
The system displays the Element Properties dialog box (Name tab).
2. Click the **Standard Rule** tab.
3. From the standard rule list, select **Use Constant**.
4. Select a constant from the constant list. The constant is mapped to the current element. If the necessary constant is not present in the constant list, you need to create it by using the Map Constants dialog, which you can access by clicking **Edit**.
5. Select an element from the qualifies list if you want to indicate that it qualifies the selected element (establish a qualifying relationship between the two elements).

**Note:** If the qualified element is not generated because of a lack of data, the constant is not moved to the current element.

---

## About Literal Constants and Qualifying Relationships

Literal constants are used by the system as a repository to store information that is used later in the form. A qualifying relationship establishes a relationship between an element and its qualifier.

You create the literal constant and name it. You can "hard code" (enter) a value that is loaded into the element that uses the constant. However, to use a constant (the

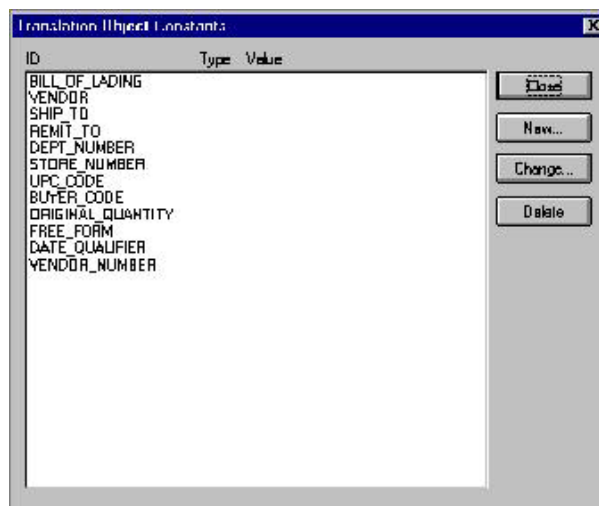
information that is stored in it), you need to use a standard rule, as explained below. Constants are typically used to define qualifying relationships.

A qualifier contains a code that further defines the element. Qualifying relationships are defined using an standard rule.

You can perform the following functions with literal constants and qualifying relationships:

- Defining Literal Constants
- Editing Literal Constants
- Deleting Literal Constants
- Mapping Literal Constants
- Generating Qualifiers

The following shows the Map Constants dialog box.



---

## 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 - 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.

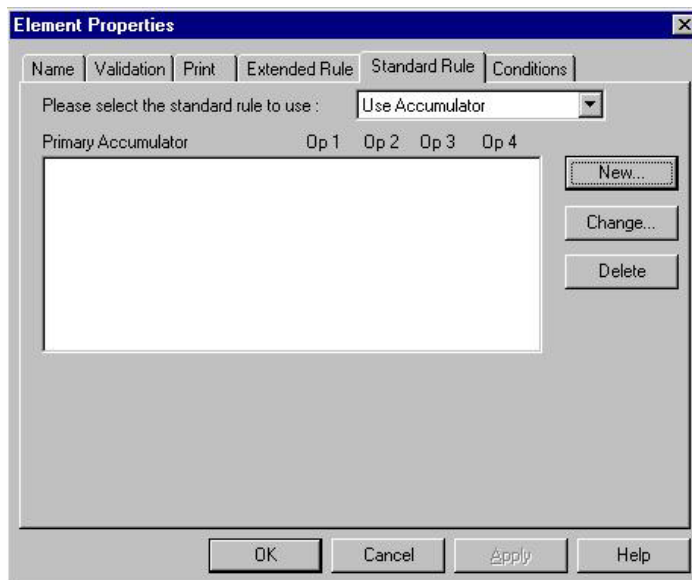


Table 5. Use Accumulator standard rule parts and functions

Part	Function
Primary accumulator	Lists existing accumulators and their associated operations that were created for this element, field, or TFD.

Table 5. Use Accumulator standard rule parts and functions (continued)

Part	Function
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. <b>Note:</b> The selected calculation is deleted without warning.

Table 6. Edit Accumulator Entry dialog box parts and functions

Part	Function
Primary accumulator	Specifies the primary accumulator. <b>Notes:</b> <ul style="list-style-type: none"> <li>• 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.</li> <li>• 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.</li> </ul>
Name	Contains a descriptive alias that enables you to differentiate what the accumulators you create are used for.
First	Specifies the first operation that the system performs. <b>Notes:</b> <ul style="list-style-type: none"> <li>• The First box is active only after you select a Primary Accumulator.</li> <li>• 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.</li> </ul>
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 7. 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).
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). <b>Note:</b> 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).



Table 7. Accumulator operations and functions (continued)

Part	Function
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).

---

## Using an Accumulator in a Form

### Procedure

1. Double-click an existing element, or create a new element.  
The system displays the Element Properties dialog box (Name tab).
2. Click the **Standard Rule** tab.
3. From the standard rule list, select **Use Accumulator**.
4. The Primary Accumulator list contains all existing calculations that were created for this element. To create a new calculation for this entry, click **New** to access the Edit Accumulator Entry dialog box.
5. To change an existing Primary Accumulator entry, highlight a calculation in the Primary Accumulator list and click **Change** to access the Edit Accumulator Entry dialog box to edit the selected calculation.
6. To delete an existing Primary Accumulator entry, highlight a calculation in the Primary Accumulator list and click **Delete**.

**Important:** The selected calculation is deleted without warning.

7. Verify that your entries are accurate, and click **OK**.  
You exit the Element Properties dialog box.

---

## 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 8. Loop Count standard rule parts and functions

Part	Function
group	Contains the loop that you want to count. <b>Note:</b> 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 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 9. 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.

Table 9. Standard rule tab (Use Code) parts and functions (continued)

Part	Function
Raise compliance error	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).  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.
Code description	Contains the element, field, or TFD where you want the description of the code that is used to appear when the selection is made.  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.

Table 10. 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 11. 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 12. Edit Code List Entry dialog box parts and functions

Part	Function
Value	Specifies the actual value of the code.
Description	Contains the code value description. <b>Note:</b> 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.

---

## Using Use Code in a Form

### Procedure

1. Double-click an existing element, or create a new element .  
The system displays the Element Properties dialog box (Name tab).
2. Click the **Standard Rule** tab.
3. From the standard rule list, select **Use Code**.
4. Verify that the code list you want to use is listed in the "Please select the code list to use" list. If it is not, click **Edit** to access the Edit Code List dialog box.  
This enables you to load or create a code table.
5. If the element must contain, for compliance reasons, one of the codes from the specified table, select the compliance error check box.  
For example, if an element 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 element, the system flags it as an error.
6. To store the code description in a field, select an element from the drop-down list where you want the description of the code to appear when the selection is made.

**Note:** 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."

---

## Code List Tables

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.

### Use Code Standard Rule

Once you load or create a code table, you can use a "Use Code" standard rule. For screen entry translation objects, you can use it to select an element from a list that contains a predefined code table. You can also specify whether or not a compliance error is generated if the element does not contain one of the values in the code table. For print translation objects, the Use Code function enables you to extract and print the description of a code and validate the contents of a element against the values in the code table for print translation objects.

## Code list function

The Code List function also enables you to store a code's description in an element/field, create a unique code table, use code values from a code table, and flag whether or not the system will generate an error if a validation against the code table fails. You can import and export code lists and copy and paste code lists between forms.

An element with a Use Code standard rule applied to it allows values to either be checked against or selected from the codes in a specified code table. Codes are typically used to further qualify the element.

### For example

The Product/Service ID (0234) element contains a product code that you expect to be the customer's product code. If you receive a value in that element that is something other than the customer's product code, you want the system to generate an error. So, you first use the code table for the Product/Service ID Qualifier (0235), but specify when you load it that the code table should only include the code "BP." In this code table, the code "BP" is described as a "Buyer's Part Number." Then, you make the 0235 code table exclusive for the Product/Service ID Qualifier (0235) element. Therefore, by using this code with the qualifier for the Product/Service ID element, you are indicating that the Product/Service ID element must contain not just a product code, but product code from the customer.

## Code List Table functions

You can perform the following functions with Code List Tables:

- Defining a Code List
- Modifying a Code List or Code List Entry
- Deleting a Code List or Code List Entry
- Importing a Code List
- Exporting a Code List
- Loading Code Tables from the Standard
- Copying and Pasting Code Lists
- Validating Data Against Code List Tables
- Loading Code Item Descriptions

---

## 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

The EDI 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.

### About this task

Use this procedure 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 (Name tab).
2. Click the **Standard Rule** tab to access standard rule options.
3. From the standard rule list, select **Use Code**.
4. Click **Edit Table**.  
The system displays the Edit Code List dialog box.
5. Click **Load**.  
The system displays the Load Code List dialog box.
6. Do one of the following:
  - To select specific codes only, highlight each code that you want and 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:** We recommend that you only add the codes that you and your trading partners are able to create or accept. Adding all the codes in the code table (using the Add all button) 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.
9. Click **OK** to exit the Element Properties 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.

---

## Loading Code Item Descriptions

Typically, you only load code item descriptions for print translation objects.

### About this task

Use this procedure to load a code item description.

### Procedure

1. Double-click the element for which you need to load a code item description.  
The system displays the Element Properties dialog box.

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

4. 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.
5. 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.  
For example, if the code you used is "BP," it would be useful to view the description of the code (Buyer's Part Number). If you selected element "0350" from the Store Field list, the description for the BP code is mapped to element "0350."
6. Click **OK** to add this standard rule to the element.



---

## 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 13. Data types that supported by extended rules*

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

Table 13. Data types that supported by extended rules (continued)

Data Type	Description	Example
Real	a whole number that may have a decimal component	Declare <b>r</b> as a real number and initialize. <pre>real r;      //Declaration r = 0;      //Initialization of 'r'</pre>
String	contains one or more printable characters	Declare <b>s</b> as a 20-character string and initialize. <pre>string[20] s; //Declaration s = "";      //Initialization of 's'</pre>
Datetime	contains a date or time	Declare <b>d</b> 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 <b>a</b> 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 &lt; 11 do begin     a[i] = 0;     i = i + 1; end  Declare <b>p</b> 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 &lt; 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 <b>ob</b> as an object and initialize. <pre>object ob; //Declaration ob = CreateObject("ADODB.Connection");            //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.

### Levels of extended rules

This table describes the form component level at which you can attach extended rules.

Form Component Level	Attach using the ..
Groups, sub-groups, repeating segments, and repeating elements	Loop Extended Rule tab of the component's associated Properties dialog box.
Elements	Element Level Extended Rule dialog box.

### Scope

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 form. The scope of an extended rule is defined as:

Pre-session extended rules (defined on the Session Rules dialog box) are in scope for every rule in the translation object.

On New, On Store, On Open, and On Delete extended rules (defined on the Loop Extended Rules tab of the appropriate Properties dialog box) are in scope until the conclusion of the corresponding action (creating a new group, storing a group, opening an existing group, and deleting a group). On Open and On Delete extended rules are not available for print forms; only for screen entry.

Field level extended rules are only in scope for the duration of the element.

An extended rule attached to the current form component depends on the type and state of the form 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. An extended rule that is attached to an element is only in scope for the duration of the 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 or in the Pre-Session rule.

The translator builds the data storage area for a form based on the EDI structure. Therefore, extended rules address the form based on the hierarchy of the EDI file.

## Screen entry form processing

The translator processes screen entry forms in the following manner:

Stage	Description
1	Reads the data into the Document Editor and processes all cumulative standard rules (e.g., accumulators, etc.). Does not process any extended or non-cumulative standard rules).
2	Executes group level extended rules in the following situations: <ul style="list-style-type: none"><li>• On_New when a new group is created</li><li>• On_Open when an existing group is edited</li><li>• On_Store when a group is saved</li><li>• On_Delete when a group is deleted</li></ul>
3	Executes field level rules in the following sequence when focus is lost from the field: <ul style="list-style-type: none"><li>• Non-cumulative standard rules</li><li>• Extended rules</li></ul>
4	When the form is saved or recalculated, all standard rules are executed.

## Print form processing

The translator processes print forms in the following manner:

Stage	Description
1	Reads the data and processes all cumulative standard rules (e.g., accumulators). Does not process any extended or non-cumulative standard rules).
2	Executes group level extended rules in the following situations: <ul style="list-style-type: none"><li>• On_New just prior to printing a group</li><li>• On_Store just after printing a group</li></ul>
3	Executes field level rules in the following sequence after the field is printed: <ul style="list-style-type: none"><li>• Non-cumulative standard rules</li><li>• Extended rules</li></ul>

---

## 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 Form Component Extended Rule

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

### Procedure

1. Right click the form component and select **Extended Rules** (or **Extended Rule** if the form component is an element).

- If the form component is an EDI file, repeating group, repeating segment, the system displays the Loop Extended Rules tab of the appropriate Properties dialog box.
- If the form component is an element, the system displays the Extended Rule tab of the Element Properties dialog box.

- 2.

3. If you are not defining the rule for a group, skip this step.

If you are defining the rule for a group, do one of the following:

- To execute the extended rule when you create a new group in the translation object, select **On New**.

- To execute the extended rule when you store a group, select **On Store**.
  - To execute the extended rule when you open an existing group, select **On Open**.
  - To execute the extended rule when you delete a group, select **On Delete**.  
You can define an On New, On Store, On Open, and On Delete rule for a single group
4. In the extended rule list, type the extended rule. (See the Alphabetic Language Reference for more information about rule syntax.)

**Note:** The Extended Rules dialog boxes contain line and character number (within a line) indicators. These indicators enable you to easily debug compile errors.

5. 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.
6. Correct any errors that the system flagged and click **Compile** again. Repeat this process until there are no errors generated.
7. 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 14. 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
<<	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

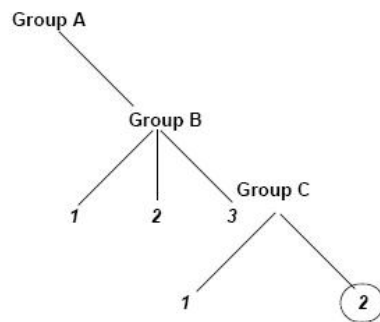
To address a field within a group or create a field within a group in internal storage within the scope of the current hierarchy, the syntax is `$GROUP.#FIELD_NAME`  
`$N1.#0234`

## 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

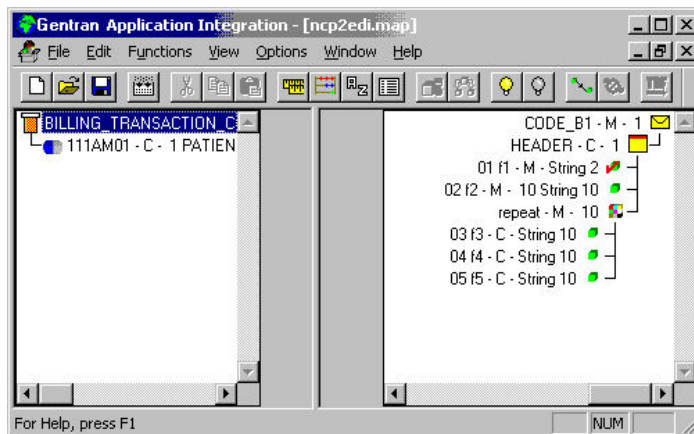
You can access a specific occurrence of a repeating element (for EDI data) and access a specific occurrence of a field within a repeating composite (for EDI data).

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.



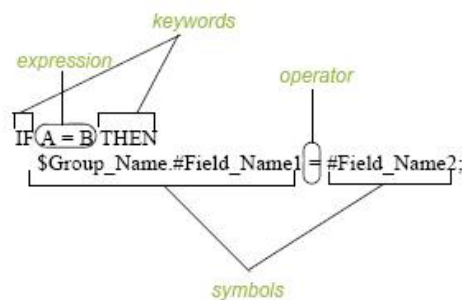
```
string [32]strMsg;

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, ntoa syntax

The atoi function converts strings into integers.

The aton function converts string into real numbers.

The ntoa 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 15. 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

```
error(100, "Number not valid");  
//This raises compliance error 100 with error text "Number not valid" in the  
//translator report.
```

## Compliance codes

Table 16. 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

Table 16. General Messages (continued)

Message Number	Message Type	Messages Generated
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 <b>Note:</b> 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.
10003	Information	Block name <b>Note:</b> 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 <b>Note:</b> 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 <b>Note:</b> 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.

Table 16. General Messages (continued)

Message Number	Message Type	Messages Generated
10017	Information	Map iteration count <b>Note:</b> 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 17. 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
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 18. 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 19. XML Messages

Message Number	Message Type	Message Generated
610	Error	XML particle or group error

Table 19. XML Messages (continued)

Message Number	Message Type	Message Generated
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,#fieldda,15);
messagebox(msg,0);
// Instead of outputting the contents of #fieldda 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 20. 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) <b>Note:</b> 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

Table 20. Format specifiers (continued)

Format Specifier	Description
%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)
%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:\addrum.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(#FIELDA) then
#FIELDB = "EA";
//Return a non-zero value if the condition is true (data is present in
//the specified instance of the FIELDA 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(#FIELDC) then
#FIELDC = "100";
//Populate FIELDC 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(tem_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);
```

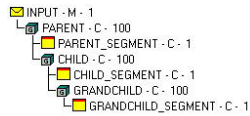
```
//This will populate x with the current loop count / iteration for the  
//outer most parent group, from where the rule is written.  
//The group will be located off of the root level of the map.
```

```
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 21. 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 22. 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:

- interger\_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 Gentrans: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
```

```
    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("^0D^0A",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 message box, 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 23. 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) <b>Note:</b> 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

Table 23. Format specifiers (continued)

Format Specifier	Description
%%	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] PNumChar2;
PNumChar2=mid(#PNumber, 1, 1);
position=strstr("0123456789", PNumChar2);

//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 unread 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
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 24. 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 119 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 25. Window display numbers*

Number	Window_Display	Definition
0	SW_HIDE	Hides the window and activates another window.

Table 25. Window display numbers (continued)

Number	Window_Display	Definition
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 windows.
3	SW_SHOWMAXIMIZED	Activates the window and displays it as a maximized window.
3	SW_MAXIMIZE	Maximizes the window.
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 109 and “update” on page 114 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. Form Customization

---

### About Customizing Forms

On a screen entry form, you can customize the way data is arranged in fields, or the format of numbers, dates, or times. You can even prevent fields in the screen entry form from being displayed when the translation object (compiled form) is used.

After you select the segments, groups, and elements for the print form, you can customize the format of the EDI data as it is generated by Sterling Gentran:Server.

---

### Customizing Field Labels

Field labels identify the contents of the fields included in a screen entry or print form. The system uses the element name from the EDI standard as the default to label a field. You can customize the default field label to be any name you choose.

#### About this task

We recommend that you label every field. If you decide not to label a field, follow the steps below, but delete the existing field label. For example, you might decide not to label the fields of a company address to be entered in a standard address format, because it is obviously an address. In this instance, the name of the company is on the first line; the street address is on the second line; and the city, state, and zip code are on the third line and do not need to be labeled.

Use this procedure to customize a field label.

#### Procedure

1. Double-click the element for which you want to customize the field label.  
The system displays the Element Properties dialog box (Name tab).
2. Do one of the following:
  - To customize a field label on a screen entry form, click the **Display** tab.
  - To customize a field label on a print form, click the **Print** tab.
3. In the "Please enter a label for this field" box, modify the text to customize the label.
4. Click **OK** to save the new label and exit the Element Properties dialog box.
5. Click **Generate Layout** from the main toolbar to refresh the form display.

---

### Hiding Fields

Usually, you want all EDI data on a screen entry or print form to be displayed. However, there may be times when you want to prevent the data in a field from being displayed.

#### About this task

You might want to hide a field when it contains a qualifier that is explained by the label of the associated data field. In this instance, you might set a constant for the field (so that the field always contains the appropriate qualifier without it having

to be entered each time), then hide the field. You also might want to hide a field if it is a mandatory segment that must be kept activated, but does not need to be displayed or printed.

Use this procedure to hide a field in a screen entry or print form.

### Procedure

1. Double-click the element that you want to hide on the form.  
The system displays the Element Properties dialog box (Name tab).
2. Do one of the following:
  - To hide a field on a screen entry form, click the **Display** tab.
  - To hide a field on a print form, click the **Print** tab.
3. Specify that the field should not be included on the form.
4. Click **OK** to exit the Element Properties dialog box.
5. Click the **Generate Layout** icon to refresh the form display.

---

## Setting an Initial Value for a Field

If you typically need the same value in a field each time that field is used, you might want to set an initial value for the field. After you set an initial value, that value displays in the field every time that field is used in the form.

### About this task

For a screen entry form, the user can override the initial value for a field if you want to enter data that is different from the initial value. If you do not want the user to be able to override the field value, then you should use a constant value for the field, instead of an initial value.

If you know that one person at your company is nearly always the person who should be contacted by telephone, you might set an initial value for a Telephone Contact field on an invoice, by setting the telephone contact's name as the initial value. There are occasions when someone else at your company is the contact person, however, so you want to be able to override the initial value in the Telephone Contact field when those exceptions occur.

Use this procedure to set an initial value for a field.

### Procedure

1. Double-click the element for which you want to set an initial value.  
The system displays the Element Properties dialog box (Name tab).
2. Click the **Display** tab.  
The system displays the Display options.
3. In the initial value box, type the value that you want to be the default displayed for this field.
4. Click **OK** to save the initial value for this field and exit the Element Properties dialog box.

---

## Using a Constant Value for a Field

If you always use the same value in a field each time that field is used, you might want to use a constant value for the field so the user does not have to enter the same information each time. After you set the constant value, that value displays in the field every time the field is used.

### About this task

**Note:** You cannot override the constant value for a field. If you want to be able to override the field value, then you should set an initial value for the field, instead of a constant value.

You might set a constant value for a qualifier field on an invoice if you know that you never need to override the value in the field. In this instance, you would probably hide the field, as well.

Use this procedure to use a constant value for an element.

### Procedure

1. Double-click the element for which you want to set a constant value.  
The system displays the Element Properties dialog box (Name tab).
2. Click the **Standard Rule** tab.
3. From the standard rule list, select **Use Constant**
4. From the constant list, select the constant that you want to use. If the constant that you want to use for this field is not in the list, you need to create or edit a constant.
5. Click **OK** and the data that was stored in the selected constant is loaded in the field.

---

## Creating Definitions for a List Box

When Sterling Gentran:Server identifies a group in a transaction set, it creates a separate frame on the Screen Entry form to contain the data within the group. It also creates a list box on the parent frame to allow the user access to the items in the group.

### About this task

To identify the item in the group, at least one field needs to be displayed in the list box. Each set of data can be displayed as a column in the list box. For example, you might have a list box of items on a purchase order. Within the Items list box, you can set up three columns of data:

- Quantity
- Description
- Unit Price

To define the list box columns, you would perform the steps below for the Quantity field, then repeat the steps for the Description and Unit Price fields.

Use this procedure to create a definition for a list box.

## Procedure

1. Double-click the element for which you want to define a list box.  
The system displays the Element Properties dialog box (Name tab).
2. Click the **Display** tab.
3. Select the display option to indicate that this field should be displayed as not-editable on the form.
4. Click the **Summary List** tab.
5. Select **Check here if this field should be displayed** on summary lists check box.
6. From the column list, select the sequence of this column in the list box. For example, if this is to be the first column in the list box, select **1**.
7. In the width box, type the width of this column in number of characters allowed.

**Note:** Make sure that the column width is long enough to accommodate the average length (in characters) of the values expected for this column of data.

8. Click **OK** to save the list box definition and exit the Element Properties dialog box.
9. Repeat **Steps 1** through **8** for each element in the group that contains data you want to define in the list box.
10. To label the columns of data set up in the list box, you must add column headings (static text) to the form.

---

## Creating Help Text for Fields

You can create context-sensitive help text for a field or list box to aid screen entry translation object users in entering the information required (in the Document Editor). The help text displays during data entry when the a user selects a field or an entry in the list box and presses F1.

### About this task

Use this procedure to create help text for a field.

### Procedure

1. Double-click the element for which you want to create Help text.  
The system displays the Element Properties dialog box (Name tab).
2. Select the **Help** tab.
3. In the help text list, type the Help text.

**Note:** Sterling Gentran:Server does not automatically wrap help text when users access help in Document Editor. Therefore, you may want to insert hard returns rather than allowing the list to scroll to the right.

4. Click **OK** to save the help text and exit the Element Properties dialog box.

---

## Creating Help Text for List Boxes

List boxes contain information from selected elements in a repeating group. The information from each element displays as a column in the list box.



## About this task

Use this procedure to create Help text for a list box.

### Procedure

1. Right-click the group for which you want to create help text and select **Properties**.  
The system displays the Group Properties dialog box (Name tab).
2. Click the **Display** tab.
3. In the Help Text list, type the help text.
4. Click **OK** to save the Help text and exit the Group Properties dialog box.

---

## Creating Help Text for Frames

The help text displays during data entry when the a user clicks "Frame?" or selects "Frame Help" from the Document Editor Help menu, when the frame is currently in focus.

## About this task

Use this procedure to create Help text for a frame.

### Procedure

1. Right-click the group for which you want to create help text and select **Properties**.  
The system displays either the File Format Properties dialog box or the Group Properties dialog box, depending on the level of the group.
2. Click the **Display** tab.
3. In the Frame Help Text list, type the help text. For groups, the Frame Help Text list is only be active if the Max Usage (found on the Group Properties dialog box, Looping tab) is greater than "1."
4. Click **OK** to save the help text and exit the Properties dialog box.

---

## Formatting Entry Fields

An entry field is a field that can be edited. When you create a form, Sterling Gentran:Server makes every field an entry field by default. For entry fields, you can change the field type to Display (the field is display-only) or Hidden (the field is not displayed at all).

## About this task

Use this procedure to change a display-only or hidden field to an entry field.

### Procedure

1. Double-click the element for which you want to define a list box.  
The system displays the Element Properties dialog box (Name tab).
2. Click the **Display** tab.
3. Select the editable option to indicate that this field can be edited.
4. Click **OK** to exit the Element Properties dialog box.
5. Click **Generate Layout** from the main toolbar to refresh the display.

---

## Formatting Display-Only (Non-Editable) Fields

When you create a form, Sterling Gentran:Server makes every field an entry field by default. You can change the default to Display if you want a field to be display-only (non-editable). Data can be viewed but not edited in display-only fields.

### About this task

Use this procedure to format a display-only field.

### Procedure

1. Double-click the element for which you want to define a list box.  
The system displays the Element Properties dialog box (Name tab).
2. Click the **Display** tab.
3. Select the display option to indicate that this field should be displayed as not-editable on the form.
4. Click **OK** to exit the Element Properties dialog box.
5. Click **Generate Layout** from the main toolbar to refresh the display.

---

## Chapter 8. Form Formatting

---

### Selecting One or More Fields or Field Labels

You must know how to select one or more items (fields and field labels) in a form to be able to move, resize, align, or change the spacing between them.

#### About this task

The last item you select is outlined by diagonal lines. This item is called the reference. All other items you selected are now outlined by a solid line.

When you arrange a group of items, the reference item is significant because the alignment, size, and spacing between all other items in the group is relative to the alignment, size, and spacing of the reference item.

Use this procedure to select a field or a field label.

#### Procedure

1. In the Layout Window, click anywhere on the field or field label.

The Lock Labels icon on the main toolbar enables you to specify that when you click a field (in the Layout Window), its corresponding label is selected at the same time.

The field or field label is outlined by diagonal lines. The selected field also displays handles, which mark the corners and sides.

2. If required, press **CTRL** and click another field or field label.
3. Repeat these steps for all other fields and field labels in the form you want to select.

#### Note:

- When you select two or more items, make sure the item you want as the reference is the last item you select.
- If you selected any fields by mistake, press **CTRL** and click the item you want to deselect.
- If the item you deselect is the reference, the item that was selected immediately prior to the last item (the reference) becomes the new reference.

---

## Grouping Fields and Field Labels

You can group fields and field labels that are in the same area of a form.

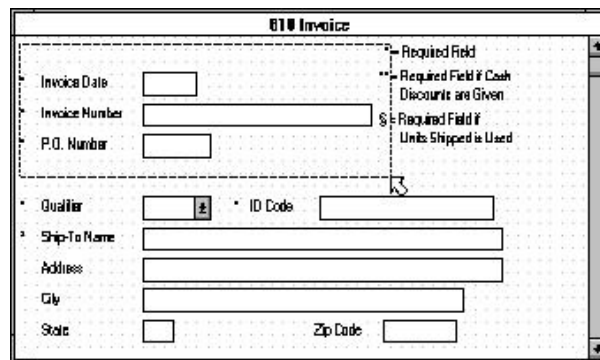
### About this task

Use this procedure to group fields or field labels.

### Procedure

1. In the Layout Window, click and drag the cursor across the form items that you want to group.

As you drag the mouse, the grouped area is designated by dotted lines similar to the following illustration.



2. Release the mouse button.

All items completely contained within the selection area are selected. The last item in the group is the reference. All the selected items are now a group, and can be moved together, as well as resized or spaced relative to the reference.

---

## About Resizing Fields

When you generate a form, each field is automatically sized to the length specified in the Display Length field on the Display Properties dialog box. If you change the display length of a field after the form is generated for the first time, you must resize that field in the Layout Window.

You can resize a field in one of these two ways:

- Use the Size to Length function.
- Use the mouse to reshape the field manually.

In addition, you can resize groups of fields or field labels so that they are all the same size.

**Note:** Resizing the field graphically maintains the integrity of the form.

---

## Using Size to Length

You can resize one or more fields according to the display length specified on the Display Properties dialog box.

## About this task

Use this procedure to resize a field or group of fields with the Size to Length function.

### Procedure

1. In the Layout Window, select the form fields that you want to resize. See Grouping Fields and Field Labels for more information.
2. Select **Display > Size To Length** to resize according to the display length specified on the Display Properties dialog box for each selected item.

---

## Resizing a Field Manually

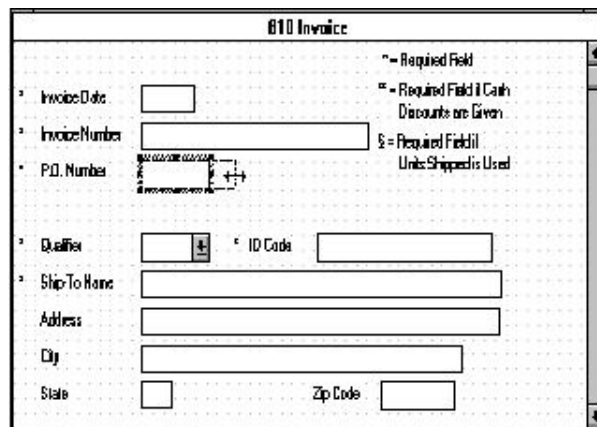
Use this procedure to manually resize one field.

### About this task

#### Procedure

1. In the Layout Window, select a form field.
2. Click and drag one of the handles.

The following diagram shows a field being resized.



---

## Resizing Groups of Fields Manually

You can resize several fields at the same time if you want them all to be the same size. For example, you might want to resize a group of fields that are similar in size to give the form a more consistent appearance.

### About this task

Use this procedure to resize a group of fields manually.

#### Procedure

1. In the Layout Window, select the form fields you want to resize.

**Note:** Make sure that the field that is to be used as the reference for resizing all other selected fields is selected last.

2. Select **Display > Size Equally**.

The system equally resizes all selected items. The reference is used as the basis for determining the size of all other selected items.

---

## Changing Grid Settings

Sterling Gentran:Server provides a grid, indicated by the dots in the background of the form, that you can use to assist you in formatting the layout of a screen entry or print form. The dots on the grid form horizontal and vertical lines that guide you in aligning and spacing fields and field labels.

### About this task

The dots on the grid are evenly spaced, with the spacing between them measured in pixels. The grid is not printed (for print forms) – it is only displayed in the background of the template to aid you in arranging fields and field labels.

Sometimes it is useful to have items snap to the grid when you move them. With the "Snap controls to the alignment grid" feature turned on, as you move, align, or space items, they snap to the lines of the grid. This feature can help you align items with one another or evenly space them.

Use this procedure to change the grid settings.

### Procedure

1. Select **Options > Preferences**.  
The system displays the Preferences dialog box.
2. Click the **Layout** tab.  
The system displays the layout options.
3. To display the grid, click **Show alignment grid**.
4. To snap the fields and field labels to the grid, select **Snap controls to the alignment grid**.

**Note:** If you are using alignment lines, you should not use the "Snap controls to the alignment grid" function, as this hinders the effectiveness of the alignment lines.

5. To change the spacing between the dots on the grid, type the number of pixels in the Grid spacing box. The default setting is five pixels. Increasing the number of pixels increases the spacing between the dots; decreasing the number of pixels decreases the spacing between the dots.
6. Click **OK** to exit the Preferences dialog box.

---

## Using Alignment Lines

Sterling Gentran:Server enables you to create alignment lines (lines that guide you in aligning the components of the Layout Window).

### About this task

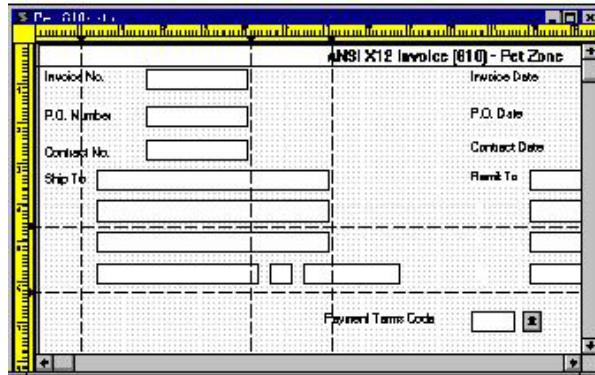
Use the procedure to display, move, or remove alignment lines.

### Procedure

1. In the Layout Window, double-click the vertical or horizontal ruler just below the graduation where you want the alignment line to be located.

**Note:** If you are using alignment lines, you should not use the Snap Controls To Grid function (on the Grid Settings dialog box), as this hinders the effectiveness of the alignment lines.

The system displays the alignment line.



2. To move the alignment line to another position, click and drag the black triangle that connects the alignment line to the ruler.
3. To remove an alignment line, double-click the black triangle that connects the alignment line to the ruler.

---

## Moving Fields and Field Labels

You can move fields or field labels from their default positions to other locations in a form. You can move either individual items or groups of items.

### About this task

If you intend to resize fields, we recommend that you do so before moving them, since the arrangement of the fields might depend on the size of the fields. For example, you might be able to arrange more fields in a row if the fields are smaller. Therefore, you should resize the fields before you arrange them, so you can accurately determine how many fields can fit in a row.

Use this procedure to move a field or field label.

### Procedure

In the Layout Window, click the field that you want to move and drag it to the new location in the form.

#### Note:

- You can use Lock Label on the main toolbar to move a label with its associated field.
- If you are moving a group of selected items, all items in the group stay in position relative to each other as they are moved from one location to the next.

---

## Aligning Fields and Field Labels

When fields and field labels are created, they are automatically left-aligned in the screen entry or print form, with one field per line. However, you can align particular fields and field labels with one another in different ways to change the way they are arranged in the form.

## About this task

If you intend to resize or move fields or field labels, we recommend that you do so before you align them. If you align fields first, then move or resize them, the fields might move out of alignment.

You use Align Controls to specify how you want items on your form to appear:

- **Top, Bottom, Left, Right:** Aligns all selected items with the reference item.
- **Horizontal Centre, Vertical Centre:** Aligns all selected items in the frame.

Use this procedure to align fields and field labels.

## Procedure

1. Select two or more fields that you want to align in a form.

**Note:** Make sure that the field or field label to be used as the reference for aligning all other selected items is selected last.

2. Select **Display > Align Controls**.
3. From the submenu, select how you want the selected items to be aligned.

**Note:** Use caution in selecting the alignment method to ensure that items do not overlay each other. Use Top, Bottom, or Horizontal Centre alignment for aligning a horizontal row of items; use Left, Right, or Vertical Centre alignment for aligning a vertical column of items.

---

## Spacing Fields and Field Labels

When fields and field labels are resized and moved, the spacing between them can become uneven. You can change the spacing between items to give the form a more consistent appearance.

## About this task

If you intend to resize or move fields or field labels, we recommend that you do so before you space them. If you space fields first, then move or resize them, the spacing between the fields might change.

You use the Space Evenly cascading menu to specify how far apart fields display vertically or horizontally on the form:

- **Across:** Evenly spaces all selected items with the next selected item to the left or right (horizontal spacing).
- **Down:** Evenly spaces all selected items with the next selected item above or below (vertical spacing).

Use this procedure to evenly space several fields and field labels.

## Procedure

1. Select three or more items that you want to be evenly spaced in a form.

**Note:** Make sure that the field or field label to be used as the reference for evenly spacing all other selected items is selected last.

2. Select **Display > Space Evenly**.
3. From the submenu, select the type of spacing you want for the selected items.



---

## Resizing and Positioning a Frame

You can define how a frame is sized and positioned in the Document Editor by resizing and positioning that frame in the Forms Integration subsystem. When a form is created, the default location for frames is the upper left corner of the Layout Window. When a screen entry translation object is opened in Document Editor, its frame is sized and positioned the same as it was when the source form was last compiled.

### About this task

For example: If you want the frame to be centered on the user's desktop, then you must center the frame in the Layout Window in the Forms Integration subsystem before compiling it.

**Important:** When you resize the frames of a screen entry form, you must resize based on the resolution of the target monitor (i.e., the monitor on which the end user views the translation object). For example, if you size the frames of a screen entry form on a Super VGA monitor, the compiled translation object may not fit in the display area of a VGA monitor.

Use this procedure to resize and position a frame.

### Procedure

1. In the Layout Window, point to the frame border. When the pointer becomes a double-headed arrow, drag the border to resize.
2. To position the frame in the Layout Window, click and drag the frame title to the new position in the Layout Window.

---

## Tips for Resizing Drop-Down Combo Box

A drop-down combo box consists of a text box and a drop-down list. When the combo box is closed, the text box and the drop-down arrow are displayed. When the combo box is opened, the available field values (and a scroll bar, if there are more values than can fit in the drop-down list) are displayed. You can resize a drop-down combo box in a screen entry form to make the size appropriate for the field values contained in the list.

Please remember the following tips about drop-down combo boxes:

- When you change the width of the combo box, the width of the text box changes. (The width is the same whether the drop-down combo box is opened or closed.)
- When you change the height of the combo box, the height of the drop-down list is changed. When closed, the drop-down combo box is long enough to contain one item. When opened, the drop-down combo box is long enough to contain several items.
- The width of the drop-down combo box should be appropriate to the width of the values available in the list. You should make the width a few spaces wider than the longest value available. For example, if the values available are two-character codes, the combo box does not need to be twenty characters wide. However, if the values in the list consist of several words or a long string of characters, then the combo box should be wide enough for users to read the longest field value.

- The height of the drop-down combo box should be appropriate to the number of values available in the list. For example, if there are just four values available, you might make the drop-down combo box long enough to contain all of the values, so that users do not have to scroll through a short list. However, if there are one hundred values available, you might make the drop-down combo box long enough to view up to eight values at once.

**Note:** The scroll bar is automatically added to the drop-down combo box when the number of available values exceeds the vertical space provided in the combo box.

---

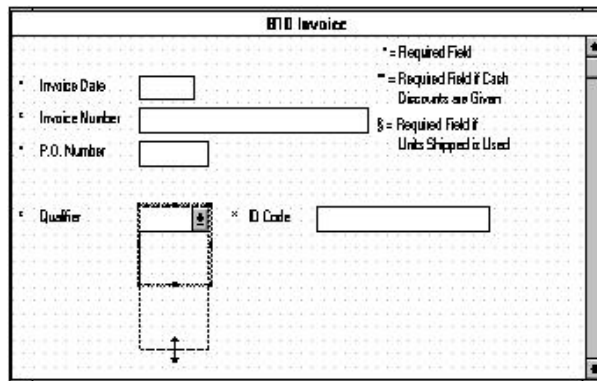
## Resizing Drop-Down Combo Boxes

### About this task

Use this procedure to resize a drop-down combo box.

### Procedure

1. In the Layout Window, click the drop-down combo box you want to resize in the form.
2. Drag a handle of the combo box.




---

## Setting Tab Sequences

The tab sequence is the order in which the cursor moves from field to field in a screen entry form when a user presses the Tab key. When you create a screen entry form, the system automatically positions the fields and sets the tab sequence for those fields. However, you usually want to modify the tab sequence of the fields to meet your screen entry requirements.

### About this task

**Note:** You can set tab sequences for screen entry forms only.

Use this procedure to set the tab sequence of fields in a screen entry form.

### Procedure

1. Select **Display > Set Tab Order**.  
Numbers are displayed in the fields in the form to indicate the current tab sequence.

The screenshot shows a form titled "EDI Invoice" with the following fields and legend:

- Invoice Date:  1
- Invoice Number:  2
- P.O. Number:  3
- Qualifier:  4
- Ship-To Name:  5
- Address:  6
- City:  7
- State:  8
- ID Code:  9
- Zip Code:  10

Legend:

- \* = Required Field
- \*\* = Required Field if Cash Discounts are Given
- \*\* = Required Field if Units Shipped is Used

- Click the fields in the order in which you want the tab sequence to be set. For example, if you want a field that is currently set as third in the tab sequence to become first in the tab sequence, then click that field first. The field that was first in the tab sequence is now the second, the second is now the third, etc. After you have clicked all the fields, the tab sequence numbers are no longer displayed in the fields.

**Note:** If you want to change the tab sequence for some of the fields without having to click them all, then select Set Tab Order from the Display menu when you have finished setting the tab sequence for the fields you want to change. The tab sequence numbers are no longer be displayed.

---

## Adding and Positioning Static Text

Static text is text that is always included in the screen entry or print form. Examples of static text include legends and column headings. Field labels, which usually accompany fields to identify the contents of the fields, are not considered static text.

### About this task

Use this procedure to add and position static text to a screen entry or print form.

### Procedure

- Click **Display > Add Text**.  
Text is added to the current frame, displayed in the upper left corner. If the translation object is larger than the frame (there is a scroll bar on the right side of the frame), the static text displays in the upper left corner of the part of the translation object that displays, not at the top of the translation object.
- Type the static text in the new selected text field.  
The system displays the Text Properties dialog box.
- Click **OK** to add the static text.
- To move the static text, click and drag it to a new location in the frame.

---

## Modifying Frame Titles

The system automatically titles the frames in a screen entry form when the form is created, based on the name of the file, segments, or groups. You can modify the frame title of any repeating group (EDI file, segment, or group) in a screen entry form. For example, if you use several purchase order forms, each for a different trading partner, you might give each purchase order form a slightly different title to distinguish one form from the others.

## About this task

Use this procedure to modify a frame title.

### Procedure

1. Highlight the form component (EDI file, segment, or group) of the frame title you want to modify.
2. Select **Edit > Properties**.
3. Click the **Display** tab.
4. In the Frame Caption box, type the new frame title.
5. Click **OK** to display the new frame title in the appropriate frame.

---

## Modifying List Box Names

When Sterling Gentran:Server identifies a group in a transaction set, it creates a separate frame to contain the data within the group. It also creates a list box in the parent frame to allow the screen entry translation object user access to the items in the group.

### About this task

The system automatically sets the names for list boxes in a form when the form is created, based on the name of the segment or group. You can modify the names of the list boxes in screen entry forms. For example, you might want to modify a list box name to give it a name that is more meaningful to you than the default name that the system gives it.

Use this procedure to modify a list box name.

### Procedure

1. Highlight the form component (segment or group) containing the list box name you want to modify.
2. Select **Edit > Properties**.
3. Click the **Display** tab.
4. In the List Caption box, type the list box name.
5. Click **OK** to display the new list box name in the form.

---

## Adding Column Headings to List Boxes

A single list box in a form may contain two or more columns of data, depending on which fields have been designated list box fields. Although the system automatically generates field labels for list boxes, it does not automatically generate column headings for them. If you want the columns in a list box to be labeled, you must add the column headings by using static text.

### About this task

Use this procedure to add column headings to a list box.

### Procedure

1. In the Layout Window, select **Display > Add Text**.  
Text is added to the current frame, displayed in the upper left corner. If the translation object is larger than the frame (there is a scroll bar on the right side

of the frame), the static text displays in the upper left corner of the part of the translation object displays, not at the top of the translation object.

2. Type the static text in the new selected text field.  
The system displays the Text Properties dialog box.
3. Click **OK** to add the static text.
4. To move the column heading, click and drag the column heading to a new location in the frame.

User Exits



---

## Chapter 9. User Exits

---

### 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 26. 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.

---

## 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.

---

## 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

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. Open the map in which you want to apply a user exit.



2. Determine where the user exit will be located in the map (e.g., 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.

3. Construct the user exit.
  - See the Extended Rules section for more information about creating extended rules for various form components.
  - See the Alphabetic Language Reference section for more information about the createobject, deleteobject, and getiid commands.



---

## Chapter 10. Translator Command Line Interface

---

### About the Command Line Interface

This topic describes how to use translator command line interface to invoke the services of TXDE.EXE (print and screen entry translator).

**Note:** This interface is currently used by all applications that invoke the services of the TXDE.EXE translator.

### Command Line Syntax

The syntax of the command line is:

```
txde [-r] [-n]
      [-a server]
      [-e eventid]
      [-p filename]
      [-d [dockey] ]
      [-u param -u param ...]
```

### Syntax parameters

This table describes the command line syntax parameters.

Parameter	Description
-r	Read-only mode
-n	Do not move document to InDrawer after print or export
-a	Server (name of audit server to which TXDE.EXE connects)
-e	EventID (identifier of an audit event to which this translator belongs)
-P	Print mode <b>Subparameter:</b> filename The "filename" subparameter is the path to file containing document keys (one per line).
-d	Screen entry mode <b>Subparameter:</b> dockey The "dockey" subparameters is the key for the document to edit, if you are modifying an existing document.
-u	User-defined parameters accessible from extended rules <b>Subparameter:</b> param The "param" subparameter is the parameter value.



---

## Chapter 11. 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 27. 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 27. 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 27. 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 27. 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 27. 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 27. 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 28. 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 28. 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 28. 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 28. 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.



---

## 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 2012. Portions of this code are derived from IBM Corp. Sample Programs. © Copyright IBM Corp. 2012.

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

#### **Trademarks**

IBM, the IBM logo, and [ibm.com](http://www.ibm.com)<sup>®</sup> 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.





Product Number: 5725-D09

Printed in USA