Gentran Integration Suite™

Map Editor Guide

Version 4.2

Sterling Commerce
An IBM Company

Contents

Chapter 2 Mapping EDI Documents   67

Chapter 7 Mapping XML Documents   129

Chapter 8 Using Standard Rules   155

Chapter 9 Using Extended Rules 187

Gentran Integration Suite Map Editor Guide

Chapter 10 Extended Rules Library 253

Chapter 11 Using User Exits   267

Chapter 12 Managing Maps   273

Appendix A Error Messages   281

Appendix B Map Editor Properties   295

Appendix E Hierarchical Levels (HL) in Maps   383

Appendix F Map Conversion   399

Appendix G COBOL Copybook Conversion for Use with Map Editor   409

Chapter 1

Map Editor Basics

The *Map Editor Guide* explains how to use the Gentran Integration Suite® Map Editor to map the data formats supported by Gentran Integration Suite.

This documentation assumes knowledge of:

✦ UNIX® operating system

✦ Windows® operating system

✦ Data formats

✦ Data mapping concepts

You can use the Gentran Integration Suite Map Editor to map documents in the following data formats: Electronic Data Interchange (EDI), positional, variable-length-delimited, Japanese Center for Informatization of Industry (CII), Extensible Markup Language (XML), and Simple Query Language (SQL).

This section covers the following topics:

✦ About Gentran Integration Suite Mapping

✦ About the Data Formats in Gentran Integration Suite

✦ About the Map Editor

✦ Mapping Preparation and Analysis

✦ Customizing the Map Editor Display

✦ Creating a Map

✦ Opening a Map

✦ Using Data Definition Format and Integrator File Definition Files

✦ Defining Map Details

✦ Adding Map Components

✦ Defining Map Component Properties

✦ Using Copy, Cut, and Paste

✦ Splitting Groups

✦ Promoting Groups

✦ Finding Information in the Map

✦   Formatting Data in Fields

✦   Completing a Map

✦   Testing a Map

About Gentran Integration Suite Mapping

The following concepts are essential to an understanding of mapping in Gentran Integration Suite.

For more information, see:

Translation

When you have an electronic document in one format and the document is needed in a different format, you must translate the data in your document from one format to another. In Gentran Integration Suite, you use the Translation service to translate data.

Mapping

To translate data from one format to another, you must specify how the data in one format relates to data in another format.

To relate one format to another for the translator, you must define a set of instructions in the Map Editor. These instructions indicate the relationship between the two formats.

Source Map

In the Map Editor, you specify mapping instructions for translation in a *source map* (a file with the extension .mxl—this is the default extension, when the map is saved as an XML-formatted file—or .map). The source map displays mapping instructions graphically. The data format that you are translating from is represented in a visual layout on the left side. The data format that you are translating to is represented in a visual layout on the right side.

Note:	To save your source map as an XML file (.mxl file extension), you must have the Microsoft XML Core Services (MSXML) 4.0 installed on the same computer as Map Editor. If you do not have Microsoft XML Core Services (MSXML) 4.0 installed, you cannot save source maps as .mxl files and must use the .map extension.

Tip:	One benefit of using the .mxl format is that you could potentially manipulate it using other text editors.

The data formats that you can map using the Map Editor are:

✦ Electronic Data Interchange (EDI)

✦ Positional

✦ Variable-length-delimited

✦ Japanese Center for Informatization of Industry (CII)

✦ Extensible Markup Language (XML)

✦ Structured Query Language (SQL)

The translator cannot use a source map to translate data. The source map must be compiled into a translation object, which the translator can use to translate data.

Translation Object

To use the instructions in a source map, you must compile the map. A compiled map has a different extension, .txo, and is called a *translation object*. It provides instructions for translating one format to another in a way that can be interpreted by a translator. The translator does the work of converting a file from one format to another.

Sterling Integrator Map Type

The *Sterling Integrator map type* is a type of map specific to Gentran Integration Suite. Use this type of map to translate documents in Gentran Integration Suite. Other Sterling Commerce products use other types of maps, which can be opened by Gentran Integration Suite and then converted to Gentran Integration Suite maps. See *Defining Map Details* on page 35 for more information about designating a map as the Sterling Integrator type.

The following figure represents a Gentran Integration Suite map. Each map has two sides. Each side represents a data format: the input side represents the data format you are

translating from, and the output side represents the data format you are translating to. Each part of a data format is represented by specific map components.

Input data format layout   Output data format layout
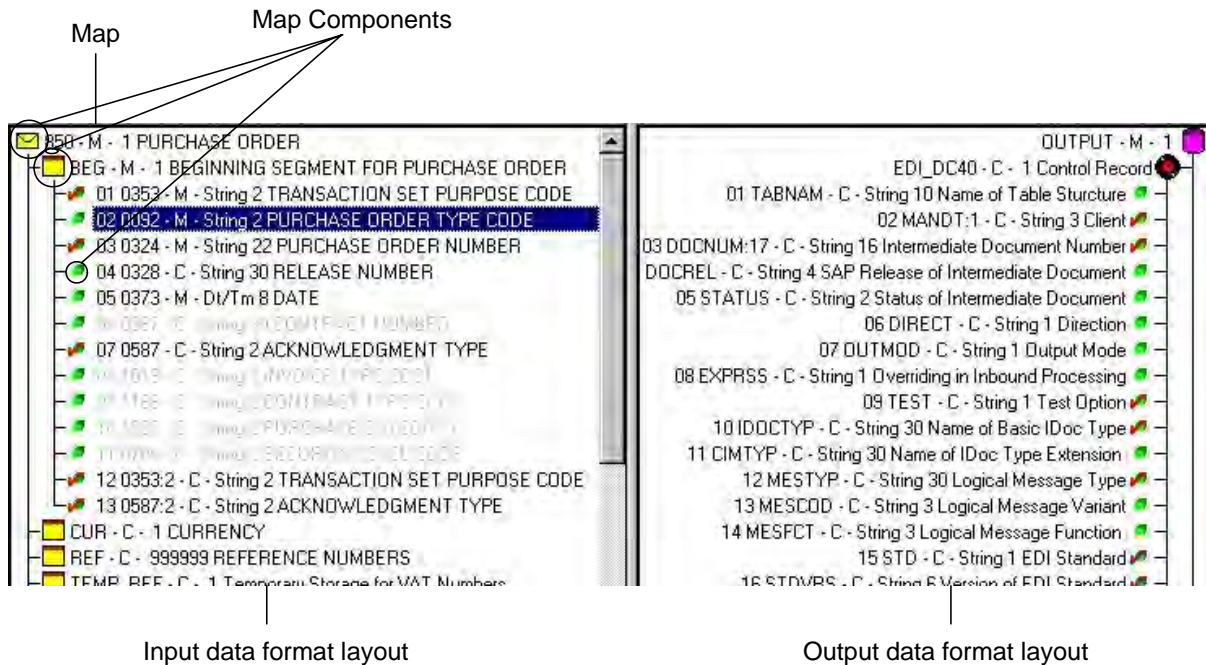
XML Encoder Object

The *XML encoder object* is a compiled map that translates positional, variable-length-delimited, CII, and EDI data formats into XML. It has the extension .ltx. To create an XML encoder object, you select Compile XML Encoder from either the input or output side of an EDI or positional file.

About the Data Formats in Gentran Integration Suite

The Map Editor enables you to map several data formats.

For more information, see:

- ✦ *EDI Data Format* on page 19
- ✦ *Positional Data Format* on page 19
- ✦ *Variable-Length-Delimited Data Format* on page 19
- ✦ *CII Data Format* on page 19
- ✦ *XML Data Format* on page 20
- ✦ *SQL Data Format* on page 20

EDI Data Format

The Map Editor generates an Electronic Data Interchange (EDI) file for you, according to the standard (agency), version, and transaction set you select. The format includes all the groups, segments, composites, and elements that are defined by the standards agency for the version of the document you selected.

You can modify the Map Editor-generated EDI file by changing the properties of the map components and by using the Promote, Split, Copy, Cut, and Paste functions.

If you want to use a specialized version of an EDI standard that is not available in Gentran Integration Suite, you can either load an EDI file definition or define the EDI file yourself. For more information about loading an EDI file definition, see *Creating an EDI Layout from an EDI Standard* on page 70.

Whether the Map Editor generates the EDI file or you define it, the EDI map components that you use depend on the type of map you are creating. The components can include the standard, version, and transaction set (document) selected, and the groups, segments, composites, and elements that your organization requires.

The EDI file must contain all the data that you expect to receive from your trading partner (if the map is inbound) or must send to your trading partner (if the map is outbound). The file must contain this data so that Gentran Integration Suite can process it accurately.

For more information about customizing the properties of EDI map components, see Chapter 2, *Mapping EDI Documents*.

Positional Data Format

The positional data format defines characteristics of a file such as delimiters or record length. A positional data format is also referred to as a fixed format or an application file.

For more information about the positional data format, see Chapter 3, *Mapping Positional Documents*.

Variable-Length-Delimited Data Format

Gentran Integration Suite enables you to map variable-length-delimited files, such as the comma-separated text file (.csv).

For more information about defining a variable-length-delimited data format, see Chapter 4, *Mapping Variable-Length-Delimited Documents*.

CII Data Format

The CII data format provides the Japanese syntax definition for EDI messages. The CII implementation in Gentran Integration Suite is based on the CII Syntax Rule, available in both Japanese and English. The CII Syntax Rule specifies details such as looping structures and data types, but it does not include standard message types. Message types are provided by industry groups. Gentran Integration Suite provides a number of these standard message types in the EDI Standards, which you can download.

For more information about the CII data format, see Chapter 5, *Mapping CII Documents*.

XML Data Format

In the XML data format, the Gentran Integration Suite XML implementation conforms to the rules of the XML language 1.0 specification, as published by the World Wide Web Consortium (W3C), with some exceptions. The W3C base 1.0 XML specification enables you to perform the following tasks with XML:

✦ Specify the number of times that a group can repeat.

✦ Specify the number of times an element in a mixed group can repeat.

✦ Repeat an element (with a structure different from the structure of the original element) in a different part of the document. For example, for an invoice, you can define an address element twice—once under Ship To and once under Bill To.

For more flexibility, Gentran Integration Suite diverges from the W3C base 1.0 XML specification. Gentran Integration Suite supports the W3C specification with the following exceptions:

✦ The XML document must meet the well-formed document criteria specified for XML. If the document is not well-formed, Gentran Integration Suite generates an error message.

✦ Gentran Integration Suite supports external parameter entities but does *not* support mapping of external entities, notations, elements of type ANY, comments, conditional sections, internal DTDs, unparsed entities (non-XML data), or processing instructions. In most cases, Gentran Integration Suite disregards the items listed as not supported.

✦ Gentran Integration Suite can read and write UTF-8 and UTF-16 encoded files.

For more information about the XML data format, see Chapter 7, *Mapping XML Documents*.

SQL Data Format

The Structured Query Language (SQL) data format enables you to create a map directly from a database schema, which saves time and ensures that the map is synchronized with the most current version of the database. You can also specify several data sources so that Gentran Integration Suite can query or update multiple databases during translation.

The Map Editor enables you to:

✦ Specify data sources that Gentran Integration Suite uses to query for data or update data in multiple databases in a single translation session.

✦ Add data sources and then test the connection and edit the connection string.

✦ View a model of the database schema, including lists of tables and columns.

✦ Generate fields directly from your database schema.

✦ Check the consistency of your database.

For more information about the SQL data format, see Chapter 6, *Mapping SQL Documents*.

About the Map Editor

The Map Editor is a stand-alone Windows program that you download from Gentran Integration Suite. The Map Editor enables you to create maps (.mxl or .map) and compile them into either translation objects (.txo.) or XML encoder objects (.ltx).

After you have created and compiled maps, you check them in to Gentran Integration Suite. For more information about checking maps in to Gentran Integration Suite, see Chapter 12, *Managing Maps*.

> **Note:** Fields with colored field names are required. If you skip a required field, a message prompts you to supply the missing information.

For more information, see:

- ✦ *Removing the Map Editor* on page 22
- ✦ *Navigating in the Map Editor* on page 22

Installing the Map Editor

You install the Map Editor after Gentran Integration Suite has been installed because you download the Map Editor from within Gentran Integration Suite.

Before you install and run the Map Editor, meet the requirements listed in the *Release Notes* and in the note below.

> **Note:** To save your source map as an XML file (.mxl file extension), you must have the Microsoft XML Core Services (MSXML) 4.0 installed on the same computer as Map Editor. If you do not have Microsoft XML Core Services (MSXML) 4.0 installed, you cannot save source maps as .mxl files and must use the .map extension.

To download and install the Map Editor:

1. From the Gentran Integration Suite **Deployment** menu, select **Maps**.

2. In the Download and Install section next to Download Map Editor (EN), click **Go!**

> **Note:** For the Japanese version, click **Go!** next to Download Map Editor (JP).

3. From the **File Download** dialog box, select a download option, then click **OK**.

   - ◆ If you choose to run the file, the operating system downloads the installation files immediately.

   - ◆ If you choose to save the file, the operating system prompts you to save the file. Browse to the location where you want to download the file and click **OK**. If you want to continue installing, run the file you just saved from the location you specified.

4. In the Security Warning page, select **Always trust content from Sterling Commerce (Mid America), Inc.** if you do not want to see similar security messages in the future when you download software from Sterling Commerce. Click **Yes**.

5. In the Map Editor Setup Welcome window, click **Next**.

6. In the Choose Destination Location window, select where you want to install the Map Editor:

   ◆ If you accept the default location, click **Next**.

   ◆ If you want to specify a different location, click **Browse**, specify the path to the folder, click **OK**, and click **Next**.

   > **Note:** If you specify a folder name that does not exist, you get a message asking you if you want to create that folder.

7. In the Select Program Folder window, specify the program folder where you want the Map Editor to be located.

   ◆ If you accept the default folder, click **Next**.

   ◆ If you want to specify a different folder, type a new name in place of Sterling Commerce or from the Existing Folders list, select a folder and click **Next**.

   The Map Editor setup wizard installs the Map Editor.

8. In the Setup Complete window, click **Finish**.

Removing the Map Editor

To remove the Map Editor:

1. On the computer where the Map Editor is installed, select **Start** > **Programs** > **Sterling Commerce** > **Uninstall Map Editor**.

   This is the default location of the Map Editor and Uninstall options. If the Map Editor is installed to a different location during the installation, then browse for the Uninstall option there.

2. Confirm that you want to completely remove Map Editor and all of its components by clicking **Yes**.

3. Click **OK** to complete the removal of the Map Editor.

Navigating in the Map Editor

The following table lists the parts of the Map Editor window:

Part	Function
Main Menu bar	Contains drop-down menus. Unavailable items are shaded.

Part	Function
Main toolbar	Enables you to access some of the most common functions in the Map Editor. Unavailable items are shaded. The Main toolbar is dockable, so you can affix it to any edge of the client window.
Status bar	Displays status information about a selection, command, or process; defines commands as you select each item in the menu; indicates any current keyboard-started modes for typing.

The Map Editor window enables you to navigate in four ways:

✦ Select the command from the Main Menu bar.

✦ Click the appropriate button on the Main toolbar.

✦ Click the appropriate part of the map.

✦ Right-click a map component to access a shortcut menu that contains all the available functions for that map component. The shortcut menus enable you to quickly and easily access available functions. The content of the shortcut menus varies, depending on the type and level of the selected map component.

When you start the Map Editor, the Main Menu bar contains a subset of commands. The full set of commands opens after you create a new map or open (load) an existing map.

Note: The transaction data file (TDF) button is disabled in Gentran Integration Suite. TDF is a proprietary Sterling Commerce data format that is supported in the Gentran:Server for Windows product. The TDF format is *not* supported in Gentran Integration Suite.

Mapping Preparation and Analysis

Mapping is a complex process that will be more successful if you plan and prepare before you create your first map. If your preparation and analysis are complete, you have all the information you must create the map in an efficient and logical manner. If you omit this critical task and proceed directly to creating the map, it is likely that creating the map will be a long and arduous task. Also, the map may be invalid because of oversights and omissions.

For more information, see:

✦ *Planning on Paper* on page 23

Planning on Paper

The first and most important task when creating a map is to analyze the mapping requirements. These are the steps of mapping analysis:

1. Analyze the input and output documents.

2.   Map the information (correlate the two documents).

3.   Use temporary storage map components (if necessary).

Analyzing Documents

The first step of mapping analysis is analyzing the input and output documents. You must define these documents to Gentran Integration Suite.

If you have a document format layout for either the input or output, start with that. Otherwise, you can create one by determining which fields are necessary to process the data correctly. Then, group the fields logically under records.

Mapping Information

To reconcile the two sides of the map, you must relate each input field to its corresponding output field and select a method for mapping it. To map information to a field, you use linking, standard rules, extended rules, or a combination of all three.

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

✦   Standard rules – Give you access to mapping operation functions that are more complex than simple linking but less complex than extended rules.

✦   Extended rules – Enable you to perform virtually any mapping operation you require.

The best practice is to determine which mapping operations are necessary on a field-by-field basis for the input and output documents.

After you finish correlating your input and output formats on paper, you can begin creating the map.

Using Temporary Storage Map Components

After analyzing both sides of your map, you may determine that you must establish temporary storage (work) areas for the map to handle specific data. See *Creating Temporary Records and Fields* on page 93 for more information about using temporary records and fields.

Example 1

For example, in an EDI-to-Positional map, items such as Ship To and Bill To name and address information may be extracted using extended rules from a group in EDI data. However, to map the shipping and billing information to your application file format, you must create temporary storage segments and elements on the EDI side of the map that do

not repeat (because you can only map components which occur at the same hierarchical level on both sides of the map). You can use an extended rule to extract the shipping and billing information from the EDI group that contains the Shipping/Billing information, and move it to the appropriate temporary storage elements (the extended rule must be run after the translator reads an occurrence of the EDI group). Then you can link the shipping and billing information directly from the temporary storage elements to your application fields.

Example 2

Another example (again using an EDI-to-Positional map) is if you do not know the order in which delivery and cancel dates appear in the EDI data. To map the delivery and cancel by date information to your application file, you must create a temporary storage segment and elements on the EDI side of the map. You can use an extended rule to extract the delivery and cancel by date information from the appropriate Date segment and move it to the appropriate temporary storage elements. Then you can map the delivery and cancel by date information directly from the temporary storage elements to your application fields.

Summary of Creating a Map

The following outline lists the steps for creating a map using the Map Editor.

1. Prepare and analyze:

   ◆ The layout of your input and output documents

   ◆ How you move data to or from each document field

2. Set the default date format (one time only).

   The first time you use the Map Editor, you must establish the default date format.

   For information about setting the date format, see *Setting the Default Date Format* on page 30.

3. Customize global display options (one time only).

   The first time you use the Map Editor, you must customize the global display options.

   For information about modifying global display options, see *Customizing Global Display Options* on page 27.

4. Create a map.

   For information about creating a map, see *Creating a Map* on page 31.

5. Define your input document.

   For information about defining a document, see:

   ◆ Chapter 2, *Mapping EDI Documents*

   ◆ Chapter 3, *Mapping Positional Documents*

   - ◆   Chapter 4, *Mapping Variable-Length-Delimited Documents*

   - ◆   Chapter 5, *Mapping CII Documents*

   - ◆   Chapter 7, *Mapping XML Documents*

   - ◆   Chapter 6, *Mapping SQL Documents*

6. Define your output document (EDI, positional, variable-length-delimited, CII, XML, or SQL). For information about defining a document, see any of the sections listed in the previous step.

7. Map the appropriate data for each field. For more information about:

   - ◆   Simple links, see *Creating Simple Links* on page 57.

   - ◆   Autolink function, see *Using Autolink in the Map Editor* on page 58.

   - ◆   Standard rules, see Chapter 8, *Using Standard Rules*.

   - ◆   Extended rules, see Chapter 9, *Using Extended Rules*.

   - ◆   User exits, see Chapter 11, *Using User Exits*.

8. Determine whether you must use temporary storage map components.

   For information about using temporary storage map components, see *Using Temporary Storage Map Components* on page 24.

9. Save and compile the map.

   For information about compiling a map, see *Compiling a Map* on page 59.

10. Print the mapping report.

   Validate and review the map, and make modifications as needed.

   For information about printing the mapping report, see *Printing a Mapping Report* on page 64.

11. Test the map.

   For information about testing a map, see *Testing a Map* on page 64.

Customizing the Map Editor Display

The Map Editor enables you to customize the display of maps in several ways. For more information, see the following sections:

✦ *Customizing Global Display Options* on page 27

✦ *Customizing Global Colors* on page 27

✦ *Customizing Global Fonts* on page 28

✦ *Customizing the Global Display of Links* on page 28

✦ *Customizing the Auto-Increment Map Version* on page 29

✦ *Customizing Confirmations* on page 29

✦ *Making the Two Sides of a Map Equal* on page 30

✦ *Setting the Default Date Format* on page 30

The Map Editor **Preferences** dialog box enables you to set global defaults for Gentran Integration Suite. You can change the display options at any time.

Customizing Global Display Options

To customize global display options for the Map Editor:

1. From the Map Editor **Options** menu, select **Preferences**.

   The Preferences dialog box opens with the Tree tab displayed by default.

2. Do you want to turn on the default display of group, record, and field descriptions?

   Typically, you want to have all the descriptions displayed for reference. However, for a variety of reasons, you may not want the descriptions displayed. Depending on the size of your monitor, it may be easier to see the entire map if the descriptions are not displayed. You can also experiment with the font size of the map. See *Customizing Global Fonts* on page 28 before you turn off the display of descriptions.

   ◆ If Yes, click the check box in front of the appropriate settings.

      To turn off the default display of group, record, and field descriptions, clear the appropriate check box.

   ◆ If No, go to the next step.

3. Do you want to change the default display colors?

   You may find it helpful to set active items to one color, inactive items to another color, and active, linked items to a third color. Differentiating among these items by color is helpful when you are creating or troubleshooting a map.

   ◆ If Yes, see *Customizing Global Colors* on page 27 for more information.

   ◆ If No, go to the next step.

4. Do you want to change the default display font?

   ◆ If Yes, see *Customizing Global Fonts* on page 28 for more information.

   ◆ If No, go to the next step.

5. Click **OK** to save changes.

Customizing Global Colors

The Colours function enables you to select foreground and background colors to visually define various map components. Customized colors apply to all maps. The use of color is optional.

Tip: You may find it helpful to set active items to one color, inactive items to another color, and active, linked items to a third color to differentiate among them.

To customize colors for all maps:

1. From the Map Editor **Options** menu, select **Preferences**.

2. Click **Colours**.

3. From the **Item** list, select the type of map component.

4. From the **Attributes** list, select the attributes of the map component you selected.

5. Select the foreground and background colors for the item.

6. Repeat steps 3 - 5 for each item you want to customize.

7. When you are finished customizing the colors for all items, click **OK** to save changes.

Customizing Global Fonts

The Font function enables you to change the font type, style, and point size of the font that is used in the display of all maps. The default font that Gentran Integration Suite uses is MS Sans Serif 9 point.

You can shrink the font if you must view more of the map on your monitor, enlarge the font, or change the type and style to suit your preferences.

To customize the display font for all maps:

1. From the Map Editor **Options** menu, select **Preferences**.

2. Click **Font**.

3. From the **Font** box, select the type of font.

   The default varies according to the default language of the computer. For example, the English default font is MS Sans Serif and the Japanese default font is MS Gothic.

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 points.

6. Click **OK** to save changes.

Customizing the Global Display of Links

Mapping links are the visual lines that connect the fields on the input side of the map to mapped fields on the output side of the map.

Note: This topic refers to elements, fields, and transfer form data (TFDs) generically as fields.

To customize the global display of mapping links:

1. From the Map Editor **Options** menu, select **Preferences**.

2. Click the **Links** tab.

3. Select the linking option that you want to set as the default for all maps.

    ◆   Show no links – Do not display mapping links.

    ◆   Show links to or from the currently selected field – Display only mapping links for the currently selected field (this option enables you to concentrate on the selected field and removes the confusion of viewing many links at once).

    ◆   Show links to or from all visible fields – Display all mapping links.

4. Click **OK** to save changes.

Customizing the Auto-Increment Map Version

Gentran Integration Suite automatically increments the map version number using a global option you set. You can set a global option on the Preferences dialog box to specify when map version numbers are incremented. The available options for automatically incrementing the map version number are:

✦   Only when manually changed (this is the default).

✦   Ask whether to increment when saving a map.

✦   Ask whether to auto-increment when compiling a map.

✦   Always auto-increment when saving a map.

✦   Always auto-increment when compiling a map.

The Auto-increment function updates the minor version number of the map in the Map Details dialog box, up to 255. If the minor version number exceeds 255, the Map Editor updates the minor version number to zero and increases the major version number. For example, version number 1.255 is auto-incremented to 2.0.

To enable auto-incrementing maps:

1. From the Map Editor **Options** menu, select **Preferences**.

2. Click the **Version** tab.

    The Map Editor displays the auto-increment version options.

3. Select the appropriate option.

4. Click **OK** to set the auto-increment option.

Customizing Confirmations

The Confirmations tab on the Preferences dialog box enables you to specify when you want confirmation messages displayed.

To customize the Map Editor confirmations:

1. From the Map Editor **Options** menu, select **Preferences**.

2. Click the **Confirmations** tab.

    The Map Editor displays the confirmation options.

3. Set the global confirmation options by either selecting the **Confirm everything** check box (displays all confirmation messages) or by clearing the **Confirm everything** check box and then selecting individual confirmation messages by action performed.

Note: To increase the likelihood that the links in your maps are valid, select the **link objects at different levels** and **link objects with different maximum usages** confirmations.

---

4. Click **OK** to save the confirmation options.

Making the Two Sides of a Map Equal

When you open a map, the input and output sides of the map are displayed in equal proportions. After manipulating the map and moving the center dividing bar between the input and output sides, you restore the two sides to their original proportions.

The Equalize function enables you to restore the two sides of the map with focus in equal proportions.

To make the two sides of a map equal, select **Equalize** from the **View** menu.

Setting the Default Date Format

You typically establish the default date format for all date fields *one time only*; however, you can override this default in the **Field Properties** dialog box.

The Date Format function changes the default date format for all maps. However, the format of the existing date fields does not change. The default is used only for new maps.

To set the default date format:

1. From the Map Editor **Options** menu, select **Preferences**.

2. Click 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 save changes.

Note: 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 **Date Formats** from the **Options** menu. If you change a default date format or add a new one, the change is only effected on the current machine. If you then modify the map on a difference machine, you need to make the same Date Format change or addition on that machine.

Creating a Map

Create Gentran Integration Suite maps for document-to-document mapping.

Each map has two sides. Each side represents a data format; the input side is what you are translating from and the output side is what you are translating to. Each data format has specific map components.

For information about creating a map using the customize option for the EDI, variable-length-delimited, CII, XML, or SQL data format, or using the additional positional format options, see the respective sections.

When you create a map, for both the input and output sides, you choose whether you want to use a preloaded standard, an existing file format, or want to create a new file format for that side of the map (including selecting from standards that you have previously downloaded to the standards database). The preloaded standards are downloaded when you download the Map Editor to your machine or any optional standards (such as those included with the Financial Services XML Standards bundle).

Note: If the map you are creating contains greater than 20,000 objects, you will receive a message noting that this map contains a very large number of objects. For best performance, it is recommended that you consider whether any unnecessary objects in the map can be removed, do not expand the entire object tree—expand only the section of the tree you are currently mapping, consider using the "Show links to or from the currently selected element" option instead of the "Show links to or from all visible elements" option, and save the map using the.MAP file format (using the Save As function).

To create a map:

1. From the Map Editor **File** menu, select **New**.

2. In the New Map wizard, answer the following questions and then click **Next**.

   ◆ What kind of map are you creating?

      Accept the default, Sterling Integrator.

      Caution: Always accept the Sterling Integrator default when you are using the Gentran Integration Suite product.

      Note: The other map options enable you to import maps from other Sterling Commerce products in order to convert them to Sterling Integrator-type maps.

   ◆ What is the name of the map?

      Type the unique name of the map. The Map Editor adds the default .mxl extension.

◆   What is your name?

Type your name if it differs from the user name prompted by the New Map wizard. The New Map wizard displays Input Format fields. You must complete the format of the input side of the map. This is the format of the data that is translated by Gentran Integration Suite.

3.   In the Input Format window, specify how you want to define the data format by selecting one of the following:

◆   Create a new data format using this standard

Note: This selection allows you to use a preloaded standard (that is, you do not have to load the data format for the standard to the Map Editor).

•   Select the standard and click **Messages**

•   Complete the Map Wizard

◆   Load the data format from a saved definition

Note: This selection allows you to use an existing file format.

•   Type the path and file name of the saved definition (.ddf or .ifd extension).

•   Click **Browse** to display the **Open File Definition** dialog box.

◆   Create a new data format using this syntax

Select a format.

4.   In the Output Format window, specify how you want to define the data format by selecting one of the following:

◆   Create a new data format using this standard

Note: This selection allows you to use a preloaded standard (that is, you do not have to load the data format for the standard to the Map Editor).

•   Select the standard and click **Messages**

•   Complete the Map Wizard

◆   Load the data format from a saved definition

Note: This selection allows you to use an existing file format.

•   Type the path and file name of the saved definition (.ddf or .ifd extension).

•   Click **Browse** to display the **Open File Definition** dialog box.

◆   Create a new data format using this syntax

Select a format.

5.   Click **Finish** to create the map. The map opens in the Map Editor window.

6. In the Map Editor, select **File** > **Save** to save the map. Do not use spaces or apostrophes in the map name. To save a map as a .map file, select **File > Save As** and then select **Source Maps (*.map)** from the **Save as type** list.

Note: Prior to opening an .mxl (XML-formatted) file, the Map Editor verifies that you have the Microsoft XML Core Services (MSXML) 4.0 installed on the same computer as Map Editor. If you do not have the Microsoft XML Core Services (MSXML) 4.0 installed, the Map Editor cannot save or load .mxl source files.

Opening a Map

To open a map or several maps at one time:

1. From the Map Editor **File** menu, select **Open**.

2. In the **Open** dialog box, select one .mxl or .map file, or select several files by one of the following methods:

    ◆ Hold **Shift** as you select a range of files.

    ◆ Hold **Ctrl** as you select individual files.

Tip: If the Open dialog box does not display .mxl or .map files, browse to the folder where your maps are located. Later, you can change the default folder that opens when you browse for maps. For more information, see *Changing the Default Folder in the Open Dialog Box* on page 33.

---

3. Click **Open**. When you load a .mxl file, the Map Editor references the Mapper.xsd schema file (installed with the Map Editor) to validate the .mxl file prior to processing it. Prior to opening an .mxl (XML-formatted) file, the Map Editor verifies that you have the Microsoft XML Core Services (MSXML) 4.0 installed on the same computer as Map Editor. If you do not have the Microsoft XML Core Services (MSXML) 4.0 installed, the Map Editor cannot save or load .mxl source files.

For more information, see *Changing the Default Folder in the Open Dialog Box* on page 33.

Changing the Default Folder in the Open Dialog Box

You can change the folder that opens by default in the **Open** dialog box.

To change the default folder:

1. From the Map Editor **Options** menu, select **Preferences**.

2. In the **Preferences** dialog box, click the **Files** tab.

3. In the **When browsing for maps, start in this folder** field, either type or browse to the folder that you want to open by default when you browse for a map.

    The default installation folder is located in Program Files\Sterling Commerce\Map Editor\Source Maps.

---

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

Using Data Definition Format and Integrator File Definition Files

The *data definition format* (DDF) is a Sterling Commerce format that contains an XML-formatted description of the input or output side of a source map and has the file extension .ddf. The *Integrator File Definition* (IFD) is a Sterling Commerce format that contains a binary description of the input or output side of a source map and has the file extension .idf. Both DDF and IFD files include the hierarchical and looping structure of the data, the map objects (groups, records, fields) and their attributes (for example, names, descriptions, data types). A DDF or IFD file can be loaded (imported) or created (for export) from the Map Editor.

Note:	DDF and IFD files do not contain standard or extended rules, links, or any other map information unrelated to the data format. The one exception is the Use Code code lists. If the map references a Use Code standard rule and instructs Gentran Integration Suite to raise a compliance error if the code is not found in the Use Code code list, Gentran Integration Suite saves the standard rule and the associated code list.
	For more information about code lists and using the Use Code standard rule, see Chapter 8, *Using Standard Rules*.

For more information, see:

✦ *Requirement for Using DDF Files* on page 34
✦ *Opening a DDF or IFD File* on page 34
✦ *Creating a DDF or IFD File* on page 35

Requirement for Using DDF Files

Because DDF files are formatted in XML, you must have the Microsoft XML Core Services (MSXML) 4.0 installed on the same computer as the Map Editor. You can obtain Microsoft XML Core Services (MSXML) 4.0 by installing the most current release of Microsoft Internet Explorer. If you do not have the Microsoft XML Core Services (MSXML) 4.0 installed, the Map Editor cannot support DDF files.

Opening a DDF or IFD File

Gentran Integration Suite enables you to open, or import, a DDF or IFD file in two ways—when you create a new map, and when you open a DDF or IFD file into one side of an existing map. Either way, using a DDF or IFD file provides you with a quick way to create either side of a map.

When you create a new map, the New Map wizard enables you to select a DDF or IFD file to use as the basis for the input or output side of the map. For more information about using a DDF or IFD file when creating a new map, see *Creating a Map* on page 31.

When you open a DDF or IFD file, Gentran Integration Suite assigns acceptable defaults if attributes are not included. For example, if a value is not specified for Minimum Length, a value of zero is assigned.

To open a DDF or IFD file into an existing map:

Caution: Opening a DDF or IFD file into an existing map replaces the selected side of the map. After you open a DDF or IFD file into a map, you cannot undo the operation.

---

1. In the Map Editor, right-click the **data format** icon on the side of the map where you want to open the DDF or IFD file and select **Open File Definition** from the shortcut menu.

Note: If you already used Gentran Integration Suite to create that side of the map, you are prompted with a message that warns you that the existing file format will be replaced. Click **Yes** to continue.

---

2. Browse to locate the DDF or IFD file.
3. Click **Open** to open the selected DDF or IFD file into the map.

Creating a DDF or IFD File

Gentran Integration Suite enables you to create (save as a DDF/IFD) a DDF or IFD file so that you can use it as a starting place in other maps or as a means to update existing maps.

To create a DDF or IFD file:

1. In the Map Editor, right-click the **data format** icon on the side of the map from which you want to create a DDF or IFD file and select **Save File Definition As** from the shortcut menu.
2. Browse to the location where you want to save the DDF or IFD file.
3. Click **Save** to save the DDF or IFD file.

Defining Map Details

The Map Details dialog box enables you to edit the details of the map, including the description and version information.

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

Note: When you import a map from Gentran:Server for Windows (or from Gentran:Server for UNIX, if the maps were created using the Application Integration MAPPER.EXE), you must change the Map Function to Gentran Integration Suite.

For more information about Map Details Dialog Box properties, press F1 for Help or see Appendix B, *Map Editor Properties*.

To specify map details:

1. From the Map Editor **Edit** menu, select **Details**.

2. To change the map description, type the new description in the **Description** box.

3. If you are using a map that was created with Gentran:Server for Windows NT 2.x and you need the rules to run exactly as they did in that product, then select the **Gentran:Server for Windows NT 2.x Compatible Rule Execution** check box.

4. If you want to change the map version, type the appropriate version numbers in the **Major version** and **Minor version** boxes.

5. Click **OK** to save changes.

Adding Map Components

If you did not use a DDF or IFD file or schema, you must manually add the map components of the data format.

Tip: While you are working on the map, it is a good practice to save the map often.

---

1. In the Map Editor, right-click the map component that precedes what you want to add and select either:

   ◆   Create Sub – Makes the new component subordinate to the selected map component

   ◆   Insert – Puts the component at the same level as the selected map component

2. When you import a map from Gentran:Server for Windows (or from Gentran:Server for UNIX, if the maps were created using the Application Integration MAPPER.EXE), you must change the Map Function to Sterling Integrator.

3. In the map component properties dialog box, name the component. You can specify the other properties now or later.

   For more information about properties, press F1 for Help or see Appendix B, *Map Editor Properties*.

4. Click **OK** to save changes and close the dialog box.

Defining Map Component Properties

After you have added a map component to a map, you must define the properties of the map component. If you want to view or must modify the properties of a map component, follow the same procedure.

See Appendix B, *Map Editor Properties* for more information about map component properties.

Tip: While you are working on the map, it is a good practice to save the map often.

To define, modify, or view property information:

1. In the Map Editor, right-click the map component. From the shortcut menu, select **Properties**.

2. In the properties dialog box, specify the properties as necessary.

   For more information about properties, press **F1** for Help or see Appendix B, *Map Editor Properties*.

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

Using Copy, Cut, and Paste

Use the Copy, Cut, and Paste functions to move information in the map. You must use these functions if you want to create nested looping structures.

You can cut or copy a single map component (loop, record, or field) and paste it in another location in the map. Copied map components retain all the information of the original map component, though Map Editor changes the name of the pasted map component by adding a colon (:) and a number to differentiate it from all other map components. If the copied map component contains subordinate map components (for example, a record contains subordinate fields), the subordinate map components are also copied. You can also cut, copy, and paste a map component from one map to another.

To cut, copy, and paste a map component:

1. In the Map Editor, select the map component that you want to cut or copy.

2. From the Map Editor **Edit** menu, select **Copy**.

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

---

3. Select the map component that you want the cut or copied selection to be pasted *after*.

    You must match the type of input or output you cut or copied from to the type of the side of the map where you paste it. For example, if you copied an XML component, you can paste that component into only the XML side of a map.

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

    If the map component that you selected is a group, Gentran Integration Suite 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**.

Splitting Groups

The Split function enables you to split (break) the following objects into two loops:

✦ Group

✦ Repeating record

✦ Repeating element

✦ Repeating composite

If you split a group (or any one of these map components), the individual iterations of the two groups that you created by splitting a group add up to the original number of iterations of the group before being split. For example, you might have a group with 50 iterations. You could split that group into two groups: one that iterates 20 times and one that iterates 30 times.

Use the Split function when you need more than one instance of a map component that occurs multiple times. The Split function is available only if a group or repeating record, element, or composite is selected.

To split a group or repeating record, element, or composite:

1. In the Map Editor, select an item from which you want to extract one iteration.

2. In the Main toolbar, click **Split**.

3. In the **First Loop Entries** box, type the number of iterations that you want the group or repeating record, repeating element, or repeating composite to split.

    The number you type must be greater than zero and less than the maximum number of iterations of the loop.

    For example, if the X loop repeats a maximum number of five times, and you type 2, the resulting split generates one X loop that repeats a maximum of two times and a second X loop that repeats a maximum of three times.

4.   Click **OK** to complete the split.

> **Note:**   You can use the Copy and Paste functions (and then change the number in the **Maximum usage** box on the map component's **Properties** dialog box) to accomplish the same task.

Promoting Groups

The Promote function extracts one iteration (instance) of one of the following map components:

✦   Group

✦   Repeating record

✦   Repeating element

✦   Repeating composite

Promote is a specialized version of the Split function. For example, if you have a group with 50 iterations and you select Promote, you then have two groups: one group with an iteration of 1 and a group that iterates 49 times.

This function enables you to map unique data from a document and to type a specialized definition. Gentran Integration Suite specifies that only one-to-one (no loop) or many-to-many (loop) mapping relationships are valid. The Promote function is available only if a group or repeating record, element, or composite is selected.

To promote a group or repeating record, element, or composite:

1.   In the Map Editor, select an item from which you want to extract one iteration.

2.   In the Main toolbar, click **Promote**.

The Map Editor extracts one iteration (instance) of the looping structure.

> **Note:**   Promoting is the same as splitting off one iteration. You can use the Copy and Paste functions (and change the number in the **Maximum usage** box on the appropriate **Properties** dialog box of the map component) to accomplish the same task. Depending on the circumstances, you can use the Split or Copy and Paste functions instead.

Finding Information in the Map

The Map Editor gives you the ability to quickly and easily find information in your maps. The Find feature enables you to specify a search string for which the Map Editor then searches the entire map, including all the parameters for each map component (for example, data type, field length, and so forth). The Find Next feature then enables you to search the map for the next instance of the specified text.

To search for information in a map:

---

1.   From the Map Editor Edit menu, select **Find**.

2.   In the **Find What** box, type the text you want to locate in the map.

3.   If you want to find text only if it conforms to the capitalization you specify, click the **Match Case** check box.

4.   To find the specified text between the insertion point and the beginning of the document, click **Up**.

      To find the specified text between the insertion point and the end of the document, click **Down**.

5.   Click **Find Next** to locate the next occurrence of the specified text.

> **Note:**   You can also press **F3** or select **Find Next** from the **Edit** menu to locate the next occurrence of text you specified using the Find feature.

Formatting Data in Fields

When you define or modify a field, you must specify the type and format of the data. The options that are available for the format of the field depend on which type you select (string, number, date/time, Bin Len, or Bin Data).

> **Note:**   This topic refers to elements, fields, and transfer form data (TFDs) generically as fields.

For more information, see the following sections:

✦   *Using a String Field* on page 40

   ◆   *Creating Syntax Tokens* on page 41

   ◆   *Deleting a Syntax Token* on page 44

   ◆   *Deleting a Character Range* on page 44

✦   *Using a Number Field* on page 44

✦   *Using a Plus Sign in a Number Field* on page 47

✦   *Using a Date/Time Field* on page 53

Using a String Field

A string field contains one or more printable characters. If you specify that a field is a string field, you must also specify the format of the string by assigning a syntax token. A *syntax token* is a data sub-type that applies only to string fields and can contain of one or more character ranges and single characters, such as $, Q, or @. A *character range* is a pair of characters that define the start and end characters of the range. The Map Editor provides several syntax tokens, including syntax token K, which includes all Japanese double-byte characters and excludes single-byte characters. The Map Editor also provides a Free

Format option, which indicates that any characters are acceptable in the field, and the translator does not check the characters for compliance. You can also create special syntax tokens to meet your specific needs.

You assign a syntax token to a field to check the field for compliance during translation. For example, if a field with syntax token A assigned to it (character range A through Z and a through z) contains a number, the translator generates a compliance error. You specify that a field is a string and assign a syntax token in the **Field**, **Element**, or **TFD Properties** dialog box, on the **Validation** tab, in the data format field.

To specify a string field:

1. In the Map Editor, double-click an existing field or create a new one.

    For more information about creating fields, see *Adding Map Components* on page 36.

2. In the appropriate **Properties** dialog box, click the **Validation** tab.

3. From the data-type list, select **String**.

4. From the data format list, select **Free Format** or the appropriate syntax token.

    Free Format indicates that any characters are acceptable in the field. The translator does not check the characters for compliance.

    If you select Free Format, the drop-down list becomes an editable text box. Free Format indicates that any characters are acceptable in the field. The translator does not check the characters for compliance.

5. In the text box, type the appropriate characters or complex syntax for this field.

6. Click **OK** to save your selections.

Creating Syntax Tokens

Some syntax tokens are provided in the Map Editor. If you need a different syntax token, you can create a syntax token to meet your needs.

Note: When you create a syntax token, it applies only to the current map. You must create a syntax token for each map you create.

Creating a Syntax Token for Western European Languages

The Map Editor uses the ANSI character set when determining the start and end range for a syntax token.

To create a syntax token:

1. From the Map Editor **Edit** menu, select **Syntax Tokens**. The **Syntax Tokens** dialog box opens.

2. Click **New**.

3. In the **Edit Syntax Token** dialog box, in the **Token** field, type the unique one-character alphanumeric value that the Map Editor recognizes as containing the permitted range of characters you designate.

   The token can be only one unique character, uppercase or lowercase, alphabetical or numeric, 1 – 9.

4. To specify the character range, click **New**.

   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 permitting the character ranges A – Z and a – z. This definition indicates that token A permits only uppercase and lowercase alphabetical characters.

5. In the **Edit Character Range** dialog box, in the **Start character** box, type the character that begins the permitted character range.

   For example, if the character range you want to define is B – D, type **B** in the **Start character** box.

   If you type a character, such as é, that is not accepted, you must type it in hexadecimal code. To type hexadecimal characters, type **0**(zero)**x** (that is, **0X**), followed by the hexadecimal code. For example, the hexadecimal equivalent of é is 0xE9.

   > **Note:** You can now use the null character (hexadecimal 0x00) in your maps.

   The Start character and End character can each be only one character, uppercase or lowercase, alphabetical or numeric, 1 – 9.

   For more information about possible character ranges, see *Character Ranges for Western European Languages* on page 43.

6. In the **End character** box, type the character that terminates the permitted token range. Use the guidelines from step 5.

7. Click **OK** to return to the **Edit Syntax Token** dialog box.

8. Add additional character ranges as necessary.

9. Click **OK** to save the syntax token.

Character Ranges for Western European Languages

For Western European languages, refer to the ANSI character chart (Windows codepage 1252 or ISO-8859-1, Latin-1). This chart also displays ranges so that you can type appropriate ranges for the characters in your language. If no chart is available, use the following guidelines:

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

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

✦ Scandinavian users must also add the following ranges in order to include Œ and œ.

Start	End
0x8C	0x8C
0x9C	0x9C

Creating a Syntax Token for Asian Languages

Double-byte character set (DBCS) syntax tokens enable you to create a map that accepts double-byte characters. The DBCS button is active on the Syntax Tokens dialog box if:

✦ You are running the Map Editor on a Chinese, Japanese, or Korean version of the Windows operating system.

✦ You have added font support for one of these languages to a Windows 2000 system.

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

To create a DBCS syntax token:

1. From the Map Editor **Edit** menu, select **Syntax Tokens**.

2. Click **DBCS**.

3. In the **DBCS Syntax Tokens** dialog box, select the codepage that most closely matches the character set that you are mapping.

   For example, if you are mapping Japanese characters, use codepage 932.

4. Click **New**.

5. In the **Edit DBCS Syntax Token** dialog box, in the **Token** field, type the unique one-character alphanumeric value that the Map Editor recognizes as containing the permitted range of characters you designate.

6. Select the lead byte from the **Lead-Byte** box.

   Double-byte characters are composed of a lead byte and a trail byte. For example, the character ▯ is code point 93F9 (lead byte 93, trail byte F9).

7. To exclude individual characters or groups of characters from the token (you do this if there are certain characters that you will not accept in your data), make characters unavailable by either:

   ◆ Clicking a character to make it appear shaded

   ◆ Dragging the cursor across a group of characters to make them appear shaded

8. Click **Close** to save the syntax token.

Deleting a Syntax Token

To delete a syntax token:

1. From the Map Editor **Edit** menu, select **Syntax Tokens**.

2. If you are deleting a DBCS syntax token, click **DBCS**.

3. Select the token that you want to delete and, if you are certain, click **Delete**. The selected syntax token is deleted without a warning message.

Deleting a Character Range

To delete a character range from a syntax token:

1. From the Map Editor **Edit** menu, select **Syntax Tokens**.

2. If you are deleting a character range for a DBCS syntax token, click **DBCS**.

3. Select a syntax token and click **Change**.

4. Select the character range that you want to delete and, if you are certain, click **Delete**. The selected syntax token is deleted without a warning message.

Using a Number Field

A number field contains an implied decimal (integer), a real number, an overpunch, or a packed number (for Positional maps only) that can be mathematically manipulated. If you specify that a field is a number, you must also specify the format of N (implied decimal) and the number of decimal places, R (real) and the number of decimal places, Overpunch and the number of decimal places, and Packed decimal and the number of decimal places (for Positional only).

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.

Overpunch is a signed numeric with implied decimal position. The Overpunch format specifies that the last digit represents the sign of the number and the number value. The sign is internal and trailing, zero is always positive, the field is always right-justified, and field contains a zero-filled dollar-cents amount with the number of positions you specify to the right of the implied decimal point. All other positions are to the left of the implied decimal point. A dollar field with an Overpunch sign replaces the rightmost digit in the field. For example, the dollar field of $99.95 is represented as 999E with truncation. A negative dollar amount of $2.50 is represented as 25} with truncation.

Note: Overpunch signs are used in dollar fields to represent positive and negative dollar amounts without expanding the size of the field to hold a plus or minus character. The rightmost (least significant) digit of a dollar field must be an Overpunch sign, not a digit. The signed value designates the positive or negative status of the numeric value.

For the Packed decimal format, two digits are put ("packed") into each byte of storage, and thus each byte (except for the low-order byte) can contain two decimal numbers—thus halving the storage requirements. The low-order byte contains one digit in the leftmost position and the sign (positive or negative) in the rightmost position. The sign portion of the low-order byte indicates whether the numeric value represented in the digit portions is positive or negative. Since a digit only has ten possible values (0 – 9), it can be represented in only 4 bits. Therefore, you can get two digits in each eight-bit byte (for example, decimal 92 would be encoded as the eight-bit sequence 1001 0010). Normal character representation only stores one character (digit) per byte, so packed data only requires half the storage of character (unpacked) data. Each four bits of a byte is called a *nibble*, and each nibble contains one digit of the value, stored in binary form, known as Binary Coded Decimal (BCD). The last nibble—the low-order byte—is used to store the sign for the number. This nibble stores only the sign, not a digit: C hex indicates a positive sign, D hex indicates a negative sign, and F hex indicates that the number is unsigned.

Note: When you use packed decimal numbers on the input side of a map, the translator truncates trailing zeros when it converts the value to a floating point numeric value.

The nibble (one-half byte) used for marking the sign for packed decimal number in the Map Editor is different for zSeries (mainframe) operating systems and iSeries (AS/400) operating systems.

✦ For zSeries data, the last nibble of a packed decimal number is 0x0C if the sign is positive and 0x0D if the sign is negative.

✦ For iSeries data, the last nibble of a packed decimal number is 0x0F if the sign is positive and 0x0D if the sign is negative.

Incoming data is handled by the Gentran Integration Suite libraries and outgoing data is handled by the **datatype.PackedDecimal** property in the **translator.properties** property file. If the datatype.PackedDecimal property exists then packed data is converted using the zSeries format (unsigned). If the translator.properties file is missing or if the datatype.PackedDecimal property is missing from the property file, the packed data is converted as signed by default.

The following table lists the available number options:

Field	Type of Decimal Point	Number of Decimal Places
R0	Explicit	None
R1	Explicit	Up to one
R2	Explicit	Up to two
R3	Explicit	Up to three
R4	Explicit	Up to four
R5	Explicit	Up to five
R6	Explicit	Up to six
R7	Explicit	Up to seven
R8	Explicit	Up to eight
R9	Explicit	Up to nine
N0	Implied	None
N1	Implied	One
N2	Implied	Two
N3	Implied	Three
N4	Implied	Four
N5	Implied	Five
N6	Implied	Six
N7	Implied	Seven
N8	Implied	Eight
N9	Implied	Nine
Overpunch0	Implied	None
Overpunch1	Implied	One
Overpunch2	Implied	Two
Overpunch3	Implied	Three
Overpunch4	Implied	Four
Overpunch5	Implied	Five
Overpunch6	Implied	Six
Overpunch7	Implied	Seven
Overpunch8	Implied	Eight
Overpunch9	Implied	Nine

Field	Type of Decimal Point	Number of Decimal Places
Packed0	Implied	None
Packed1	Implied	One
Packed2	Implied	Two
Packed3	Implied	Three
Packed4	Implied	Four
Packed5	Implied	Five
Packed6	Implied	Six
Packed7	Implied	Seven
Packed8	Implied	Eight
Packed9	Implied	Nine

If you select an implied decimal (N format) for a field, and the data in that field has fewer than the specified number of decimal places, the translator pads the data with zeros to the right 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.

To specify a number field:

1. In the Map Editor, double-click an existing field or create a new one.

   For more information about creating fields, see *Adding Map Components* on page 36.

2. In the appropriate **Properties** dialog box, click the **Validation** tab.

3. From the **data-type** list, select **Number**.

   Number indicates that the field is an implied decimal number, a real number, an overpunched number, or a packed decimal number (for Positional formats only) that can be manipulated mathematically.

4. From the **data format** list, select the appropriate option.

5. Click **OK** to save your choices.

Using a Plus Sign in a Number Field

Map Editor now enables you to allow a "+" (plus sign) in positive numeric fields (fields with the data-type Number) on the input side of a map, and generate a "+" (plus sign) in positive numeric fields on the output side of a map. For example, you can allow and generate **+5.5** instead of **5.5**. The Map Editor already enables you to allow and generate a "-" (negative sign) in numeric fields.

The configuration of the ability to use a plus sign occurs in Gentran Integration Suite at two levels:

1. *Global Translator Properties Configuration* on page 48

2. *Field-level Validation Configuration* on page 51

The field-level validation setting enables you to override settings you configure for Global Translator Properties. Additionally, Gentran Integration Suite provides you with an optional method of specifying the default that will be used for new fields, which can then be modified. This Preferences dialog box (Standard Formats tab) specifies the initial value that will be used for new fields (in the field-level properties dialog box) for both existing and new maps. See *Standard Format Preferences Configuration* on page 52 for more information.

Caveats for Allowing and Generating the Plus Sign in Numeric Fields

The following requirements must be noted prior to configuring Gentran Integration Suite (Map Editor and/or translator properties) to allow and generate the plus sign in numeric fields:

✦ Ensure that the field length is sufficient to accommodate the plus sign—otherwise the data may be truncated. On the input side of the map, the field in which the plus sign will be read must be long enough to accept the addition of a plus sign in the data. And, on the output side of the map, the field to which the data is being mapped must be long enough to accommodate the generation of the plus sign in the data.

✦ If you currently use the plus sign as a delimiter (this is a default for EDIFACT), when you add the plus sign to input data, you must use preface the plus sign with the release character you have specified. For example, if you specified the plus sign as the delimiter for elements and specified the release character as a question mark (?), then you would specify the value "5.000" in the data as follows:

$$+?+5.000$$

delimiter   release   plus sign in data
                 character
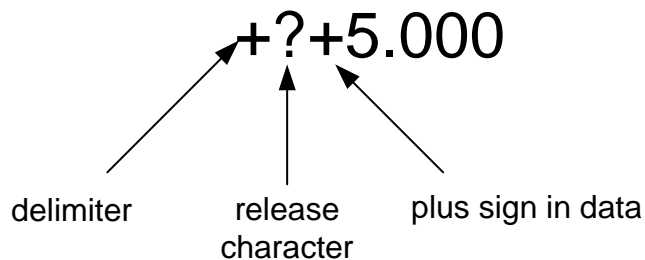
If you currently use the plus sign as a delimiter (this is a default for EDIFACT), when a plus sign is generated in your output data, it will be preceded by the release character that you specified.

Global Translator Properties Configuration

You can configure global (system-level) translator properties in the *install_dir*/**properties/customer_overrides.properties** file. This is a global setting that defines defaults for all maps or you can also define defaults for specific maps (by map names).

By default, the customer_overrides.properties file is empty; however, you can specify the use of a plus sign in numeric fields as follows:

✦ *Using "+" in All Maps* on page 49
✦ *Using "+" in a Single Map* on page 49

Using "+" in All Maps

The **translator.input.allowsigneddecimal** property enables the use of a plus sign on the input side (only) for all maps. The default is TRUE (enabled), but you can configure it to FALSE (disabled). This will trigger a translation error when a plus sign precedes a number.

Note:	Regardless of whether you set the **translator.input.allowsigneddecimal** property to true (enabled) or false (disabled), it may be overridden by any plus sign parameters specified by using field-level validation.

The **translator.output.generatesigneddecimal** property enables the generation of a plus sign on the output side (only) for all maps. The default is FALSE (disabled), but you can configure it to TRUE.

Note:	Regardless of whether you set the **translator.input.allowsigneddecimal** property to true (enabled) or false (disabled), it may be overridden by any plus sign parameters specified by using field-level validation.

You can disable the use of the "+" for the input side of all maps, as follows:

1. If you want to disable the use of a plus sign on the input side of all maps, in the *install_dir*/**properties/customer_overrides.properties** file, set the **translator.input.allowsigneddecimal** property to **False**, as shown in the example below:

```
translator.input.allowsigneddecimal=false
```

2. Save the **customer_overrides.properties** file.

You can enable the use of the "+" for the output side of all maps, as follows:

1. If you want to be able to generate a plus sign on the output side of all maps, in the *install_dir*/**properties/customer_overrides.properties** file, set the **translator.output.generatesigneddecimal** property to **True**, as shown in the example below:

```
translator.output.generatesigneddecimal=true
```

2. Save the **customer_overrides.properties** file.

3. Restart your Gentran Integration Suite instance so the property file changes are recognized.

Using "+" in a Single Map

The **translator.<mapname>.input.allowsigneddecimal** property enables and disables the use of the plus sign on the input side (only) of a specific map, where <mapname> is the

name of the map as specified on the Map Details dialog box (Description field). The default is TRUE (enabled), but you can configure it to FALSE (disabled).

Note: Regardless of whether you set the **translator.<mapname>.input.allowsigneddecimal** property to true (enabled) or false (disabled), you may override it with any plus sign parameters specified by using field-level validation.

The **translator.<mapname>.output.generatesigneddecimal** property enables the generation of a plus sign on the output side (only) of a specific map, where <mapname> is the name of the map as specified on the Map Details dialog box (Description field). The default is FALSE (disabled), but you can configure it to TRUE (enabled).

Note: Regardless of whether you set the **translator.<mapname>.input.allowsigneddecimal** property to true (enabled) or false (disabled), it is overridden by any plus sign parameters specified by using field-level validation.

You can disable the use of the "+" on the input side in a single map, as follows:

1. If you want to disable the use of the plus sign on the input side of a map, in the *install_dir*/**properties/customer_overrides.properties** file, add the line **translator.<mapname>.input.allowsigneddecimal**=**False** and replace **<mapname>** with the actual name of the map in which you want to disallow the use the plus sign, as specified on the Map Details dialog box (Description field) for that map. This triggers a translation error when a plus sign precedes a number.

    For example, if the name of the map in which you want to disallow the use of the plus sign on the input side of the map is "RomanMap.mxl," replace the <mapname> with **RomanMap**, as shown in the example below:

    ```
    translator.RomanMap.input.allowsigneddecimal=false
    ```

    Note: If a map Description contains spaces, you must amend it in the customer_overrides.properties file by putting a backslash before every space. Thus if the map Description is **Roman Map**, the example statement is:

    ```
    translator.Roman\ Map.output.generatesigneddecimal=false
    ```

2. Save the **customer_overrides.properties** file.

You can enable the use of the "+" on the output side in a single map, as follows:

1. If you want to be able to generate the plus sign on the output side of a map, in the *install_dir*/**properties/customer_overrides.properties** file, add the line **translator.<mapname>.output.generatesigneddecimal=True** and replace

<mapname> with the actual name of the map in which you want to use the plus sign, as specified on the Map Details dialog box (Description field) for that map.

For example, if the name of the map in which you want to use the plus sign is "RomanMap.mxl," replace the <mapname> with **RomanMap**, as shown in the example below:

```
translator.RomanMap.output.generatesigneddecimal=true
```

Note: If a map Description contains spaces, you must amend it in the customer_overrides.properties file by putting a backslash before every space. Thus if the map Description is **Roman Map**, the example statement is:

```
translator.Roman\ Map.output.generatesigneddecimal=true
```

2. Save the **customer_overrides.properties** file.

3. Restart your Gentran Integration Suite instance so the property file changes are recognized.

Field-level Validation Configuration

The Map Editor also enables you to specify whether the plus can be allowed or generated at the field level, on the Validation tab of the appropriate Properties dialog box (Element Properties, Field Properties (for positional fields, variable length delimited data fields, and SWIFTNet fields), CII TFDs, XML Pcdatas, and XML Attributes).

Note: This property is not valid for SQL fields and packed decimal numeric fields.

Using the field-level validation setting enables you to override (for a specific field) any settings that you configured in the customer_overrides.properties file.

Using the Validation Tab

The Validation tab enables you to set validation options for the lowest level map components. To access the Validation dialog box, right-click the map component and select **Properties** to access the appropriate Properties dialog box, and then select the **Validation** tab.

A new section on the Validation tab, Positive Number Format, enables you to specify how the Map Editor should handle the use and generation of a plus sign for the map component whose properties you are editing.

By default, the Positive Number Format section will initially display the field option that you configured on the **Standard Formats** tab of the Preferences dialog (see *Field-level Validation Configuration* on page 51), and you can change the parameter to a setting that will only affect the map component you are currently accessing (when the data-type of the map component is Number). If you do not configure the Standard Formats tab, the default is **Use properties file**.

Configuring the Positive Number Format for a Specific Map Component

Within Map Editor you can override the setting on the Preferences dialog box (Standard Formats tab) and customer_overrides.properties file setting, and configure the use of the plus sign for positive number format for a specific field, as follows:

1. From the Map Editor, right-click the map component and select **Properties** to access the appropriate Properties dialog box.

2. Click the **Validation** tab.

3. Select the appropriate option, as follows:

    ◆   **Use properties file setting**—specifies that the rule or rules configured in the customer_overrides.properties file should be followed for this map component

    ◆   **Don't allow/generate '+' prefix**—specifies that regardless of whether the Preference dialog box (Standard Formats tab) or the customer_overrides.properties file contains a rule or rules to allow the "+" prefix, the plus sign will not be allowed for this map component

    ◆   **Allow/generate '+' prefix**—specifies that regardless of whether the Preference dialog box (Standard Formats tab) or the customer_overrides.properties file indicates that the "+" prefix is not allowed in maps, the rule or rules will be overridden and the plus sign will be allowed to be used and generated for this map component

---

Note: By default, the Positive Number Format section will display the global option that you configured on the **Standard Formats** tab of the Preferences dialog (see *Field-level Validation Configuration* on page 51), and you can change the parameter to a setting that will only affect the map component you are currently accessing (when the data-type of the map component is Number).

---

4. Click **OK** to save changes.

Standard Format Preferences Configuration

Gentran Integration Suite provides you with an optional method of specifying the default that will be used for new numeric fields, which can then be modified (see *Field-level*

Validation Configuration on page 51 for more information on how to modify this parameter). This Preferences dialog box (Standard Formats tab) specifies the initial value that will be used for the field-level properties dialog box in both existing and new maps.

The Map Editor Preferences dialog box enables you to set global default values for the Map Editor dialog boxes. To access the Preferences dialog box, select **Options > Preferences**.

The Standard Formats tab of the Preferences dialog box enables you to specify the six- and eight-character formats in which date fields will be interpreted. These defaults are used when documents are initially loaded from a standard. A new section, Positive Number Format Default, enables you to specify the initial value will be used on the field-level configuration dialog boxes.

Configuring the Positive Number Format Defaults for Field-Level Configuration

Within Map Editor you can specify the default value for the use of a plus sign for numeric field, as follows:

1. From the Map Editor **Options** menu, select **Preferences**.

2. Click the **Standard Formats** tab.

3. Select the appropriate option, as follows:

   - **Use properties file setting**—specifies that the rule or rules specified in the customer_overrides.properties file is followed

   - **Don't allow/generate '+' prefix**—specifies that regardless of whether the customer_overrides.properties file contains a rule or rules to allow the "+" prefix, the plus sign will not be used for the field

   - **Allow/generate '+' prefix**—specifies that even if the customer_overrides.properties file indicates that the "+" prefix is not allowed in maps, the rule or rules will be overridden and the plus sign will be used for the field

4. Click **OK** to save changes.

Using a Date/Time Field

A date/time field contains a date or time value. If you specify that a field is a date/time field, you must also specify how the date or time value is formatted.

The following table lists the valid date and time formats:

Note:	Additional date and time formats can be created and added to the Map Editor.

Format	Description
YYMMDD	Two-digit year, two-digit month, two-digit day

Format	Description
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)
DDMMMYY	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)
DDMMMYYYY	Two-digit day, three-letter abbreviation of the month, four-digit year (example: 04JUL2003)
MMMDDYYYY	Three-letter abbreviation of the month, two-digit day, four-digit year (example: JUL042003)
YYDDD	Last two digits of year, three-digit Julian day (example: 99349 for the 349th day of 1999)
DDDYY	Three-digit Julian day, last two digits of year (example: 34999)
YYYYDDD	Four-digit year, three-digit Julian day (example: 1999349)
DDDYYYY	Three-digit Julian day, four-digit year (example: 3491999)
YY/MM/DD	Last two digits of year, separator, two-digit month, separator, two-digit day (example: 99/12/05)
DD/MM/YY	Two-digit day, separator, two-digit month, separator, last two digits of year (example: 05/12/99)
MM/DD/YY	Two-digit month, separator, two-digit day, separator, last two digits of year (example: 12/15/99)
YYYY/MM/DD	Four-digit year, separator, two-digit month, separator, two-digit day (example: 1999/12/15)
DD/MM/YYYY	Two-digit day, separator, two-digit month, separator, four-digit year (example: 15/12/1999)
MM/DD/YYYY	Two-digit month, separator, two-digit day, separator, four-digit year (example: 12/15/1999)
YY/MMM/DD	Two-digit year, separator, three-letter abbreviation of the month, separator, two-digit day (example: 99/JUL/20)
DD/MMM/YY	Two-digit day, separator, three-letter abbreviation of the month, separator, two-digit year (example: 20/JUL/99)

Format	Description
MMM/DD/YY	Three-letter abbreviation of the month, separator, two-digit day, separator, two-digit year (example: JUL/20/99)
YYYY/MMM/DD	Four-digit year, separator, three-letter abbreviation of the month, separator, two-digit day (example: 2003/JUL/25)
DD/MMM/YYYY	Two-digit day, separator, three-letter abbreviation of the month, separator, four-digit year (example: 25/JUL/2003)
MMM/DD/YYYY	Three-letter abbreviation of the month, separator, two-digit day, separator, four-digit year (example: JUL/25/2003)
YY/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)
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)
ISO-8601	YYYYMMDDTHHMMSS.mmmZ 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)
ISO-8601 Date	ISO-8601 date format: YYYYMMDDZ, Four-digit year, two-digit month, two-digit day, Z (Zulu time) indicator (example: 20031209Z)
XSDTIME	XSD:Time format: HH:MM:SS[.sss][+-HH:MM]. HH is hours MM is minutes SS is seconds .sss is fractional sections and is optional. If present, it can be .s (tenths), .ss (hundredths), or .sss (thousandths). +-HH:MM is the timezone difference and is optional. If present, + or - will be present followed by HH:MM where HH is hours and MM is minutes. Long example: 14:30:30.050+05:00 Medium example: 14:30:30.05 Short example: 14:30:30

Format	Description
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-MM-DDTHH:MM:SS	Four-digit year, separator, two-digit month, separator, two-digit day, T represents a blank separator, two-digit hour, separator, two-digit 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)

To specify a date/time field:

1. In the Map Editor, double-click an existing field or create a new one.

2. In the appropriate **Properties** dialog box, click the **Validation** tab.

3. From the data-type list, select **Date/Time**.

    Date/Time indicates whether the field is a date or time.

4. From the data format list, select the appropriate date or time option.

5. Click **OK** to save your choice.

Completing a Map

Completing a map can involve creating simple links (or using the Autolink function), compiling the map, compiling an XML encoder object, printing a mapping report, and testing the map.

For more information, see:

✦   *Creating Simple Links* on page 57

✦   *Using Autolink in the Map Editor* on page 58

✦   *Compiling a Map* on page 59

✦   *Compiling Maps Using the Command Line* on page 60

✦   *Saving and Compiling Maps as the Sterling Integrator Map Type from the Command Line* on page 61

✦ *Compiling an XML Encoder Object* on page 63

✦ *Printing a Mapping Report* on page 64

✦ *Testing a Map* on page 64

Creating Simple Links

The Link function enables you to map a field from the input side of the map to a field on the output side of the map. The link between two map components is represented visually with a connecting line.

Note: To increase the likelihood that the links in your maps are valid, in the Preferences dialog box Confirmations tab, select the **link objects at different levels** and **link objects with different maximum usages** confirmations. See *Customizing Confirmations* on page 29 for more information.

The Map Editor uses the following linking rules:

✦ Link map components with the same maximum usages (loops).

✦ Link map components at the same mapping level. For example, a map header level map component to a header level map component, detail level to detail level.

✦ Link map components with the same data type. For example, map a string field to a string field, a numeric field to a numeric field.

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

Note: You must have at least one direct link (using the Link function) to every record on the output side of the map, so the translator can create the record. This link must pass data from the input side of the map to create an output record.

To link two fields:

1. From the Map Editor **Functions** menu, select **Link**.

2. Select a field on the input side of the map. The cursor changes to a Link arrow:

LINK

3. Select the field on the output side of the map that the data from the field selected on the input side of the map is mapped to.

A line connects the two fields to visually represent the link.

If a line does not appear when the two fields are linked, check the **Preferences** dialog box. In the **Links** tab of the **Preferences** dialog box, select **Show links to or from all visible elements**.

Using Autolink in the Map Editor

The Autolink function automatically creates links between input and output fields that have the same name or which contain logically equivalent business data. This functionality can be used regardless of which format you have selected for the input and output sides of your map.

Just like with the Link function, the link between two map components is represented visually with a connecting line. See *Creating Simple Links* for more information on the Link function.

Note: To increase the likelihood that the links in your maps are valid, in the Preferences dialog box Confirmations tab, select the link objects at different levels and link objects with different maximum usages confirmations. See *Customizing Confirmations* for more information.

The Map Editor uses the following linking rules:

✦ Link map components with the same maximum usages (loops).

✦ Link map components at the same mapping level. For example, a map header level map component to a header level map component, detail level to detail level.

✦ Link map components with the same data type. For example, map a string field to a string field, a numeric field to a numeric field.

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

Note: You must have at least one direct link (using the Link or Autolink function) to every record on the output side of the map, so the translator can create the record. This link must pass data from the input side of the map to create an output record.

When you run Autolink, by default a link is created between any field on the input side of the map that has the same name as a field on the output side of the map. Additionally, there is parameter on the Name tab of the lowest level map components (such as fields and elements), Business Name, that the Autolink function also uses to match fields on the input and output sides of the map if you choose to do so (and if a field on the input side of the map has the same Business Name as a field on the output side of the map, Autolink establishes a link between the two). The Business Name parameter enables you to add another name to each field to indicate the business data that the field contains.

Best Practice

Use Autolink *before* you create manual links between the input and output sides of your map, rather than running Autolink after you have already established links.

Running Autolink

To use Autolink:

---

1. From the Map Editor **Functions** menu, select **Auto Link**.

2. Choose whether to autolink by Field Name or by Business Name and click **OK**.

   Map Editor automatically links all the fields on the input and output sides of the map that have either the same Name or the same Business Name.

Note: If a line does not appear when the two fields are linked, check the **Options > Preferences** dialog box. In the **Links** tab of the **Preferences** dialog box, select **Show links to or from all visible elements**.

Compiling a Map

The Compile function compiles the map. The map that you create using Gentran Integration Suite is a source map (.mxl or .map extension), which is the graphical representation of the data you are mapping. When that source map is compiled, the result is a translation object (.txo extension), which is what the translator uses to translate data.

You use the Compile function after the map is completed and saved.

To compile a map:

1. From the Map Editor **File** menu, select **Compile**.

   The Translation Object Name dialog box opens.

2. In the **File name** box, type the name of the translation object. Use the default .txo extension. Do not use spaces or apostrophes in the name.

Caution: Do not overlay the source map with the translation object by erroneously selecting the .mxl or .map extension for the compiled translation object. To save a map as a .map file (save the source file), select **File > Save As** and then select Source Maps (*.map) from the **Save as type** list.

A good practice is to name the translation object (.txo extension) with the same file name as the source map (.mxl or .map extension). Preserving the same file name (with different extensions) means that the relationship between the source map and the translation object remains evident.

Note: To save your source map as an XML file (.mxl file extension), you must have the Microsoft XML Core Services (MSXML) 4.0 installed on the same computer as Map Editor. If you do not have Microsoft XML Core Services (MSXML) 4.0 installed, you cannot save source maps as .mxl files and must use the .map extension.

---

3. In the **Save in** list, change the drive\folder where the translation object is stored, if necessary.

4. Click **Save** to compile the map.

Note: A progress dialog box displays and updates during the compilation process.

---

5.   Verify that no errors occurred and click **OK**.

The date that the map was compiled on is automatically loaded into the Compiled on box in the Map Details dialog box.

6.   From the **File** menu, select **Save** to save the source map with the Compiled on date.To save a map as a .map file, select **File > Save As** and then select **Source Maps (*.map)** from the **Save as type** list.

> **Note:**   Prior to opening an .mxl (XML-formatted) file, the Map Editor verifies that you have the Microsoft XML Core Services (MSXML) 4.0 installed on the same computer as Map Editor. If you do not have the Microsoft XML Core Services (MSXML) 4.0 installed, the Map Editor cannot save or load .mxl source files.

Compiling Maps Using the Command Line

The Map Editor enables you to automatically compile a single map (or a group of maps located in a specified folder) from the command line.

> **Note:**   The compiled translation objects are written to same folder as the map source files.

Syntax

The command line syntax (from the folder where the map or maps you want to compile are located) follows:

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

> **Note:**   You can use either - or / to designate the compile (c or C) option. The [mapSourceFile] is either a single map or a map file name that contains a wildcard character (*). Using the wildcard character causes MAPPER.EXE to compile all maps in the designated folder with that file name pattern. MAPPER.EXE does not recurse through subfolders.

Example 1

In this example, all maps in the GentranIntegrationSuite folder are compiled.

```
mapper.exe -c c:\GentranIntegrationSuite\*.mxl
```

Example 2

In this example, all maps with names beginning with pet in the GentranIntegrationSuite folder are compiled.

```
mapper.exe -c c:\GentranIntegrationSuite\pet*.mxl
```

Saving and Compiling Maps as the Sterling Integrator Map Type from the Command Line

The Map Editor also enables you to automatically save and compile a single map (or a group of maps located in a specified folder) as a different map type from the command line. This function gives you a quick and easy alternative to having to manually open a non-Sterling Integrator-type map in the Map Editor, access the **Map Details** dialog box, change the map type to Sterling Integrator, and then save and compile the map.

Note:	This option is very flexible; you may use it to, for example, save and compile an Export map created in Gentran:Server for Windows (or maps created using Gentran:Server for UNIX with the Application Integration MAPPER.EXE) to the Sterling Integrator map type.

Tip:	The compiled translation objects are written to same folder as the map source files. Therefore, always save a copy of your original maps to another folder prior to running any command line option, in case you make an error in the syntax.

Syntax

The command line syntax (from the folder where the map or maps you want to compile are located) follows:

```
mapper.exe -c -s [numeric map type] [map filenames]
```

Note:	You can use either - or / to designate the compile (c or C) and save (s or S) options. The [map filenames] is either a single map or a map file name that contains a wildcard character (*). Using the wildcard character causes MAPPER.EXE to compile all maps in the designated folder with that file name pattern. MAPPER.EXE does not recurse through subfolders.

Numeric Map Types

The available numeric map types are listed in the following table:

Note: If you change a map type to Sterling Integrator (map type 24 in the following table), you can then, if necessary, use this command line save function to change the map type to any other map type listed in the following table (if, for example, you want to change the map back to its original type). However, if you compiled a map using a later version of the translator, it will not be backward-compatible. That is, if you use the Gentran Integration Suite version 2.2 translator to save and compile a map (Sterling Integrator map type), and then you save (with the same 2.2 translator) and change the map to a type used in Gentran:Server for Windows or Gentran:Server for UNIX, you will not be able to use the map with a version of Gentran:Server that uses a previous version of the translator.

Numeric Map Type	Map Type
0	Import
1	Export
2	Print
3	Screen Entry
4	Turnaround
5	Transaction Build
6	Transaction Break
7	Functional Group Build
8	Functional Group Break
9	Interchange Build
10	Interchange Break
11	Functional Acknowledgement Inbound
12	Functional Acknowledgement Outbound
13	System Import Header
14	Direct Map
15	Standard to Standard
16	Standard to Application
17	Application to Application
18	Application to Standard
19	XML to Application
20	Application to XML

Numeric Map Type	Map Type
21	XML to Standard
22	Standard to XML
23	XML to XML
24	Sterling Integrator

Example 1

In this example, all maps in the GentranIntegrationSuite folder are saved and compiled as the Sterling Integrator map type.

```
mapper.exe -c -s 24 c:\GentranIntegrationSuite\*.mxl
```

Example 2

In this example, the PET_810.map (in the GentranIntegrationSuite folder) is saved as the Sterling Integrator map type.

```
mapper.exe -s 24 c:\GentranIntegrationSuite\PET_810.map
```

Compiling an XML Encoder Object

Gentran Integration Suite typically translates files using translation objects. It can also translate documents into XML using an XML Encoder object (which is also referred to as a *Lightweight Translation Object*). The standard map has the extension .txo; the XML encoder object has the extension .ltx.

An XML encoder object converts the following file formats into XML:

✦ EDI
✦ Positional
✦ Variable-length-delimited
✦ CII

For an XML encoder object, you do not map fields between formats. Instead, you select input or output and compile a map into an XML encoder object.

XML encoder objects send data to a decision engine where the data is directed and processed according to predefined factors.

All elements in XML encoder object output correspond to the blocks and fields of the original file. Each element has the same name as the block or field that it corresponds to in the original file.

To create an XML encoder object:

1. In the Map Editor, select the root element on the side from which you want to create an XML encoder object.

2. Right-click the root element and select **Compile XML Encoder Object**.

3. In the **XML Encoder Object Name** dialog box, specify the location and name of the XML encoder object. Do not use spaces or apostrophes in the name.

4. Click **Save**. A message indicates that the XML encoder object was compiled.

Printing a Mapping Report

The Print function enables you to print a mapping report for the current map. You can also print a mapping report to a file by selecting the **Print to File** option from the **File** menu.

> **Note:**   You can also create a Map Report that contains information about all your maps and an EDI Translation Detail report that contains EDI translation data organized by envelope level (transaction, group, or interchange) for a specified time period.
>
> You can also create Translation Status and EDI Compliance reports, which enable you to view any errors or warnings and provide actions to complete to reconcile errors or warnings.

To print a mapping report:

1. From the Map Editor **File** menu, select **Print**.

2. Select the reports that you want to print by selecting the appropriate check boxes in the Report section.

3. Click **OK**.

4. In the **Print** dialog box, set the appropriate options and click **OK** to print.

Testing a Map

The Map Test feature enables you to remotely test a map (.txo) on a client machine prior to checking the map in to the Gentran Integration Suite server. This feature executes the map through the translator and returns results to you in the Web browser and text editor configured as the default for the machine, so you can make any necessary changes prior to checking in and using the map.

> **Note:**   You can turn off the Map Test service to prevent users who have access to a trading partner's Gentran Integration Suite system from attempting to use the Map Test feature to run translation on that trading partner's system. Turning off the Map Test service prevents the possible execution of JDBC maps which could access production data. To turn off the Map Test service, access the **translator.properties** file and set the **maptest.MaptestServiceEnabled** property to **False**.

To test your map:

1. In Map Editor, open the map.

2. If the map accesses a database (for example, JDBC or ODBC), configure the map to point to a non-production database. Any map that accesses a database may change the database; for the map test, you should ensure that the map is not changing a production database.

3. Ensure that the map is completed and compile the map. See *Compiling a Map* on page 59 for more information.

4. Select **Functions > Map Test**.

5. In the **Login Server and port** box, type the Gentran Integration Suite server name and *dashboard* port in the format [Gentran Integration Suite server name]:[dashboard port]. The port number required for this parameter is *not* the base port for the install but rather the dashboard port. The dashboard URL and port are typically listed in the UNIX shell after Gentran Integration Suite is successfully started. This box cannot be left blank.

   Note: The dashboard port is displayed to your administrator when your Gentran Integration Suite server is started. If you specify your base port, the Map Test will fail.

6. In the **Login Server User Name** box, type the Gentran Integration Suite user name that is used by the Map Editor to access the Map Test service.

7. In the **Login Server User Password** box, type the Gentran Integration Suite password that is used by the Map Editor to access the Map Test service.

8. In the **Proxy Server and port** box, type the proxy server name and port for the client machine in the format [proxy server name]:[port].

   Note: This parameter is only necessary if you are executing translation beyond a system firewall.

9. In the **Translation Object** box, type the name of the compiled translation object (or click **Browse**, select the translation object, and click **Open**). The path you supply must be valid and the translation object extension must be .txo. This parameter is mandatory.

10. In the **Data File** box, type the name of the data file you wish to use when translating data with the test map (or click **Browse**, select the data file, and click **Open**). If you supply a path, it must be valid.

11. Click **Run Test**. While the test is running, this state is indicated on the dialog box. The Map Editor forwards the translation object and data file to the Gentran Integration Suite system where it is translated. Results are returned in the Web browser and text editor configured as the default for the machine, reporting success or any errors encountered.

    Map Editor also returns the translated data (if present, as a .txt file), an XML-formatted translation report (which is empty if no errors occurred), and the

file/path location of the where Map Editor stored the data and translation report. If there is no translation report, Map Editor returns a file stating that no translation report is available. The Map Editor also returns the location of the files in a dialog box, to enable you to save them. These files are overwritten each time the Map Test feature is used. On the client machine, the translation report and output data are written to the directory where the translation object is located.

> **Note:** If you have not configured your operating system to specify default programs to open .txt and .xml files, the test result files may not be automatically displayed. If this is the case, however, you can locate and open the files using the file/path location returned in the post-test message box.

If Map Editor is unable to test the map, it displays an error message noting the problem encountered, which enables you to correct the specified error and re-test the map.

Chapter 2

Mapping EDI Documents

The Map Editor enables you to map to and from electronic data interchange (EDI) documents.

This section covers the following topics:

✦ About the EDI Data Format

✦ Creating an EDI Layout from an EDI Standard

✦ Activating Map Components

✦ Converting to Another Standards Version

✦ Verifying EDI Delimiters

✦ Using Auto Trim

✦ Defining and Modifying Relational Conditions

✦ Using Loop Start and Loop End Segments

✦ Using Binary Data Segments

✦ Insert Segment or Transaction Set from Standards

✦ Using the Generate UBFI Function

✦ Setting Up a Syntax Record for an EDIFACT UNA Segment

About the EDI Data Format

The Map Editor generates an EDI layout for you using the standard (agency), version, and transaction set you select. Included in the EDI file are groups, segments, composites, and elements that are defined by the standards agency for the version of the document you select.

You can modify the Map Editor-generated EDI file by modifying the properties of the map components, and using the Promote, Split, Copy, Cut, and Paste functions.

However, if you want to use a specialized version of an EDI standard that is not available in the Gentran Integration Suite standards database, it may be appropriate for you either to load an EDI file definition or to define the EDI file yourself. Alternatively, you could use an EDI standard which is similar to what is required and customize it yourself.

Whether the Map Editor generates the EDI file or you load or define it, the EDI map components that you use depend on the map you are creating. The map components can include the standard, version, and transaction set (document) selected as well as the groups, segments, composites, and elements your company requires. Determine which map components you are using before generating or defining an EDI file.

For more information, see:

✦ *EDI Components* on page 68

EDI Components

The following table lists the components that make up the EDI layout in the Map Editor, the icons that represent the components, and descriptions of the components. For information about adding a map component to a layout, see Chapter 1, *Map Editor Basics*. For information about the properties of these map components, see Appendix B, *Map Editor Properties*.

Component	Icon	Description
EDI root element		The *EDI root element* represents the EDI document that Gentran Integration Suite is mapping. At the EDI file root element, you define delimiters and syntax records. It is a group and can contain groups and segments.
Group		A *group* is a looping structure that contains related segments and groups that repeat in sequence until either the group data ends, or the maximum number of times that the loop is permitted to repeat is exhausted. Groups are defined by the EDI standards. A group that is subordinate to another group is a subgroup (and corresponds to a nested looping structure, a loop within a loop). When a group contains an extended rule or a standard rule, an asterisk appears to the right of the group icon.
Segment		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. **Note:** If a segment occurs more than once in a map, it is identified by its name <ID>. The second and subsequent occurrences are identified by <ID>:*n*, where *n* is the number of the occurrence in the map.

Component	Icon	Description

Composite		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 and certain ANSI X12 standards). A composite can occur once or repeat multiple times.
		
		Note: If a composite occurs more than once in a map, it is identified by its name <ID>. The second and subsequent occurrences are identified by <ID>:*n*, where *n* is the number of the occurrence in the map.
		
		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 enable 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.
		
		Example:
		Your delimiters are set as follows:
		Segment ~
		Element *
		Tag *
		Sub Element :
		Repeating Ele ^
		
		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~`

Component	Icon	Description
Element		An *element* is the smallest piece of information defined by the EDI standards. An element can have different meanings depending on the context. Elements are typically not considered to have useful meaning until they are combined into segments.
		An element is the EDI map component that corresponds to a field in other data formats.
		Note: If an element occurs more than once in a map it is identified by its name <ID>. The second and subsequent occurrences are identified by <ID>:*n*, where *n* is the number of the occurrence in the map.
		A *repeating element* is an EDI element with the ability to loop (occur more than once) within a particular EDI segment. To enable 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 translator from mistaking the repeating elements for typical elements.
		Example:
		Your delimiters are set as follows:
		Segment ~
		Element *
		Tag *
		Sub Element :
		Repeating Ele ^
		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~`
		When an element has a link performed against it, a red check mark appears over the element icon.
		When an element contains an extended rule or a standard rule, an asterisk appears to the right of the element icon.

Creating an EDI Layout from an EDI Standard

When you create a new map, you can either manually create an EDI layout or you can use a wizard that creates a layout for you based on an EDI standard. The wizard saves you time and effort and minimizes the risk of having an invalid standard format.

To create an EDI layout from an EDI standard:

1. From the Map Editor **File** menu, select **New**.

2. In the **New Map Wizard**, complete the questions in the first window and click **Next**.

Note: Be sure that **Sterling Integrator** is selected in the **What kind of map are you creating** list.

---

3. If you are translating from EDI, in the Input Format window select **Delimited EDI** and click **Customize**. If you are translating from another format, select that format and continue to the next screen.
4. To specify HIPAA, select the **HIPAA Transaction** check box and click **Next**.
5. Select the ODBC data source that contains the standards database and click **Next**.

Note: The default data source names used by Map Editor are HIPAA Standards and Sterling Integrator Standards.

---

6. Select the standards agency, version, and transaction set and click **Next**.
7. Click **Finish** to load the transaction set you selected.
8. If you are translating to EDI, in the Output Format window, select **Delimited EDI** and click **Customize**.
9. To specify HIPAA, select the **HIPAA Transaction** check box and click **Next**.
10. Select the ODBC data source that contains the standards database and click **Next**.

Note: The default data source names used by Map Editor are HIPAA Standards and Sterling Integrator Standards.

---

11. Select the standards agency, version, and transaction set and click **Next**.
12. Click **Finish**. The Map Editor displays the new map in the Map Editor window.

Activating Map Components

When Gentran Integration Suite generates the EDI sides of the map, Map Editor includes all the groups, segments, composites, and elements that are defined by the standard agency for the version of the document you selected. Map Editor activates all the groups, segments, composites, and elements that are defined as "mandatory" (must be present) by the standard. Map Editor does not enable you to deactivate the mandatory groups, segments, composites, and elements.

When translating data, the translator does not process groups, segments, composites, and elements (or records and fields) that are not activated. Therefore, you must activate the groups, segments, composites, and elements that are not defined as mandatory by the standard, but that you have determined that you need to use in mapping. If a segment is received in the input file and is not activated in the map, the translator will not be able to match the segment to a component in the map—and this can cause unexpected results.

When the translator is not able to match a segment in the data to a segment in the map, a warning code (25 – Unknown Block) is issued to the translation report to indicate that a segment was found in the input file that does not match a segment defined on the input side of the map.

> **Tip:** As an alternative to activation, you can use the Auto Trim feature if you have a sample EDI file defined. For more information about using Auto Trim, see *Using Auto Trim* on page 75.

To activate groups, segments, composites, and elements:

1. From the **Functions** menu, select **Activate**.
2. Open (double-click) each group that contains segments or groups that need to be activated.

> **Note:** As an alternative to opening each map component (steps 2, 5, and 7), you can select **View > Expand All** to open every map component. You can also select a map component and select **View > Expand Branch** to open that map component.

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

3. Select each inactive segment that you need to use.
4. Open (double-click) each segment that contains composites or elements that need to be activated.
5. Select each inactive composite that you need to use.
6. Open (double-click) each composite that contains elements that need to be activated.
7. Select each inactive element that you need to use.
8. Once you have activated all the necessary groups, segments, and elements, select Activate from the Functions menu to turn activation mode off.

Converting to Another Standards Version

The Update EDI Version feature enables you to convert the delimited EDI format side of your map to the most recent standards version from the standards database, and to save it to a new map.

The Update EDI Version feature:

✦ Migrates the old map to the new EDI standard version.
✦ Identifies the differences between the required (mandatory) segments, elements, and groups in the old version as compared with the new version.
✦ Inserts the new required segments, elements, and groups into the old map.

✦ Ignores temporary segments and elements.

✦ Provides you with a report that denotes the differences between the two standard versions and lists the segments and elements that are required for the new standard version and were not in the old map, as well as the map components that are not required for the new version.

Note: The report displays a delete indicator (%delete%) next to the field and record names that are no longer used in the new standard. You *must* manually delete field and record names from your map. The Map Editor does not delete these fields and records automatically because there may be links or rules associated with them that would then be lost. When you delete the specified fields and records manually, you are able to see where the information is mapped in the older standard, and then determine how in the map (using the new standard) the information should be mapped.

To upgrade to a newer EDI standards version:

1. In Map Editor, open the map.

2. Select a map component on the side of the map you want to upgrade.

3. Select **Functions > Update > EDI Version**.

4. Select the ODBC data source that contains the Gentran Integration Suite standards database and click **Next**.

5. Select the standard version to which you want to upgrade and click **Next**.

6. Select the appropriate customization options and click **Next**:

   ◆ Insert new items from the old version

   ◆ Field data type and minimum/maximum length (update with the new version)

   ◆ Field conditional relationships (update with the new version)

   ◆ Conditional/Mandatory (update with the new version)

   ◆ Min/Max usage (update with the new version)

7. Type the name of the new map or accept the Gentran Integration Suite default, and click **Next**.

8. Click **Finish** to start the standard version upgrade process and create the new map.

   If the conversion process fails, the new map and report still exist and contain conversion information up to the point of failure. If the conversion process fails, you are made aware of the failure when the new map is not created. If a failure occurs, verify that your standards database is installed correctly. For more information about installing EDI standards, see the documentation on *Downloading the Standards Database*. If the problem still exists, contact Customer Support.

Verifying EDI Delimiters

If you are using an EDI standard that contains composite elements or subelements, you must verify that Gentran Integration Suite is specifying the correct delimiters. *Delimiters* are flags that you define for Gentran Integration Suite to separate specific EDI components.

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

Verifying EDI delimiters in Gentran Integration Suite is required *only* if you are using a standard with composite elements or subelements. However, it is a good practice to verify EDI delimiters for any standard you use.

To verify EDI delimiters:

1. In the Map Editor, right-click the EDI file icon. From the shortcut menu, select **Properties**.

2. Select the **Delimiters** tab to access delimiter options.

3. Verify that the **Specify defaults** check box is selected.

4. Verify the required delimiters for the EDI standard you are using.

> **Note:**   If the delimiters differ from the defaults specified, type either the character or the hexadecimal value in the correct box.
>
> Also, if you are mapping delimited data and the setup outside the map does not enable you to specify the release character, type the value you expect in the **Release Character** box.

5. Are you using an X12, EDIFACT, or TRADACOMS standard on the output side of a map and you want to suppress leading zeros for the EDI elements that use an R-format numeric (including decimal values less than one)?

    ◆ If Yes, select the **Suppress leading zero on Numeric R* format values (e.g., 0.25---> .25)** option.

    ◆ If No, continue with Step 6.

6. Are you using an X12, EDIFACT, or TRADACOMS standard on the output side of a map and you want to pad numeric values (regardless of format) for the EDI elements with leading zeros?

    ◆ If Yes, select the **Pad with leading zero on Numeric values (e.g., 25---> 000025)** option. When you select this option, Numeric values (regardless of format) are padded with leading zeros out to the maximum length of the field.

    ◆ If No, continue with Step 7.

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

Using Auto Trim

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

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

Delimiters for the EDI side of the map must be set to match the delimiters used in the sample EDI file. To set EDI delimiters, see *Verifying EDI Delimiters* on page 74.

Auto Trim Rules

To use Auto Trim, you must define the segment and element delimiters:

✦ If you do not supply the tag delimiter, the Map Editor substitutes the element delimiter for the tag delimiter.

✦ If you want to use composite elements, you must specify the subelement delimiter.

✦ If you want to use release characters, you must specify the release character.

✦ If you want to use repeating elements, you must specify the repeating element delimiter.

For more information about specifying EDI delimiters, see *Verifying EDI Delimiters* on page 74.

Caution: To use the Auto Trim function, the sample EDI file must be compliant and must not contain envelope segments unless the map defines them.

To use Auto Trim:

1. In the Map Editor, right-click the **EDI file** icon (the EDI root element) on the EDI side of the map. Select **Auto Trim** from the shortcut menu.

2. From the **Look in** list, select the drive and folder where the EDI file is stored.

   The default path is Program Files\Sterling Commerce\Map Editor\Source Maps.

3. If necessary, select the type of file that you want to display from the **Files of type** list.

4. From the list, select the sample EDI file.

5. Click **Open** to begin the Auto Trim process.

   When Auto Trim is complete, the Map Editor displays a message indicating whether the process was successful. Click **OK** to acknowledge the message.

Defining and Modifying Relational Conditions

You can use relational conditions to connect fields for syntax or compliance reasons. For example, field A is invalid unless field B is present. If you create a condition that pairs fields A and B, Gentran Integration Suite 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 EDI standard.

Caution: The Map Editor enables you to edit these conditional relationships, but if you do, you may generate a compliance error.

To define and modify field and element relational conditions:

1. In the Map Editor, double-click the field or element you want to modify.

2. In the **Element Properties** dialog box, click the **Conditions** tab.

3. Select the field connection condition from the type of relationship list. Valid values are:

    ◆   Paired/Multiple – If any of the specified fields are present, all fields must be present.

    ◆   Required – At least one of the specified fields must be present.

    ◆   Exclusion – No more than one of the specified fields can be present.

    ◆   Conditional – If the first condition field is present, the rest of the fields must also be present.

    ◆   List Conditional – If the first condition field is present, at least one of the specified fields must also be present.

4. Select the first field from the condition field list.

    This is the field on which the conditional relationship depends if you chose **Conditional** or **List Conditional** from the type of relationship list.

    The condition field list is active only if you chose either **Conditional** or **List Conditional** from the Type of relationship list.

5. From the **Available fields** list, select a field or fields and click **Add**.

    The Map Editor moves the fields to the **Fields used in relationship** list, to include the fields as a part of the conditional relationship.

    The **Available fields** list contains all the fields in the map 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.

Using Loop Start and Loop End Segments

Certain EDI standards use Loop Start (LS) and Loop End (LE) segments. LS and LE segments differentiate between two or more loops of the same type. If the transaction contains LS and LE segments, you must define the LS and LE segments for the loops you are using in the map, according to what side of the map the segment is on, input or output.

When you create a new, customized EDI map, the Map Editor generates the key field for any LS and LE segments. Generating the key field automatically ensures that the loop ID specified in the LS segment matches the data correctly. The LS and LE segments on the output side do not need to be linked in order to be generated. Also for a new, customized EDI map, the Map Editor creates a standard rule on the loop ID so the correct ID is output. If you create an EDI map manually, you will not have these two functions.

Note: Key fields are intended to be used with string fields only. If you use a key field on a numeric field, the result is not guaranteed.

For more information, see:

✦ *Defining an LS Segment for Input* on page 77
✦ *Defining an LE Segment for Input* on page 78
✦ *Defining an LS Segment for Output* on page 79
✦ *Defining an LE Segment for Output* on page 79

Defining an LS Segment for Input

To define an LS segment for input:

1. In the Map Editor, right-click the **loop start** segment. From the shortcut menu, select **Properties**.

2. In the **EDI Segment Properties** dialog box, click the **Looping** tab to access the loop options and verify that the **Loop Start** option is selected.

   For more information about properties, press F1 for Help or see Appendix B, *Map Editor Properties*.

3. If you are creating a map from the standards, go to step 13. If you are creating a map from scratch, click the **Key Field** tab.

   The Map Editor displays the key field options.

4. From the **Field** list, select **Loop Identifier Code**.

   This field list contains all the elements that are contained in the segment and enables 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. Select the **Use constant** option.

   The Matching rules section enables you to access all the constants and code lists currently defined for this map.

6. Click **Edit** (to the right of the **Use constant** list).

7. In the **Map Constants** dialog box, click **New**.

8. From the **Edit Constant** dialog box, in the **ID** box, type the constant identifier.

9. From the **Type** list, select **String**. This is the category of this constant.

10. In the **Value** box, type the value of the constant.

    For an inbound map, this is the loop identifier code that you expect to receive from a trading partner.

11. Click **OK** to add the constant.

12. Select the constant you created from the list.

    The Map Editor matches the selected constant against the loop identifier code.

13. Click **OK** to exit the EDI Segment Properties dialog box.

Defining an LE Segment for Input

To define an LE segment for input:

1. In the Map Editor, right-click the **LE** segment. From the shortcut menu, select **Properties**.

2. In the **EDI Segment Properties** dialog box, click the **Looping** tab to access the loop options and verify that the **Loop End** option is selected.

   For more information about properties, press F1 for Help or see Appendix B, *Map Editor Properties*.

3. If you are creating a map from the standards, go to step 6. If you are creating a map from scratch, click the **Key Field** tab.

4. From the **Field** list, select the **Loop Identifier Code**.

   This list contains all the elements in the segment and enables you to define the Loop End segment by specifying that the loop identifier code matches the value in the **Matching rules** section.

5. Select the constant you created for the LS segment from the list.

   Gentran Integration Suite matches the selected constant against the loop identifier code. The **Matching rules** section enables you to access all the constants and code lists currently defined for this map.

6. Click **OK** to exit the EDI Segment Properties dialog box.

Defining an LS Segment for Output

If you are creating a map from the standards, you do not need to complete this procedure.

To define an LS segment for output if you are creating a map from scratch:

1. In the Map Editor, double-click the **loop identifier code** element in the **loop start** segment.

2. In the **Element Properties** dialog box, select the **Standard Rule** tab to access the standard rule options.

   For more information about properties, press F1 for Help or see Appendix B, *Map Editor Properties*.

3. From the **standard rule** list, select **Use Constant**.

   The Map Editor displays the constant options.

4. Click **Edit**.

   The Map Constants dialog box opens.

5. Click **New**.

6. From the **Edit Constant** dialog box, in the **ID** box, type the constant identifier.

7. From the **Type** list, select **String**. This is the category of this constant.

8. In the **Value** box, type the value of the constant. For an outbound map, this is the loop identifier code that the trading partner is expecting to receive from you.

9. Click **OK** to add the constant.

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

11. From the **constant** list, select the constant that you created.

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

Defining an LE Segment for Output

If you are creating a map from the standards, you do not need to complete this procedure.

To define an LE segment for output if you are creating a map from scratch:

1. In the Map Editor, double-click the **loop identifier code** element in the **loop end** segment.

2. In the **Element Properties** dialog box, select the **Standard Rule** tab to access the standard rule options.

    For more information about properties, press F1 for Help or see Appendix B, *Map Editor Properties*.

3. From the **standard rule** list, select **Use Constant**.

4. From the **constant** list, select the constant that you created.

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

Using Binary Data Segments

Certain EDI standards use binary segments, and Gentran Integration Suite enables you to create and configure map components to process binary data. Gentran Integration Suite now supports the use of binary data (BIN) segments for inbound and outbound ANSI X12 maps. When a BIN segment is used on the inbound side of an ANSI X12 map, the EDI Translation service uses Xpath to extract the binary data into process data, and substitutes the Xpath expression (Process Data/Entire Document Name) for the binary data in the Binary Data element.

Then, if the Binary Data element is linked to the output side of the map, the translator uses the Xpath expression to obtain the binary document from process data and merge it into the transaction. The Insert Segment From Standards dialog enables you to add a segment from the EDI standards.

When you load an EDI standard that contains binary segments from the Standards CD supplied with Gentran Integration Suite, Map Editor automatically creates the required elements.

For more information, see *Binary Data Segment Example* on page 80.

Binary Data Segment Example

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

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

✦ Bin Len, which is for the length (in characters) of the binary segment

✦ Bin Data, which is for the binary data

To use BIN segments for ANSI X12 maps:

1. Right-click the BIN segment and select **Activate** from the submenu to activate the segment.

2. Right-click the BIN segment and select **Properties** from the submenu (the EDI Segment Properties dialog box is displayed).

3. Click the **Special** tab.

4. Select the Binary check box and click **OK**.

> **Note:** 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.

---

5. Double-click the BIN01 element (the Element Properties dialog box is displayed).

6. Select the **Validation** tab.

7. From the data-type list, select **Bin Len** to designate this element as binary length and click **OK**.

8. Double-click the BIN02 element (the Element Properties dialog box is displayed).

9. Select the **Validation** tab.

10. From the data-type list, select **Bin Data** to designate this element as binary data and click **OK**.

Insert Segment or Transaction Set from Standards

The Insert Segment from Standards function enables you to add a segment from the EDI standards to the EDI side of your map. The Insert Transaction Set from Standards function enables you to add a transaction set from the EDI standards to the EDI side of your map.

For more information about installing EDI standards, see the documentation on *Downloading the Standards Database*.

To insert a segment from the EDI standards database:

1. Right-click the EDI map component after which you want to insert the segment and select **Insert > Segment from standards**, or if you want the segment to be the first segment in the map, right-click the EDI File icon and select **Create Sub > Segment from Standards**.

2. Select a **data source name** from which to access the standards and click **OK**.

3. Select the **Agency**, **Version**, **Segment**, and **Release** (for TRADACOMS only) from the lists and click **OK**. The Map Editor inserts the specified segment into your map.

To insert a transaction set from the EDI standards database:

1. Right-click the EDI map component after which you want to insert the transaction set and select **Insert > Transaction set from standards**, or if you want the transaction

---

set to be first in the map, right-click the EDI File icon and select **Create Sub > Transaction set from standards**.

2. Select a **data source name** from which to access the standards and click **OK**.

3. Select the **Agency**, **Version**, **Transaction**, and **Release** (for TRADACOMS only) from the lists and click **OK**. The Map Editor inserts the specified transaction set into your map as a group containing the appropriate map components.

Using the Generate UBFI Function

The Generate UBFI (Universal Batch File Interface) function is available from the Map Editor Functions menu. If you are creating an EDI-to-Positional map or a Positional-to-EDI map, you can click Generate UBFI to automatically generate an exact flat file version of the EDI data (and replace the positional side of the map with this flat file). The Map Editor also automatically links each EDI element and subelement with its corresponding field on the positional side of the map.

> **Note:**   If neither side of the map is positional, the **Generate UBFI** function is not available.

To use the Generate UBFI function:

1. Select the **Positional File Format** icon (or any map component on the Positional side of the map).

2. From the **Functions** menu, select **Generate UBFI**.

    If the Positional side of the map does not contain any map components before you start this function, the translator generates an exact flat file representation of the EDI data and automatically links the corresponding map components on both sides of the map.

3. If the Positional side of the map contains any map components prior to invoking the Generate UBFI function, the translator prompts you with a warning message stating that generating a UBFI format will replace the existing [input | output] format.

    ◆   To proceed, click **Yes**.

    ◆   To cancel the function, click **No**.

Setting Up a Syntax Record for an EDIFACT UNA Segment

For the Map Editor to correctly process an EDIFACT UNA segment, it should *only* be defined as a syntax record; do not define it in the input file format as a separate segment.

To set up a syntax record for an EDIFACT UNA segment:

1. Open the map in the Map Editor.

2. Right-click the **Input File Format** icon and select **Properties**.

3. Select the **Syntax Record** tab.

4. Select the **Use syntax record** check box.

5. Type the values in the following table:

Property	Value
Tag	UNA
Position	1
Length	9
Contains data	**Note:** Do not select this check box.
Release Character Pos	7
Decimal Separator Pos	6
Tag (Delimiter Position)	5
Segment (Delimiter Position)	9
Element (Delimiter Position)	5
Repeating Element (Delimiter Position)	8
Sub Element (Delimiter Position)	4

6. Recompile the map.

7. Save the map.

8. Check the map and translation object in to Gentran Integration Suite.

    See *Checking In Maps* on page 274 for more information about checking maps and translation objects in to Gentran Integration Suite.

Mapping Positional Documents

The Map Editor enables you to map to and from positional data documents.

This section covers the following topics:

✦ About the Positional Data Format

✦ Creating a Positional Map

✦ Importing Positional Maps from Gentran:Server for Windows and Gentran:Server for UNIX

✦ Creating Fields with the Positional Field Editor

✦ Creating Temporary Records and Fields

About the Positional Data Format

If your input or output document is a positional file, you must either define it to Gentran Integration Suite or load a previously created file definition. If you are manually creating the positional layout, remember that each level of a positional file must be created sequentially, and you must create records and groups before you create the subordinate fields.

Note: The transaction data file (TDF) button is disabled in Gentran Integration Suite. TDF is a proprietary Sterling Commerce data format that is supported in the Gentran:Server for Windows product. The TDF format is *not* supported in Gentran Integration Suite.

For more information, see *Positional Components* on page 85.

Positional Components

The following table lists the components that make up the positional layout in the Map Editor, the icons that represent the components, and descriptions of the components. For information about adding a map component to a layout, see Chapter 1, *Map Editor Basics*.

For information about the properties of these map components, see Appendix B, *Map Editor Properties*.

Component	Icon	Description
Positional root element		The *positional root element* represents the positional document that Gentran Integration Suite is mapping. The positional root element defines the characteristics of a positional file, such as delimiters or record length. It is a group and can contain groups and records.
Group		A *group* is a looping structure that contains related records and groups that repeat in sequence until either the group data ends or the maximum number of times that the loop is permitted to repeat is exhausted.
		A group that is subordinate to another group is a subgroup (and corresponds to a nested looping structure, a loop within a loop).
		When a group contains an extended rule or a standard rule, an asterisk appears to the right of the group icon.
Record		A *record* contains related fields. A record can occur once or can repeat multiple times.
		Each positional file layout must have a header record, which is generally the first record. The header record is mandatory and repeats only once (it is not a looping structure).
Field		A *field* is the smallest piece of information defined in the positional file. You map corresponding fields to move data to or from a positional file.
		When a field has a link performed against it, a red check mark appears over the field icon.
		When a field contains an extended rule or a standard rule, an asterisk appears to the right of the field icon.

Creating a Positional Map

Gentran Integration Suite enables you to quickly and easily create maps for the positional format.

To create a map:

1. From the Map Editor **File** menu, select **New**.

2. In the New Map wizard, answer the following questions and then click **Next**.

◆ What kind of map are you creating?

Accept the default, Sterling Integrator.

Caution: Always accept the Sterling Integrator default when you are using the Gentran Integration Suite product.

Note: The other map options enable you to import maps from other Sterling Commerce products in order to convert them to Sterling Integrator-type maps.

◆ What is the name of the map?

Type the unique name of the map. The Map Editor adds the default .mxl extension.

◆ What is your name?

Type your name if it differs from the user name prompted by the New Map wizard. The New Map wizard displays Input Format fields. You must complete the format of the input side of the map. This is the format of the data that is translated by Gentran Integration Suite.

3. In the Input Format window, specify how you want to define the data format by selecting one of the following:

◆ Create a new data format using this standard

Note: The choices available depend on the standard or standards that you downloaded. Use this option if you want the input side of the map to use NACHA ACH (NACHA Automated Clearinghouse), SAP IDocs, or BECS DE (Bulk Electronic Clearing System Direct Entry).

- Select the standard and click **Messages**
- Complete the Map Wizard by selecting the version of the standard, the desired message, the maximum length of the data elements, and whether to build codelists for enumerated attributes.

◆ Load the data format from a saved definition

- Type the path and file name of the saved definition (.ddf or .ifd extension).
- Click **Browse** to display the **Open File Definition** dialog box.

◆ Create a new data format using this syntax

Select a format.

Note: Use this option if you want the input side of the map to be strictly positional (you define the syntax).

---

4. In the Output Format window, specify how you want to define the data format by selecting one of the following:

◆   Create a new data format using this standard

Note: The choices available depend on the standard or standards that you downloaded. Use this option if you want the input side of the map to use NACHA ACH (NACHA Automated Clearinghouse), SAP IDocs, or BECS DE (Bulk Electronic Clearing System Direct Entry).

- Select the standard and click **Messages**
- Complete the Map Wizard by selecting the version of the standard, the desired message, the maximum length of the data elements, and whether to build codelists for enumerated attributes.

◆   Load the data format from a saved definition

- Type the path and file name of the saved definition (.ddf or .ifd extension).
- Click **Browse** to display the **Open File Definition** dialog box.

◆   Create a new data format using this syntax

Select a format.

Note: Use this option if you want the output side of the map to be strictly positional (you define the syntax).

---

5.   Click **Finish** to create the map. The map opens in the Map Editor window.

6.   In the Map Editor, select **File** > **Save** to save the map. Do not use spaces or apostrophes in the map name. To save a map as a .map file, select **File > Save As** and then select **Source Maps (*.map)** from the **Save as type** list.

Importing Positional Maps from Gentran:Server for Windows and Gentran:Server for UNIX

When you import maps into Gentran Integration Suite that were created in Gentran:Server for Windows and in Gentran:Server for UNIX (using the Application Integration MAPPER.EXE), you must change the map type. To do so, open the Map Details dialog box (from the Edit menu) and change the Map Function to *Sterling Integrator*. Or, for more information about using the command line compile to change the map type to Sterling

Integrator outside of the Map Editor, see *Saving and Compiling Maps as the Sterling Integrator Map Type from the Command Line* on page 61.

Note: For more information about how to migrate maps from Gentran:Server for Windows to Gentran Integration Suite, see Appendix C, *Map Editor Migration Information* . For more information about importing Gentran:Server for UNIX maps directly into Gentran Integration Suite, see Appendix F, *Map Conversion*.

For more information on other information related to importing maps from Gentran:Server for Windows or Gentran:Server for UNIX, see:

✦ *About Record Delimiters* on page 89
✦ *About Decimal Points* on page 89
✦ *About Double-Byte Character Sets* on page 90

About Record Delimiters

If you are importing maps from Gentran:Server for Windows that use end-of-line as the record delimiter and are converting these maps to Gentran Integration Suite maps, you must reset the record delimiters if you want the map to run correctly on both Windows and UNIX operating systems.

To set the correct record delimiters:

1. In the Map Editor, open each positional file.

2. Open the **Positional File Properties** dialog box.

3. In the **Record** tab, delete the **Record Delimiters 1** and **2**. This resets the delimiters to the Gentran Integration Suite default, which works on both the Windows and UNIX operating systems.

4. Click **OK**. You must click OK, even if the delimiter fields are empty, to reset the delimiters.

5. Save the map.

6. Compile the map.

   The end-of-line delimiters are now correctly set to the Gentran Integration Suite default, which complies with the Windows and UNIX operating systems.

About Decimal Points

If you need to specify that each decimal point in your data is defined and generated as a something other than the default period (.), you can do so on the Decimal Point tab.

To change the default decimal point setting:

1. In the Map Editor, open each positional file.

2. Open the **Positional File Properties** dialog box.

3. In the **Decimal Point** tab, select the **Define Decimal Point** check box.

---

4. In the **Decimal Point Character** box, type the value you want Gentran Integration Suite to use as a decimal point—for example, a comma (,). This resets the decimal point default to use what you specified instead of a period.

5. Click **OK**.

6. Save the map.

7. Compile the map.

    The decimal point setting is now correctly set to use your specification.

About Double-Byte Character Sets

If you are working with a positional map from Gentran:Server for Windows that uses double-byte character sets, and you are converting that map to Gentran Integration Suite, you must make some adjustments. When Gentran:Server reads and validates data, the Gentran:Server translator counts bytes. When Gentran Integration Suite reads and validates data, the Gentran Integration Suite Translation service counts characters. So, a double-byte character is counted as two positions by Gentran:Server and one position by Gentran Integration Suite.

Therefore, for any field that contains a double-byte character set in a positional field, you must adjust the length of the field to account for the difference. The Map Editor automatically adjusts the minimum and maximum fields on the Validation tab for you. For example, if a field has a maximum length of 32 specified in a Gentran:Server map, then in the Map Editor the field maximum is 16.

Creating Fields with the Positional Field Editor

Each record you create contains logically related positional fields. These fields define the structure and content of the data to Gentran Integration Suite.

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

Do not define fields if their only purpose is to explicitly contain the record tag. Gentran Integration Suite takes the tag that you define in the record into account when the automatic sequencing (Auto Position) function is used. Instead, define the record tag on the **Positional Record Properties** dialog box.

Note: You must either specify the start position of each field or use the Auto Position function; otherwise, the translator will not be able to read or write the record correctly.

To create the positional fields for a record using the Positional Field Editor:

1. In the Map Editor, right-click the positional record. From the shortcut menu, select **Edit Fields**.

    The Positional Field Editor dialog box opens.

2. Is the field you are creating the first field in the record?

    ◆   If Yes, click **New** and continue with the next step.

    ◆   If No, select the field that precedes the field you are creating in the record layout and click **New**.

        A select bar appears in the Fields section where the new field is positioned.

        Complete the field values in the **Field Details** section.

3. In the **Name** box, type the field name.

Note:	Do not use spaces or hyphens (-) in the field name. You can use the underscore (_) to separate words.

    Each positional field must have a unique name. It is useful to tag the end of the field that occurs in multiple records with a suffix that identifies the record that contains the field.

4. Do you want to designate the field as required?

    ◆   If Yes, select the **Mandatory** check box.

    ◆   If No, continue with the next step.

5. In the **Description** box, type a description of the field.

    The description briefly explains the field to enable you to differentiate it from similar fields.

6. From the **Data Type** list, select the type of the field.

    Valid values are:

    ◆   String – Alphanumeric field

    ◆   Number – Numeric, real, overpunched, or packed field

    ◆   Date/Time – Date or time field

7. From the **Format** list, select how the field is formatted. The choices for this field depend on the type of field you select from the Data Type list.

    ◆   If you select Number or Date/Time in the Data Type list, you can select the data format from the Format list.

    ◆   If you select String from the Data Type list, you must type a syntax token to denote that this field must be formatted as the specified syntax token dictates (the default syntax token is X).

8. Do you want to indicate the exact position of the field in the record?

* If Yes, type the starting position of the field in the **Start Pos** box.

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

* If No, continue with the next step.

  See step 13 for more information about the Auto Position function.

9. In the **Min Length** box, type the minimum length of this field.

10. In the **Max Length** box, type the maximum length of the field.

11. Click **New**.

    The Map Editor adds the field and creates a new field with blank values ready for you to identify, positioned after the newly added field.

12. Create the rest of the fields according to your record layout.

    To stop adding fields, click **Delete**.

13. After adding the last field, do you want Gentran Integration Suite to automatically position the fields in the record?

    > **Note:**   You must either specify the start position of each field or use the Auto Position function; otherwise, the translator will not be able to read or write the record correctly.

* If Yes, click **Auto Position**.

  Auto Position automatically calculates the start position in the record of each field, using the criteria that each field is positioned directly after the previous field and is of the length specified in the Max Length box. Click **Yes** to acknowledge the warning message that fields are sequenced in order.

    > **Note:**   The Auto Position function is valid only if you define a record tag and if you define every field in the record in the sequence that each field occurs.

* If No, continue with the next step.

14. Are you completely finished adding fields to the record?

    * If Yes, click **Close**.

    * If No, repeat steps 2 - 13.

    Complete this procedure to add fields to the other records you defined.

Creating Temporary Records and Fields

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

For more information, see:

✦ *When to Use Temporary Records and Fields* on page 93
✦ *Where to Use Temporary Records and Fields* on page 93

When to Use Temporary Records and Fields

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

✦ If you must map from a repeating map component to a single map component, or vice versa.

✦ If the Input hierarchical level in a map does not match the Output hierarchical level.

✦ If you only want to populate an Output field with data if a specified qualifier is used or if specific criteria is met.

✦ If you must use extended rules in addition to creating temporary records and fields.

> **Note:** When you use extended rules with temporary groups and records, you must use indexes when you are moving data to a looping (multiple occurrence) group or record. For more information about using indexes, see Appendix D, *Using Indexes in the Map Editor and Translator*.

Where to Use Temporary Records and Fields

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

The following restrictions apply to temporary records:

✦ You must use a record tag that you would never receive in your Input file, and the recommended default is **$$$**.

✦ If you are creating a temporary record for an XML file, use the tag **XXX**, because **$$$** is not permitted as a tag in XML.

✦ The translator does not run standard or extended rules on temporary records.

Example

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

Step 1

First, you must create a temporary storage record and fields on the EDI side of the map that do not repeat. The temporary record is located outside the N1 Group at the same hierarchical level as it, and has a maximum repeat number of 1 (that is, it does not repeat). Then you will create the appropriate temporary fields.

The following table describes the temporary storage record that you are creating:

Name	Description	Tag	Min Usage	Max Usage
ShipToDet	Ship To Details	$$$	0	1

Sample Procedure

To create the ShipToDet temporary storage record:

1. Select the **N1** group. The two temporary storage segments are located after the N1 group, at the same level.

2. From the **Edit** menu, select **Insert**. From the submenu, select **Segment** (you select **Segment** instead of **Record** because you are creating the temporary record on the EDI side of the map). The EDI Segment Properties dialog box is displayed.

3. In the **Name** box, type **ShipToDet**.

4. In the **Description** box, type **Ship To Details**.

5. Click the **Tag** tab.

6. In the **Tag** box, type **$$$**.

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

7. Click the **Looping** tab.

8. In the **Min Usage** box, type **0** (zero).

> **Note:**   A minimum usage value of zero prevents the translator from reporting an error during compliance checking.

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

10. Create the appropriate temporary fields.

    ◆   SHIPTONAME

    ◆   SHIPTOADDR1

    ◆   SHIPTOADDR2

    ◆   SHIPTOCITY

    ◆   SHIPTOSTATE

    ◆   SHIPTOPCODE

Step 2

Create an On End extended rule for the N1 Group that assigns the necessary information to the temporary fields.

Sample Procedure

To create the On End extended rule:

Note: In this example, the data is written to singly occurring record (maximum usage of 1), so the extended rule does not require an index. For more information about using indexes, see Appendix D, *Using Indexes in the Map Editor and Translator*.

---

1. Right-click the **N1** group to access the shortcut menu.

2. From the shortcut menu, select **Extended Rules** to display the Group Properties dialog box.

3. Select the **On End** option. This specifies that the rule is run when the loop terminates. The translator loads an occurrence of the N1 group (containing the billing or shipping information), and then runs this rule.

4. In the **Editor** list, type the following:

```
IF #0098 = "ST" THEN
BEGIN
$850.#SHIPTONAME = #0093;
$850.#SHIPTOADDR1 = #0166;
$850.#SHIPTOADDR2 = #0166:2;
$850.#SHIPTOCITY = #0019;
$850.#SHIPTOSTATE = #0156;
$850.#SHIPTOPCODE = #0116;
END
```

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

---

5. Click **Compile** to validate the syntax of the extended rule. Every rule in the map is compiled when you compile the translation object, after you complete the map. However, the translator enables you to compile each rule individually, so that you can verify the accuracy of the rule after you create it.

> **Note:** This compiles the rule interactively, and enables you to correct any errors that are generated. Any errors or warnings generated in the compilation process are displayed in the Errors list.

6. Click **OK** to add the extended rule to the N1 group.

Step 3

Link the temporary fields on the Input side of the map with the corresponding Output fields.

Chapter 4

Mapping Variable-Length-Delimited Documents

The Map Editor enables you to map to and from variable-length-delimited documents.

This section covers the following topics:

✦   About the Variable-Length-Delimited Data Format

✦   Creating a Variable-Length-Delimited Layout from a Delimited File

About the Variable-Length-Delimited Data Format

The variable-length-delimited data format enables you to map delimited files, such as comma-separated variable (CSV) files. Variable-length-delimited files must follow these guidelines:

✦   Records are contained on one line of data, terminated by a carriage return *and* a line feed, or just by a line feed.

✦   Fields must be delimited, even if some do not have any data in them.

✦   If a field in the variable-length-delimited file contains a delimiter, you must use quotation marks around the field.

✦   When processing Variable Delimited data, the number of fields received in the data file must be the exact number fields defined for the record in the map.

You can create variable-length-delimited layouts with the assistance of a wizard. By selecting Customize, you instruct the Map Editor to use the file you are mapping to create the basic structure of your layout for you. For more information, see *Creating a Variable-Length-Delimited Layout from a Delimited File* on page 98.

For more information, see *Variable-Length-Delimited Components* on page 98.

Variable-Length-Delimited Components

The following table lists the components that make up the variable-length-delimited layout in the Map Editor, the icons that represent the components, and descriptions of the components. For information about adding or modifying a map component in a layout, see Chapter 1, *Map Editor Basics*. For information about the properties of these map components, see Appendix B, *Map Editor Properties*.

Component	Icon	Description
Variable-length-delimited root element		The *variable-length-delimited root element* represents files that contain data separated by delimiters, such as .csv files.
Group		A *group* is a looping structure that contains related groups or records that repeat in sequence until either the group data ends or the maximum number of times that the loop is permitted to repeat is exhausted.
		When a group contains an extended rule or a standard rule, an asterisk appears to the right of the group icon.
Record		A *record* is one line of data that contains fields separated by delimiters. Each record is separated by a carriage return.
Field		A *field* is one segment of data in a record, separated by delimiters.
		When a field has a link performed against it, a red check mark appears over the field icon.
		When a field contains an extended rule or a standard rule, an asterisk appears to the right of the field icon.

Creating a Variable-Length-Delimited Layout from a Delimited File

When you create a new map, you can either manually create a variable-length-delimited layout or you can use a wizard that creates a layout for you based on the delimited file you select.

The wizard distinguishes among the following data types: real, integer, and strings. The wizard determines the longest field for each field position and sets each field length to the nearest 50, rounding up. For example, if the longest field number two in all of the records in the file is 52 characters long, then the wizard makes the field length for field number two 100.

The wizard can map only one record type at a time. If you want to map a variable-length-delimited file that contains more than one record type, you must map it manually.

To create a variable-length-delimited layout based on a delimited file:

1. From the Map Editor **File** menu, select **New**.

2. In the **New Map Wizard**, complete the questions on the first screen.

   For more information about creating a new map, see Chapter 1, *Map Editor Basics*.

3. If you are translating from variable-length-delimited format, on the Input screen, select **Variable Delimited** and click **Customize**. If you are translating from another format, select that format and continue to the next screen.

4. If you are translating to variable-length-delimited format, on the Output screen, select **Variable Delimited** and click **Customize**. If you are translating from another format, select that format and continue to the next screen.

5. In the **New Variable Length Delimited Data Wizard**, type the name or browse to the location of the delimited file.

6. Specify the field delimiter and quote character, as necessary.

7. If you want the first record to act as a header, select **Include column (field) names from first record** and click **Next**.

8. On the output side, select the type of quote handling and click **Finish**:

Type	Description
Quote fields containing delimiter (default)	Only fields that contain the delimiter are quoted.
Quote text fields and fields containing delimiter	All string fields and all fields that contain the delimiter are quoted.
Quote all fields	All fields are quoted. **Note:** By default, the translator does not quote empty output fields. To enable the translator quote empty output fields, add the line **varDelim.quoteEmptyFields=Yes** to the **translator.properties** file in the Properties directory.

9. Continue with the **New Map Wizard** as directed. When you click Finish, the Map Editor displays the new map in the Map Editor window.

10. If necessary, modify the map so the record has the exact amount of fields that are in the data.

Chapter 5

Mapping CII Documents

The Map Editor enables you to map to and from Japanese Center for Informatization of Industry (CII) data documents.

This section covers the following topics:

✦ About the CII Data Format

✦ Creating a CII Layout from a Standard

✦ Configuring a Loop

✦ About Character Encoding

✦ Relating CII Data Attributes to Map Editor Data Types

✦ Preserving Leading Spaces When Mapping to a Positional Data Format

✦ Importing CII Maps from Gentran:Server for Windows

About the CII Data Format

The *CII data format* provides the Japanese syntax definition for EDI messages. The CII implementation in Gentran Integration Suite is based on the CII Syntax Rule, available in both Japanese and English. The CII Syntax Rule specifies details such as looping structures and data types, but it does not include standard message types. Message types are provided by industry groups. Gentran Integration Suite provides a number of these standard message types in the EDI Standards, which you can download. For more information about the CII standards, see the documentation on *CII Standard Versions*.

For more information, see:

✦ *Using CII with Gentran Integration Suite* on page 102

✦ *CII Format Components* on page 102

✦ *About the CII Message* on page 102

Using CII with Gentran Integration Suite

In Gentran Integration Suite, a CII map that contains an FA, FD, or F9 data tag in extended mode on the output side of the map must have a Use Constant standard rule defined, so that a detail number is generated. A detail number is mandatory for these tags in extended mode.

For CII inbound and outbound processes, you must specify either 8-bit or 16-bit character set in the envelope. For 8-bit character set, you can choose JIS0201, SJIS, or Default, where Default is JIS0201. For 16-bit character set, you can choose JIS0208, JIS0212, SJIS, or Default, where Default is SJIS. If you do not set these two values, the default value will be used in translation.

CII Format Components

The following table lists the components that make up the CII format, the icons that represent the components, and descriptions of the components. For information about adding a map component to a layout, see Chapter 1, *Map Editor Basics*. For information about the properties of these map components, see Appendix B, *Map Editor Properties*.

Component	Icon	Description
CII root element		The *CII File root element* represents the CII document that Gentran Integration Suite is mapping.
Group		A *group* contains related groups and transfer form data (TFDs).
		When a group contains an extended rule or a standard rule, an asterisk appears to the right of the group icon.
TFD		*Transfer form data* (TFD) is a block of data that consists of a tag, a length indicator, and data. The length of the element is always indicated, so delimiters are unnecessary.
		When a TFD has a link performed against it, a red check mark appears over the TFD icon.
		When a TFD contains an extended rule or a standard rule, an asterisk appears to the right of the TFD icon.

About the CII Message

CII headers and trailers are in text, but messages are binary. To read a raw CII message outside of Gentran Integration Suite, you need a hex editor. Using a hex editor, you can see the basic structure of a CII message, which consists of:

✦ Control tags
✦ Transfer form data (TFDs)
✦ Length indicators
✦ Data bytes

A typical TFD consists of a tag, a length indicator, and data. The length of the element is always indicated, so delimiters are unnecessary.

The Gentran Integration Suite Translation service handles hexadecimal data in the range of 0x01 through 0xFF. The Translation service does not handle 0x00, and thus it cannot translate pure binary data such as bitmaps.

About CII Control Tags

The following hexadecimal control tags are used in CII:

Control Tag	Function
0xF0	Starts extended mode
0xF2	Length extender
0xF7	Length indicator
0xF8	Escape indicator
0xF9	Internal segment separator
0xFA/0xFD	Multi-detail header (loop start)
0xFB	Multi-detail return
0xFC	Multi-detail trailer (loop end)
0xFE	Message Trailer

Creating a CII Layout from a Standard

To create a CII layout from a standard:

1. From the Map Editor **File** menu, select **New**.

2. In the **New Map Wizard**, complete the questions on the first screen.

3. If you are translating from CII, on the Input screen select **CII** and click **Customize**. If you are translating from another format, select that format and continue to the next screen.

4. If you are translating to CII, on the Output screen select **CII** and click **Customize**. If you are translating from another format, select that format and continue to the next screen.

5. In the **New CII Wizard**, click **Next**.

6. Select the ODBC data source that contains the standards database.

7. If you want Japanese descriptions, select the check box and click **Next.**

   Note: Descriptions are available only in Japanese.

---

8. Click **Finish**.

9. Continue with the **New Map Wizard** as directed. When you click Finish, the Map Editor displays the new map in the Map Editor window.

Configuring a Loop

In CII terminology, a loop is expressed as a multi-detail. Multi-details are indicated through the use of specific TFDs. A multi-detail header (MDH) is used before the start of a loop, and at the end of each iteration there is a multi-detail return (MDR). The multi-detail trailer (MDT) is located at the end of the loop. The following table shows how a multi-detail is typically configured:

TFD Type	Tag	Properties
Multi-detail header (MDH)	0xFA	One-byte string
MDH	0xFD	Two-byte binary numeric type N0
Multi-detail return (MDR)	0xFB	No data
Multi-detail trailer (MDT)	0xFC	No data

Configuring Key Fields for Multi-Detail Headers

In extended mode, you must identify multi-detail headers with key fields. You must also specify key fields if there is any possibility of ambiguity in the data (for example, if a multi-detail in the CII file layout has no corresponding data).

   Note: Key fields are intended to be used with string fields only. If you use a key field on a numeric field, the result is not guaranteed.

To configure a key field for a multi-detail header:

1. In the Map Editor, double-click or right-click the multi-detail header and select **Properties**.

2. Click the **Key Field** tab.

    For more information about properties, press F1 for Help or see Appendix B, *Map Editor Properties*.

3. Select **Use constant** and click **Edit**.

4. In the **Map Constants** dialog box, type the constant you need.

Tag	Data Type	Range of Values
0xFA	String	A to Z, 1 to 9
0xFD	String	0000 to 65535

About Character Encoding

An encoding system determines the hexadecimal values that represent display characters. The Map Editor uses the default character set of the computer you are working on. The default encoding for Japanese Windows is Shift-JIS although there are several other encoding systems for Japanese characters.

When you are specifying character sets for the CII side of a map, you must indicate one character set each for 8-bit and 16-bit characters in the **CII File Properties** dialog box. Then in the **CII TFD Properties** dialog box, you indicate whether the TFD uses an 8-bit or 16-bit character set. For more information about properties, see Appendix B, *Map Editor Properties*.

Relating CII Data Attributes to Map Editor Data Types

The following table shows how CII data attributes correspond to the data types in the Map Editor:

Description of Data	CII Attribute	Map Editor Data Type
Numeric, implied decimal	9	N0 to N9
Numeric, explicit decimal	N	R0 to R9
8-bit characters	X	X, J, or other single-byte syntax token
16-bit characters	K	K, double-byte syntax token
8-digit date	Y	Date: YYYYMMDD

For more information, see *About Syntax Tokens* on page 106.

About Syntax Tokens

To process Japanese characters, you must use a syntax token that enables non-Latin characters in the data that the map will process. The Map Editor provides the syntax token K, which enables all double-byte Japanese characters.

You can create a syntax token to meet your specific needs. For more information about syntax tokens and creating them, see *Formatting Data in Fields* on page 40. If you create a double-byte character set (DBCS) syntax token, you must configure the syntax tokens on a computer with a DBCS operating system (Japanese, Korean, Traditional Chinese, or Simplified Chinese) or a Windows 2000 system to which you have added font support for one of those languages. If you do not configure syntax tokens, the DBCS button is inactive and you are unable to use it. Syntax tokens cannot be imported.

Preserving Leading Spaces When Mapping to a Positional Data Format

In the EDIFACT and ANSI standards, leading spaces in fields are eliminated. However, in the CII format, according to the CII Syntax Rule, leading spaces are typically preserved (although they can be eliminated).

To preserve leading spaces in a map for translating CII to the positional format:

1. From the Map Editor **Edit** menu, select **Details**.

2. In the **Map Details** dialog box, select **Use Configurable Trimming** and click **OK** to close the dialog box.

3. In each field on the positional side of the map, open the **Field Properties** dialog box, and click the **Position** tab.

4. If the spaces are single-byte:

   a. In the **characters used in empty portions of the map** box, type **SP**.

   b. From the data alignment options, select **beginning of the field**.

5. If the spaces in the field are double-byte and the data is encoded in Shift-JIS (windows codepage 932), for **Enter the character used in empty portions of the field**, type **0x8140** or the appropriate key code combination.

6. Click **OK**.

Importing CII Maps from Gentran:Server for Windows

When you import any map from Gentran:Server for Windows to Gentran Integration Suite, you must open the **Map Details** dialog box (select **Details** from the **Edit** menu) and select **Sterling Integrator** in the **Map Function** field.

Gentran Integration Suite and Gentran:Server for Windows process data differently. When Gentran:Server reads and validates data, the Gentran:Server translator counts bytes. When Gentran Integration Suite reads and validates data, the Gentran Integration Suite Translation service counts characters. So, a double-byte character is counted as two positions by Gentran:Server and one position by Gentran Integration Suite. When you import a map created in Gentran:Server that maps CII (input or output), the Map Editor automatically adjusts the minimum and maximum fields on the **Validation** tab for you. For example, if a TFD has a maximum length of 32 specified in a Gentran:Server map, then in the Map Editor, the TFD maximum is 16.

Mapping SQL Documents

The Map Editor enables you to map Structured Query Language (SQL) documents.

This section covers the following topics:

✦ About the SQL Data Format
✦ Considerations for SQL Mapping
✦ Overview: How to Create Map Objects
✦ Managing the SQL File Format
✦ Managing Statement Records
✦ Managing Cursor Operation Records
✦ Managing Input Records
✦ Managing Output Records
✦ Generating Database Fields
✦ Creating SQL Fields
✦ Checking Database Consistency

About the SQL Data Format

The SQL data format enables you to create a map directly from a database schema, which saves time and ensures that the map is synchronized with the most current version of the database. You can also specify several data sources so that Gentran Integration Suite can query or update multiple databases during translation.

The Map Editor enables you to:

✦ Specify data sources that Gentran Integration Suite uses to query for data or update data in multiple databases in a single translation session.
✦ Add data sources and then test the connection and edit the connection string.
✦ Test queries against data sources and generate result sets.
✦ Generate fields directly from the result sets of queries.

✦ View a model of the database schema, including lists of tables and columns.

✦ Generate fields directly from your database schema.

✦ Check the consistency of the map with your database.

Note: SQL functions cannot be inserted in any other type of map.

For more information, see *SQL Components* on page 110.

SQL Components

The following table lists the components that make up the SQL layout in the Map Editor, the icons that represent the components, and descriptions of the components. For information about adding a map component to a layout, see Chapter 1, *Map Editor Basics* and *Overview: How to Create Map Objects* on page 114. For information about the properties of these map components, see Appendix B, *Map Editor Properties*.

Component	Icon	Description
SQL root element		The *SQL root element* is where you specify the SQL data sources that Gentran Integration Suite uses during translation. It is a looping structure that contains groups, SQL statements, cursor operations, and input and output records that repeat in sequence until either the group data ends or the maximum number of times that the loop is permitted to repeat is exhausted. The SQL root element cannot be referenced by standard rules or links.
Group		A *group* is a looping structure that contains related groups, statement records, cursor operations, and input and output records that repeat in sequence until either the group data ends or the maximum number of times that the loop is permitted to repeat is exhausted. A group cannot be referenced by standard rules or links.
		When a group contains an extended rule or a standard rule, an asterisk appears to the right of the group icon.

Component	Icon	Description
Statement record		A *statement record* represents a unit of SQL, which includes a SQL query (which may or may not return a result set), a command (which does not return a result set), or a stored procedure invocation (which may or may not return a result set). Each statement record is associated with a single data source. If the schema for that data source already exists, Gentran Integration Suite displays all the tables for your reference. To use a statement record, you must first select a data source. If a schema was generated for the selected data source, Gentran Integration Suite displays a list containing all the tables and views. Then you can type a SQL statement, test its validity, and generate a result set of columns returned from the query or stored procedure. This result set is used in the SQL field generator for input records. If you specify the name of a stored procedure instead of a SQL statement, you must select the Stored Procedure option. The statement record cannot be referenced by standard rules, extended rules, or links.
Cursor operation record		A *cursor operation record* contains instructions for the translator on moving through the result set returned by a query to a new record. Each operation record is associated with a single SQL statement record that returns a result set. The translator performs cursor operations as it encounters them while processing the map. The cursor operation record is enabled only on the input side of the map and cannot be referenced by standard rules, extended rules, or links.
Input record		An *input record* contains a logical group of fields that can be mapped to the output format. The input record is enabled only on the input side of the map and cannot be referenced by standard rules or links. For example, the input side of the map is SQL and the output side is Positional, and you must generate a record using information from two separate queries. You can create a SQL input record containing fields that obtain data from the appropriate columns of both queries. Then you can link those fields directly to the corresponding positional fields.
Output record		An *output record* represents UPDATE, INSERT, or DELETE SQL statements, and contains database output fields. You specify the name of the table that Gentran Integration Suite modifies and whether to insert, update, or delete. Output record fields represent the columns Gentran Integration Suite is updating or inserting and the key to the affected row or rows. Output records can be created on both the input and output sides of a map and cannot be referenced by standard rules or links.

Component	Icon	Description
Field		A *field* corresponds to a column in a database table and contains the attributes of that column. On the input side, the field receives data from a column of the current row of the open SQL query. On the output side, the field represents a column Gentran Integration Suite updates or inserts, or a key column. Additionally, you can create fields that are not related to a database table or result set of a query.
		When a field has a link performed against it, a red check mark appears over the field icon.
		When a field contains an extended rule or a standard rule, an asterisk appears to the right of the field icon.

About the SQL Manager

The SQL Manager tool enables you to search the database for information to send to Sterling Commerce Customer Support, to help troubleshoot problems within the Gentran Integration Suite database tables. This is a tool that only the system administrator can access, unless the system administrator provides the users permission to the SQL Manager through the user accounts.

> *Caution:*   Use this tool only when Sterling Customer Support directs you to use it. You can make permanent changes to your database, which may result in damage to your Gentran Integration Suite configuration and the loss of data.

Considerations for SQL Mapping

> **Note:**   If you use the SQL syntax on both sides of a map, each side must use a separate set of data sources. One side cannot refer to the data sources belonging to the other side of the map.

For outbound processing, you need a way to determine which records in the database tables were processed. Therefore, you must be able to mark records so that when you process the map, you can select only the unprocessed records in the tables.

To mark records as processed or not processed, you must designate a column in your database to contain a processing flag. This can be a process column that you insert into your database, or an unused column. You must incorporate this column into the SQL statement

record in your map. Also, you must update this column from the map to indicate that a data record was processed.

Caution: Before adding a process column to your database or changing the function of an existing column, copy your data to separate work tables in your database. Copying your data ensures that your testing will not overwrite existing data.

The Map Editor enables you to export data directly to a database.

For more information, see *How the Translator Works with SQL Maps* on page 113.

How the Translator Works with SQL Maps

Following are some caveats about how the translator works with SQL maps:

✦ The translator operates in transactional mode and will roll back any database updates if the process fails.

✦ If the Type parameter in the JDBC pool is set to "local," the translator controls when the rollback occurs. This means that any compliance errors or errors from failed insert/update/delete statements will cause the rollback to occur.

Note: If the JDBC pool Type parameter is set to "remote," the database controls when rollbacks are issued. For a SQL map, it is recommended that the Type parameter be set to "local."

✦ On the Input side of a map you cannot pass parameters to the query being issued. That is, you cannot pass to the query: a value from the map, a variable, or any data that is hard-coded by using constants or a codelist.

✦ You cannot generate a result set on the Output side of a map because you cannot use an Input record on the Output side of a map (and an Input record is the only map component able to generate a result set).

✦ At least one field (column) in an Output Record must be defined as a TableKey.

Overview: How to Create Map Objects

The map objects that you can create depend on which map object is currently selected (has focus in the map). The following table describes the available options (N/A indicates that no map object can be created when the specified object is selected):

If the Currently-Selected Object Is	Then You Can Create
SQL File Format	◆ Group
	◆ Statement record
	◆ Cursor operation record (input only)
	◆ Input record (input only)
	◆ Output record
Group	◆ Group
	◆ Statement record
	◆ Cursor operation record (input only)
	◆ Input record (input only)
	◆ Output record
Statement Record (Query/Command)	N/A
Cursor Operation Record	N/A
Input Record	Field
Output Record	Field
Field	N/A

Managing the SQL File Format

The *SQL File Format* object represents the SQL data sources that Gentran Integration Suite is mapping, including the root element. This object is created automatically by Gentran Integration Suite and enables you to define SQL data sources, data source connection parameters, and extended rules.

Note: The SQL File Format object cannot be referenced by standard rules or links.

Gentran Integration Suite is Java-based, and uses Java Database Connectivity (JDBC) to connect to external databases. Map Editor is Windows-based, and uses Open Database Connectivity (ODBC) to connect to external databases. To use the JDBC adapter or

Lightweight JDBC adapter, you must set up a connection to the external database that contains the tables you want to access.

For more information about properties, press F1 for Help or see Appendix B, *Map Editor Properties*.

For more information, see:

✦ *Modifying SQL File Format Properties* on page 115
✦ *Creating a SQL Data Source* on page 115

Modifying SQL File Format Properties

To modify the properties of an SQL file:

1. Right-click the **SQL File** icon and select **Properties** from the shortcut menu. The Map Editor displays the **SQL File Properties** dialog box (Name tab displayed by default).

2. Do you want to create an SQL data source?

   ◆ If Yes, click the **Data Sources** tab to access data source options.

   For more information about properties, press F1 for Help or see Appendix B, *Map Editor Properties*.

   ◆ If No, continue with step 3.

3. Do you want to specify an extended rule for the SQL file?

   ◆ If Yes, click the **Loop Extended Rules** tab, define the rule, and continue with step 4.

   See Chapter 9, *Using Extended Rules* for more information about extended rules.

   ◆ If No, continue with step 4.

4. Click **OK**. The Map Editor saves your changes and closes the SQL File Properties dialog box.

Creating a SQL Data Source

To create a SQL data source:

1. In the Map Editor, right-click the SQL root element. From the shortcut menu, select **Properties**.

2. In the **SQL File Properties** dialog box, click the **Data Sources** tab to access the data source options.

   For more information about properties, press F1 for Help or see Appendix B, *Map Editor Properties*.

3. Click **SQL Data Sources**.

4. If you want to select a computer data source, select the **Machine Data Source** tab in the **SQL Data Sources** dialog box.

Note: Gentran Integration Suite does not support file data sources.

---

5. In the **Machine Data Source** tab, select the computer data source you want to use and click **OK**.

   Gentran Integration Suite closes the **Select Data Source** dialog box and displays a **Login** dialog box.

6. Verify the login information and click **OK**.

7. In the **SQL File Properties Data Sources** tab, complete the following fields:

   ◆ DSN (required)

   ◆ DATABASE (optional)

   ◆ Password (optional)

   ◆ User-defined name for this data source (optional)

8. Click **Add**.

   Gentran Integration Suite adds the data source to the list and prompts you that it is creating the schema for that data source if you selected the connect option.

9. Do you want to test the connection to a data source?

   ◆ If Yes, select the data source and click **Test Connection**.

      Gentran Integration Suite attempts to rebuild the schema whenever you perform an update or check database consistency, and it tests all queries running against that data source.

      Gentran Integration Suite tests the connection of the selected data source and prompts you with a dialog box containing the results of the test. Click **OK** to close the dialog box.

   ◆ If No, continue with the next step.

10. Click **OK** to close the SQL File Properties dialog box.

Managing Statement Records

The *statement record* represents a SQL query (which returns data and can be used later in the map) or command. Each statement record is associated with a single data source. If the schema for that data source already exists, the Map Editor displays all the tables for your reference.

To use a statement record, you first select a data source. If a schema was generated for the selected data source, the Map Editor displays a list containing all the tables and views.

Then, you can type a SQL statement, test its validity, and generate a result set of columns returned from the query or stored procedure. This result set is used in the SQL field generator for input records. Then you can connect to the data source and test it.

If you specify the name of a stored procedure instead of a SQL statement, you must select the stored procedure option.

> **Note:** The statement record cannot be referenced by standard rules, extended rules, or links.

For more information about properties, press F1 for Help or see Appendix B, *Map Editor Properties*.

For more information, see *Creating a Statement Record* on page 117.

Creating a Statement Record

To create a statement record:

1. Right-click a map object and select either **Create Sub** or **Insert** from the shortcut menu.

2. From the shortcut menu, select **Query/Command**. The Map Editor displays the **SQL Statement Record Properties** dialog box.

3. In the **Name** tab, specify the following:

   ◆ Unique statement record name

   ◆ Description (if applicable)

   ◆ Additional notes (if applicable)

4. Click the **Sql** tab to access the statement options.

5. From the **Data Source** list, select the data source you want to use for this statement.

6. In the **SQL Statement** box, type the SQL statement or stored procedure.

7. Do you want the Map Editor to return a result set when you test the statement?

   ◆ If Yes, select the **Returns a Result Set** option and continue with step 8.

   ◆ If No, continue with step 8.

8. Is the statement a stored procedure?

   ◆ If Yes, select the **SQL statement is Stored Procedure** option and continue with the next step.

   ◆ If No, continue with the next step.

9. Click **Test SQL**.

> **Note:** This function is only valid if you selected **Test Connection** on the **SQL File Format Properties** dialog box (**Data Sources** tab).

See *Managing the SQL File Format* on page 114 for more information.

The Map Editor tests the statement and, if specified, returns a result set.

10. Click **OK**. The Map Editor saves the statement record and closes the SQL Statement Record Properties dialog box.

Managing Cursor Operation Records

The cursor operation record contains instructions for the translator to move through the result set to a new record. Each cursor operation record is associated with a single SQL statement record.

> **Note:** Cursor operation records are only permitted on the input side of a map.

The Map Editor performs cursor operations when the translator encounters them as it processes the map.

> **Note:** The cursor operation record cannot be referenced by standard rules, extended rules, or links.

For more information about properties, press F1 for Help or see Appendix B, *Map Editor Properties*.

For more information, see *Creating a Cursor Operation Record* on page 118.

Creating a Cursor Operation Record

To create a cursor operation record:

1. Right-click a map object and select either **Create Sub** or **Insert** from the shortcut menu.

2. From the shortcut menu, select **Cursor Operation**. The Map Editor displays the **SQL Cursor Operation Record Properties** dialog box.

3. In the Name tab, specify the following:

    ◆ Unique cursor operation record name

    ◆ Description (if applicable)

    ◆ Additional notes (if applicable)

4. Click the **Cursor Operation** tab to access the operation options.

5. From the **Query Record Association** list, select the query that the translator uses to obtain a result set that is then moved to the record.

6. From the **Cursor Operation** list, select the cursor operation.

   For more information about the cursor operations, including their definitions, press F1 for Help or see Appendix B, *Map Editor Properties*.

7. Click **OK**. The Map Editor saves the cursor operation record and closes the SQL Cursor Operation Record Properties dialog box.

Managing Input Records

An input record contains a logical group of fields that are appropriate to be mapped to the output format.

For example, the input side of the map is SQL and the output file format is Positional, and you must generate a record using information from two separate queries. You can create an SQL input record that contains fields which obtain data from the appropriate columns of both queries. Then you can link those fields directly to the corresponding positional fields.

> **Note:** The input record cannot be referenced by standard rules or links.

For more information about properties, press F1 for Help or see Appendix B, *Map Editor Properties*.

For more information, see:

✦ *About Key Field Matching* on page 119
✦ *Auto Get Next Row Operation* on page 120

About Key Field Matching

Input records support the standard Gentran Integration Suite key field constant and code list matching. Additionally, Gentran Integration Suite enables you to match against fields that appear previously in the map. If you want to match against a previously mapped field, you can have up to three key fields.

If you are mapping from a join of master-detail records to an equivalent hierarchy, the translator can determine when the keys in the detail level no longer match the keys in the master level.

> **Note:** Key fields are intended to be used with string fields only. If you use a key field on a numeric field, the result is not guaranteed.

Auto Get Next Row Operation

If input record fields belong to the same query, you can specify that Gentran Integration Suite automatically performs a Move Next cursor operation as the input record loops.

In the **Looping** tab of the **SQL Input Record Properties** dialog box, select **Automatically get next row from Statement record**.

When to Use

The **Automatically get next row from Statement record** option must be used (selected) in the following scenarios:

✦ When retrieving all the data from one table and all the rows must be processed as one document

✦ When processing a repeating Detail record and all corresponding Detail rows must be processed in order

---

Note: You must also use Key Fields in this scenario. When the key fields no longer match, the **Automatically get next row from statement record** function is not performed.

When mapping SQL with the Map Editor, you must use **automatically get next row** when you want to move to the next row of the result set. If you do not instruct the translator to get the next row, it will never do so when running the map. Most often this function is used on a repeating record, when you want to map data from multiple rows of the result set, one row right after another.

Specifying Auto Get Next Row for an Input Record

To specify the automatically get next row operation for an input record:

1. In the Map Editor, right-click a map object. From the shortcut menu, select either **Create Sub** or **Insert**.

2. From the shortcut menu, select **Input Record**. The **SQL Input Record Properties** dialog box opens.

   For more information about properties, press F1 for Help or see Appendix B, *Map Editor Properties*.

3. In the **Name** tab, specify the following:

   ◆ Unique input record name

   ◆ Description

   ◆ Additional notes (if applicable)

4. Click the **Key Field** tab to access the key field options.

5. Select the appropriate options to define the key field.

6. Click the **Looping** tab to access the occurrence options.

7. In the **Maximum usage** box, type the number of times the record can repeat (loop).

8. Do you want to specify that Gentran Integration Suite automatically gets the next row from the statement record order or join?

    ◆ If Yes, select the appropriate option and continue with the next step.

    ◆ If No, continue with the next step.

9. Did you specify that the record repeats (loops)?

    ◆ If Yes, continue with the next step.

    ◆ If No, continue with step 11.

10. Do you want to specify an extended rule for this input record?

    ◆ If Yes, click the **Loop Extended Rules** tab, define the rule, and continue with the next step.

    ◆ If No, continue with the next step.

11. Click **OK** to save the input record and close the SQL Input Record Properties dialog box.

Managing Output Records

An output record represents an UPDATE, INSERT, or DELETE SQL statement, and contains database output fields. You specify the name of the table the translator modifies and whether to INSERT, UPDATE, or DELETE.

The fields contained within an output record represent the columns the translator is updating or inserting and the key to the affected row or rows. Output records may be created on both the input and output sides of a map.

Note: The output record cannot be referenced by standard rules or links.

At least one field (column) in an Output Record must be defined as a TableKey.

For more information about properties, press F1 for Help or see Appendix B, *Map Editor Properties*.

For more information, see *Using the Auto Get Next Cursor Operation* on page 121.

Using the Auto Get Next Cursor Operation

If output record fields belong to the same query, you can specify that the translator automatically performs an auto get next cursor operation as the output record loops.

Note: Depending on the design of the map, if the output record is located on the input side of the map, using the Auto Get Next Cursor Operation may cause unexpected results.

In the **Looping** tab of the **SQL Output Record Properties** dialog box on the input side of the map, specify the **Automatically get next row from Statement record** option.

Specifying the Auto Get Next Cursor Operation

To specify the automatically get next row from cursor operation for an output record:

1. In the Map Editor, right-click a map object. From the shortcut menu, select either **Create Sub** or **Insert**.

2. From the shortcut menu, select **Output Record**. The **SQL Output Record Properties** dialog box opens.

   For more information about properties, press F1 for Help or see Appendix B, *Map Editor Properties*.

3. In the **Name** tab, specify the following:

   ◆ Unique output record name

   ◆ Description

   ◆ Additional notes (if applicable)

4. Click the **SQL Operations** tab to access operation options.

Note:	If you choose to perform an Insert or Update operation, you can also select the **On failure, automatically switch selected operation and retry Inserts as Updates or Updates as Inserts** option to specify that if the operation is not completed, the translator will automatically switch the operation type (that is, update to insert or insert to update) and retry the operation.

5. From the **Data Sources** list, select the appropriate data source.

6. From the **Table** list, select the table you want to modify, then select the appropriate table operation.

7. Click the **Key Field** tab to access the key field options.

8. If the output record is on the input side of the map, select the appropriate options to define the key field.

9. Click the **Looping** tab to access the occurrence options.

10. In the **Maximum usage** box, type the number of times the record can repeat (loop).

11. Did you specify that the record repeats (loops)?

    ◆ If Yes, continue with the next step.

    ◆ If No, continue with step 13.

12. Do you want to specify an extended rule for this output record?

    ◆ If Yes, click the **Loop Extended Rules** tab, define the rule, and continue with the next step.

    ◆ If No, continue with the next step.

13. Click **OK** to save the output record and close the SQL Output Record Properties dialog box.

Generating Database Fields

The Generate Fields function provides a wizard that enables you to quickly and easily generate fields for an input or output record. The function differs slightly depending on whether the record is input or output.

Note: Before using this function, test your SQL queries.

For more information, see:

✦ *Generating Input Record Database Fields* on page 123
✦ *Generating Output Record Database Fields* on page 123

Generating Input Record Database Fields

To generate input record database fields:

1. In the Map Editor, right-click an input record. From the shortcut menu, select **Generate Fields**.

    The SQL Field Generator Wizard opens.

2. From the **Queries** list, select the defined queries and click **Next**.

    Gentran Integration Suite displays only the queries that return a result set.

3. From the **Columns** list, select the result columns and click **Finish**.

    Click **Choose All** to select all the columns. Click **Clear** to clear all columns.

    Gentran Integration Suite adds the specified result columns to the record and automatically generates the validation settings.

Generating Output Record Database Fields

To generate output record database fields:

1. In the Map Editor, right-click an output record. From the shortcut menu, select **Generate Fields**.

    The SQL Field Generator Wizard opens.

2. Is the output record already associated with a data source?

    ◆ If Yes, continue with the next step.

    ◆ If No, from the **Data Source** list, select the data source.

3. Is the output record already associated with a table?

    ◆ If Yes, continue with the next step.

    ◆ If No, from the **Table** list, select the database table.

4. Select the table operation and click **Next**.

    Valid values are:

    ◆   Insert

    ◆   Update

    ◆   Delete

    The SQL Field Generator Wizard: Select Column(s) dialog box opens.

5. From the **Columns** list, select the result columns and click **Finish**.

    Click **Choose All** to select all the columns. Click **Clear** to clear all columns.

    Gentran Integration Suite adds the specified result columns to the record and automatically generates all the validation settings.

Creating SQL Fields

A field functions differently, depending on whether it is located in an input or output record.

A field does not have to be associated with a query or output column. This independence enables you to define, for example, temporary fields for calculations. An input record is a temporary record if no field in the record is associated with a query.

Gentran Integration Suite automatically selects the field type if you previously generated schema or result set information. If you did not generate a schema or result set, you must specify the field type manually.

Note: When a field has an operation performed against it (link, standard rule, or as an extended rule storage field), Gentran Integration Suite displays a red check mark over the attribute icon.

For more information, see:

✦   *Field In an Input Record* on page 125
✦   *Field In an Output Record (Input or Output Side of Map)* on page 125
✦   *Field In an Output Record (Input Side of Map)* on page 125

Field In an Input Record

If the field is located in an input record, the field can receive data from a column in the current row of a result set of a query. You associate the field with a column returned by a query.

Note:	Column names are not available if you do not test the selected query. Instead, you must select a column number that is a one-based number of the position of that column in the result set.

Field In an Output Record (Input or Output Side of Map)

If the field is located in an output record on the input or output side of the map, the field represents a column that the translator updates or inserts, or a key column. You specify the name of the column and indicate whether it is a key. Gentran Integration Suite uses key columns automatically to create a WHERE clause for UPDATE and DELETE statements.

Note:	If the field is contained in an output record, the validation information is set from the column to which it is mapped.

To verify whether a link is valid when using SQL in an output record:

1. Verify all fields that are linked are of same data type.

2. Verify all columns in the table that defined as **not null** are populated with data in the map.

3. Verify that, for any linked fields, the input field length is not greater than the output field length.

4. Verify that the data to be inserted does not violate any unique key or any other constraints on the tables.

Field In an Output Record (Input Side of Map)

If the field is located in an output record on the input side of a map, both the input and output records are functional.

Note:	If the field is contained in an output record, the validation information is set from the column information.

For more information about properties, press F1 for Help or see Appendix B, *Map Editor Properties*.

To create a output record field on the input side of the map:

1. In the Map Editor, right-click a map object. From the shortcut menu, select either **Create Sub** or **Insert**.

---

2. From the shortcut menu, select **Field**. The **SQL Field Properties** dialog box opens.

 For more information about properties, press F1 for Help or see Appendix B, *Map Editor Properties*.

3. In the **Name** tab, specify the following:

 ◆ Unique field name

 ◆ Description

 ◆ Additional notes (if applicable)

4. Is this a field on the input side of the map?

 ◆ If Yes, click the **Query** tab and continue with the next step.

 ◆ If No (this is a field on the output side), select the **Column** tab and continue with step 8.

5. From the **Associate Statement Record** list, select the record with which this field is associated.

6. From the **Name** list, select the table name.

 If the **Name** list is not available, you did not test the SQL query. Instead, select a number from the **Number** box.

7. Do you want to enable nulls?

 ◆ If Yes, select the **Nulls Allowed** option and continue with step 11.

 ◆ If No, continue with step 11.

8. In the **Column** tab, select the column name from the **Name** list.

 If the **Name** list is not available, you did not select to connect to a data source to create a table in the **SQL File Properties** dialog box. Instead, select a number from the **Number** box.

9. Do you want to enable nulls?

 ◆ If Yes, select the **Nulls Allowed** option and continue with the next step.

 ◆ If No, continue with the next step.

 This option is applicable only for fields in an output record.

10. Will this field be used in a WHERE clause?

 ◆ If Yes, select the **Table Key** option and continue with the next step.

 ◆ If No, continue with the next step.

11. In the **Validation** tab, specify the following:

 ◆ Whether the field is required

 ◆ Minimum length

 ◆ Maximum length

 ◆ Type of data

♦ How the data is formatted

12. Do you want to specify a standard rule for this field?

- If Yes, click the **Standard Rule** tab, define the rule, and continue with the next step.

- If No, continue with the next step.

13. Do you want to specify an extended rule for this field?

- If Yes, click the **Extended Rule** tab, define the rule, and continue with the next step.

- If No, continue with the next step.

14. Click **OK** to save the field and close the SQL Field Properties dialog box.

Checking Database Consistency

Use the Check Database Consistency function to compare the definitions of each query, output record, and field with the associated data source, and to return a list of any inconsistencies. This function alerts you to inconsistencies in your database, but does not fix your map components.

Gentran Integration Suite also checks database consistency:

✦ When you update the data source (**SQL File Properties** dialog box, on the **Data Source** tab)

✦ As the first action of compiling the map

An inconsistency occurs if, for a field, the translator tries to extract from a query column data that is not returned by the data source.

To check database consistency:

1. In the Map Editor, right-click the SQL root element. From the shortcut menu, select **Check Database Consistency**.

   Gentran Integration Suite verifies the consistency of the selected side of the map and displays the **File Format Consistency Check** dialog box, which displays any inconsistencies.

2. Click **OK** to close the File Format Consistency Check dialog box.

Chapter 7

Mapping XML Documents

The Map Editor enables you to map to and from Extensible Markup Language (XML) documents.

This section covers the following topics:

✦ About the XML Data Format
✦ Creating an XML Layout from a DTD or Schema
✦ Creating an XML Map from a Preloaded Standard
✦ Creating an XML Map from a Financial Services Standard
✦ Overview: Creating Map Objects
✦ Managing the XML File
✦ Managing XML Elements
✦ Managing Content Particles
✦ Managing Pcdata
✦ Managing Attributes
✦ About XML Namespaces
✦ Using Namespaces in the XML File

About the XML Data Format

The Gentran Integration Suite XML implementation conforms to the rules of the XML language 1.0 specification, as published by the World Wide Web Consortium (W3C). However, to be more flexible, Gentran Integration Suite diverges slightly from the W3C base 1.0 XML specification. The Gentran Integration Suite implementation enables you to do the following with XML:

✦ Specify the number of times that a group can repeat.
✦ Specify the number of times an element in a mixed group can repeat.

✦ Repeat an element (with a structure different from the structure of the original element) in a different part of the document.

For more information, see:

✦ *XML Components* on page 130

✦ *XML Considerations* on page 131

✦ *Support for XML Schemas in Gentran Integration Suite* on page 132

✦ *Abstract Elements and Gentran Integration Suite* on page 133

✦ *Importing Large XML Files* on page 133

Gentran Integration Suite allows you to select a COBOL copybook file layout (in ASCII text) as input to the Map Editor for the positional side of a map. The conversion of COBOL copybook is accomplished using the third-party open source tool CB2XML (Copy-Book-to-XML), which is a COBOL copybook to XML converter. CB2XML accepts a COBOL copybook file layout as input and returns an XML file as output, and the Map Editor then converts the intermediate XML file to the positional format. See *COBOL Copybook Conversion for Use with Map Editor* on page 409 for more information.

XML Components

The following table lists the components that make up the XML layout in the Map Editor, the icons that represent the components, and descriptions of the components. For information about adding a map component to a layout, see Chapter 1, *Map Editor Basics* and *Overview: Creating Map Objects* on page 141. For information about the properties of these map components, see Appendix B, *Map Editor Properties*.

Component	Icon	Description
XML root element	✗	The *XML root element* represents the XML document that Gentran Integration Suite is mapping. The XML root element is a looping structure that contains elements and content particles that repeat in sequence until either the group data ends or the maximum number of times that the loop is permitted to repeat is exhausted. The root element cannot be referenced by standard rules or links.
Element	◆	An *element* contains related elements and content particles. In addition, an element can contain one pcdata and one attribute container. These objects repeat in sequence until either the element data ends or the maximum number of times that the loop is permitted to repeat is exhausted. A repeating element that contains another repeating element corresponds to a nested looping structure. The XML Element object cannot be referenced by standard rules or links.

Component	Icon	Description
Abstract element		An *abstract element* is an inconcrete element from an XML schema (for example, the term "appliance" may be considered an abstract element while "dishwasher" is a concrete one). An abstract element must be substituted with a non-abstract element for a map to be successfully compiled.
Content particle		A *content particle* contains related subordinate objects that define either a choice, a sequence, or an all. A content particle can contain only one pcdata. If specified, these objects can repeat in sequence until either the content particle data ends or the maximum number of times that the loop is permitted to repeat is exhausted. If you create a content particle that is subordinate to another content particle, the content particle corresponds to a nested looping structure (a loop within a loop). A content particle cannot be referenced by standard rules or links.
Pcdata		A *pcdata* contains character data in an XML document. Only one pcdata can be defined per element or content particle. Gentran Integration Suite automatically names the pcdata with the name of the parent element or content particle.
		When a pcdata has a link performed against it, a red check mark appears over the pcdata icon.
		When a pcdata contains an extended rule or a standard rule, an asterisk appears to the right of the pcdata icon.
Attribute container		An *attribute container* does not correspond to an XML function. Gentran Integration Suite uses attribute containers to contain the attributes of an XML element. The attribute container has no properties. When you create the first attribute of an XML element, the Map Editor automatically creates an attribute container object. An element can have only one attribute container, but the attribute container object can enclose many attribute objects.
Attribute		An *attribute* specifies information associated with an element that further defines the element. The attribute is located within an attribute container. Attributes do not have to occur in sequence in the input data.
		When an attribute has a link performed against it, a red check mark appears over the attribute icon.
		When an attribute contains an extended rule or a standard rule, an asterisk appears to the right of the attribute icon.

XML Considerations

Before you create an XML layout, consider:

✦ You have two options for creating an XML layout: manually create an XML layout or create an XML layout from a document type definition (DTD) or XML schema.

✦ If you are using an XML schema, you must install the following software. Both are available from the Microsoft Web site.

◆ Microsoft XML Core Services 4.0 RTM (XML parser)

◆ Windows Installer 2.0 (to install the preceding XML parser)

An XML schema defines rules for the structure and content of an XML document. The schema you use must meet www.w3.org schema standards. Only valid schemas will generate an XML map structure.

> **Note:** Verify the schema is valid by using a Schema Validator.

Support for XML Schemas in Gentran Integration Suite

The Translation service uses the Xerces 1.4.3 XML parser, which supports the W3C XML Schema recommendation version 1.0 (available at www.w3.org/TR/xmlschema-0).

> **Note:** Because of how the Xerces XML parser handles elements with complex content, it is best to modify the schema to use unbounded instead of maxOccurs with large values and to avoid using large values for minOccurs. Otherwise, you may see a StackOverflowError from the Xerces parser during translation.

The Map Editor uses the Microsoft XML Core Services 4.0 RTM (XML parser) to read and validate XML schemas. The Map Editor does not display the following features of XML schemas:

✦ Substitution groups

✦ Data type facets other than minimum and maximum length

✦ Abstract element types (for more information about how abstract element types are supported in the Map Editor, see *Abstract Elements and Gentran Integration Suite* on page 133)

✦ Data type derivations

✦ The Map Editor uses XML schemas to create the initial file format. You cannot create a new element after importing a schema by referring to a type defined in the schema.

✦ Annotations

✦ Prohibited attribute constraints

✦ The nillable, fixed-value, and default value attributes of an element

The Map Editor supports the following XML schema features, and:

✦ Treats list data types as strings

✦ Creates a code list to contain the enumeration values of a simple type derived by enumeration

✦ If a simple data type is derived by the union of differing integral types, treats the data type as a string (for example, a date/time type joined with an integer type would be treated as a string)

✦ The translator currently wraps a CDATA section around any text that contains one of the five *special characters* (< (less than), > (greater than), & (ampersand), ' (single quote or apostrophe), and " (double quote)

Abstract Elements and Gentran Integration Suite

An abstract element is an inconcrete element from an XML schema (for example, the term "appliance" may be considered an abstract element while "dishwasher" is a concrete one). An abstract element must be substituted with a non-abstract element for a map to be successfully compiled. For more information about changing an abstract element to an XML element, see *Creating an Element* on page 144.

Note: Abstract elements are reported during map compilation as Warnings.

Importing Large XML Files

You import XML files, including maps, into Gentran Integration Suite using the Resource Manager.

The import properties parameter is set to 10MB, but you can increase the parameter if you need to import a larger file.

Caution: If you edit the a properties file and **setupfiles.sh / .cmd** is subsequently executed for any reason, the changes you made *will be lost*. This is true for every properties file that has an ".in" version.

To change the settings for XML Import file size, modify the size of the **maxImportUpload** parameter in the following files:

✦ tp_import_export.properties

✦ tp_import_export.properties.in

To change the settings for XML Import file size:

1. Using a text editor, open the **tp_import_export.properties** file, which can be found in *<install dir>*/properties in your Gentran Integration Suite installation.

2. To change the settings for XML Import file size, modify the size of the **maxImportUpload** parameter.

3. Using a text editor, open the **tp_import_export.properties.in** file, which can be found in *<install dir>*/properties in your Gentran Integration Suite installation.

4. To change the settings for XML Import file size, modify the size of the **maxImportUpload** parameter.

5. Once you complete the edits, execute *<install_dir>***/bin/setupfiles.sh** and then stop and restart Gentran Integration Suite to verify the changes take effect.

Creating an XML Layout from a DTD or Schema

When you create a new map, you can either manually create an XML layout or you can use a wizard that creates a layout for you based on a DTD or schema.

The wizard performs the following tasks:

✦   Raises a warning if it encounters attributes that use entities or notations

✦   Changes attributes of type ENTITY or ENTITIES to type CDATA

✦   Changes attributes of type NOTATION to type ENUMERATED

✦   Disregards comments and processing instructions

✦   Discards external entities and notations

✦   Does not support XML namespaces for DTDs or conditional sections

✦   Supports internal and external parameter entities

✦   Supports external parameter entities that reference a URL only if Microsoft® Internet Explorer 3.0 or later is installed on the computer

For more information, see *Example* on page 136.

To create an XML layout from a DTD or schema:

1.   From the Map Editor **File** menu, select **New**.

2.   In the **New Map Wizard**, complete the questions on the first screen.

> **Note:**   To use XML schemas, the map type must be Sterling Integrator.

3.   If you are translating from XML, from the **Create a new data format using this syntax** list, select **XML** and click **Customize**. If you are translating from another format, select that format and continue to the next screen.

4.   In the **New XML Wizard**, select the **Customization File Type**—the type of file you want to use to create your XML layout, either DTD or XML schema.

5.   Type the name of the DTD or XML schema, type the URL, or browse to the location of the DTD or XML schema, and click **Next**.

6.   Select the root element.

   The DTD or schema does not explicitly define the root element, so you can select from all the elements defined in the DTD or all the top-level elements defined in the .XSD schema (elements subordinate to the schema element). By default, the wizard selects the first element it encounters.

7.   Specify the maximum length for data elements.

   For a DTD, you can specify the maximum length of data elements because length is not defined in the DTD and is optional for elements declared in the XML schema. For an XML schema, the maximum length can be specified for some elements. You can specify a default field length for elements that do not have a specified field length.

8. If you want the map to contain code lists equivalent to the enumerations defined in the DTD or schema, select the **Build code lists for enumerated attributes** check box.

> **Note:** The default for the **Build code lists for enumerated attributes** check box is cleared because using code lists may be redundant and thus your map is smaller and more efficient without them.

9. Click **Next**.

10. If you chose to import an XML schema, select the subelements you want to be included in the map, and click **Next**.

> **Note:** The list displays the entire tree of elements and subelements for the root element you selected, hierarchically displayed using indentation. Attributes are not included in this list.
>
> The default is that all elements and subelements are selected. When you clear the box next to an element, all its subelements are automatically cleared. Similarly, when you select the box next to a child element, its parent elements are automatically selected.
>
> Selecting only the elements and subelements you need for your map reduces the size of the map and makes it more efficient.

11. If you are translating to XML, on the Output screen (from the **Create a new data format using this syntax** list), select **XML** and click **Customize**.

12. In the **New XML Wizard**, select the **Customization File Type**—the type of file you want to use to create your XML layout, either DTD or XML schema.

13. Type the name of the DTD or XML schema, type the URL, or browse to the location of the DTD or XML schema, and click **Next**.

> **Note:** When using the file:// protocol to reference a DTD, if you are unsure of your FTP connection (host name and FTP or Internet access) and the DTD is on the Gentran Integration Suite server, provide an absolute path in the reference by omitting the host name and leaving the following slash. For example, file://home/sterling/dtd/test.dtd. If you are sure of your FTP connection, then you can specify a host name.

If you are using an XML schema and the XML parser detects any errors, the messages are displayed in an error window.

14. Select the root element.

The DTD or schema does not explicitly define the root element, so you can select from all the elements defined in the DTD or all the top-level elements defined in the .XSD schema (elements subordinate to the schema element). By default, the wizard selects the first element it encounters.

15. Specify the maximum length for data elements.

    For a DTD, you can specify the maximum length of data elements because length is not defined in the DTD and is optional for elements declared in the XML schema. For an XML schema, the maximum length can be specified for some elements. You can specify a default field length for elements that do not have a specified field length.

16. If you want the map to contain code lists equivalent to the enumerations defined in the DTD or schema, select the **Build code lists for enumerated attributes** check box.

    > **Note:** The default for the **Build code lists for enumerated attributes** check box is cleared because using code lists may be redundant and thus your map is smaller and more efficient without them.

17. Click **Next**.

18. If you chose to import an XML schema, select the subelements you want to be included in the map, and click **Next**.

    > **Note:** The list displays the entire tree of elements and subelements for the root element you selected, hierarchically displayed using indentation. Attributes are not included in this list.
    >
    > The default is that all elements and subelements are selected. When you clear the box next to an element, all its subelements are automatically cleared. Similarly, when you select the box next to a child element, its parent elements are automatically selected.
    >
    > Selecting only the elements and subelements you need for your map reduces the size of the map and makes it more efficient.

19. Click **Finish**.

    The **XML Load Warnings** dialog box opens if there are any errors. If the Map Editor made changes to the DTD to make it compliant with Gentran Integration Suite, it indicates the changes. Click **OK**.

20. Continue with the **New Map Wizard** as directed. When you click **Finish**, the Map Editor displays the new map in the Map Editor window.

Example

The following figure shows an example of an XML document:

```
<?xml version="1.0" encoding="UTF-8"?>
<services>
  <service name="BPMetaDataInfoService" activestatus="1" systemservice="1"/>
</services>
```

Gentran Integration Suite Map Editor Guide

Creating an XML Map from a Preloaded Standard

When you create a map, for both the input and output sides, you choose whether you want to use a preloaded standard, an existing file format, or want to create a new file format for that side of the map (including selecting from standards that you have previously downloaded to the standards database. The preloaded standards are downloaded when you download the Map Editor to your machine and any optional standards such as those included with the Financial Services XML Standards bundle, and include the following:

Note: For a complete list of the preloaded standards consult the Map Editor New Map Wizard.

- ✦ FIXML
- ✦ FpML
- ✦ IFX
- ✦ OFX
- ✦ TARGET2
- ✦ TWIST
- ✦ ISO20022
- ✦ SWIFTNet Funds
- ✦ SWIFTNet Trade Services
- ✦ SWIFTNet Exceptions and Investigations
- ✦ SWIFTNet Cash Reporting

Note: If the map you are creating contains greater than 20,000 objects, you will receive a message noting that this map contains a very large number of objects. For best performance, it is recommended that you consider whether any unnecessary objects in the map can be removed, do not expand the entire object tree—expand only the section of the tree you are currently mapping, consider using the "Show links to or from the currently selected element" option instead of the "Show links to or from all visible elements" option, and save the map using the.MAP file format (using the Save As function).

To create an XML layout from a preloaded standard:

1. From the Map Editor **File** menu, select **New**.

2. In the **New Map Wizard**, complete the questions on the first screen.

   Note: To use XML schemas, the map type must be Sterling Integrator.

3. If you are translating from XML, from the **Create a new format using this syntax** list, select **XML** and click **Customize**. If you are translating from another format, select that format and continue to the next screen.

4. Type the name of the XML schema, type the URL, or browse to **Map Editor\Samples\Schemas** and the folder containing the schema type you would like to load), select the schema, and click **Next**.

    If you are using an XML schema and the XML parser detects any errors, the messages are displayed in an error window.

5. Select the root element.

    The schema does not explicitly define the root element, so you can select from all the top-level elements defined in the .XSD schema (elements subordinate to the schema element). By default, the wizard selects the first element it encounters.

6. Specify the maximum length for data elements.

    For an XML schema, the maximum length can be specified for some elements. You can specify a default field length for elements that do not have a specified field length.

7. If you want the map to contain code lists equivalent to the enumerations defined in the schema, select the **Build code lists for enumerations** check box.

    > **Note:** The default for the **Build code lists for enumerations** check box is cleared because using code lists may be redundant and thus your map is smaller and more efficient without them.

8. Click **Next**.

9. If you chose to import an XML schema, do not modify any of the subelements selected in the display. Click **Next**.

10. If you are translating to XML, on the Output screen (from the **Create a new format using this syntax** list), select **XML** and click **Customize**.

11. In the **New XML Wizard**, select the **Customization File Type**—the type of file you want to use to create your XML layout, either XML schema.

12. Type the name of the XML schema, type the URL, or browse to **Map Editor\Samples\Schemas** and the folder containing the schema type you would like to load), select the schema, and click **Next**.

    If you are using an XML schema and the XML parser detects any errors, the messages are displayed in an error window.

13. Select the root element.

    The schema does not explicitly define the root element, so you can select from all the top-level elements defined in the .XSD schema (elements subordinate to the schema element). By default, the wizard selects the first element it encounters.

14. Specify the maximum length for data elements.

    For an XML schema, the maximum length can be specified for some elements. You can specify a default field length for elements that do not have a specified field length.

15. If you want the map to contain code lists equivalent to the enumerations defined in the schema, select the **Build code lists for enumerations** check box.

> **Note:** The default for the **Build code lists for enumerations** check box is cleared because using code lists may be redundant and thus your map is smaller and more efficient without them.

16. Click **Next**.

17. If you chose to import an XML schema, do not modify any of the subelements selected in the display. Click **Next**.

18. Click **Finish**.

    The **XML Load Warnings** dialog box opens if there are any errors. If the Map Editor made changes to the DTD to make it compliant with Gentran Integration Suite, it indicates the changes. Click **OK**.

19. Continue with the **New Map Wizard** as directed. When you click Finish, the Map Editor displays the new map in the Map Editor window.

Creating an XML Map from a Financial Services Standard

When you create a map, for both the input and output sides, you choose whether you want to use a preloaded standard, an existing file format, or wants to create a new file format for that side of the map (including selecting from standards that you have previously downloaded to the standards database such as the Financial Services XML Standards).

> **Note:** To create a map using any of these standards you must have purchased the appropriate license from Sterling Commerce and downloaded the Financial Services XML standards bundle to the same machine where the Map Editor is installed. For a complete list of the Financial Services XML standards consult Customer Support.

To create a map using a Financial Services XML standard:

1. From the Map Editor **File** menu, select **New**.

2. In the **New Map Wizard**, complete the questions on the first screen.

> **Note:** To use XML schemas, the map type must be Sterling Integrator.

3. If you are translating from a Financial Services XML standard, from the **Create a new data format using this standard** list, select one of the Financial Services XML standards and click **Messages**. If you are translating from another format, select that format and continue to the next screen.

---

4. Complete the Map Wizard by selecting the version of the standard, the desired message, the maximum length of the data elements, and whether to build codelists for enumerated attributes, and click **Next**.

   The default for the **Build code lists for enumerated attibutes** check box is cleared because using code lists may be redundant and thus your map is smaller and more efficient without them.

5. Click **Next**.

6. If you chose to import an XML schema, do not modify any of the subelements selected in the display. *This list is included for your information only.* Click **Next**.

> **Note:** For the OFX and IFX standards only, all elements except the root are not initially selected. The list includes all messages for the messages category selected. You must select the messages you want to use in the map.

---

7. Click **Finish**.

8. If you are translating to a Financial Services XML standard, on the Output screen (from the **Create a new data format using this standard** list), select one of the Financial Services XML standards and click **Messages**. If you are translating from another format, select that format and continue to the next screen.

9. Complete the Map Wizard by selecting the version of the standard, the desired message, the maximum length of the data elements, and whether to build codelists for enumerated attributes, and click **Next**.

   The default for the **Build code lists for enumerated attributes** check box is cleared because using code lists may be redundant and thus your map is smaller and more efficient without them.

10. Click **Next**.

11. If you chose to import an XML schema, do not modify any of the subelements selected in the display. *This list is included for your information only.* Click **Next**.

12. Click **Finish**.

    The **XML Load Warnings** dialog box opens if there are any errors. Click **OK**.

13. Continue with the **New Map Wizard** as directed. When you click **Finish**, the Map Editor displays the new map in the Map Editor window.

Overview: Creating Map Objects

The map objects that you can create depend on which map object is currently selected (has focus in the map). The following table describes the available options (N/A indicates that no map object can be created when the specified object is selected):

If the Currently-Selected Object Is	Then You Can Create
XML File (root element)	◆ Element ◆ Content particle ◆ Pcdata ◆ Attribute
Element	◆ Element ◆ Content particle ◆ Pcdata ◆ Attribute
Content Particle	◆ Element ◆ Content particle ◆ Pcdata
Pcdata	N/A
Attribute Container	Attribute
Attribute	N/A

Managing the XML File

The *XML File object* represents the XML document that Gentran Integration Suite is mapping, including the root element. This object is created automatically by Gentran Integration Suite. This table describes the two dialog boxes unique to the XML File object:

Dialog Box	Function
XML File Properties dialog box	Enables you to define entities, output format specifications, and file-level extended rules.
Entity Properties dialog box	Enables you to define entities. This dialog box is accessible through the **XML File Properties** dialog box.

Note: The XML File object cannot be referenced by standard rules or links.

For information about properties, press F1 for Help or see Appendix B, *Map Editor Properties*.

For more information, see:

✦ *Modifying XML File Properties* on page 142
✦ *Creating an Entity* on page 143
✦ *Regarding Decimal Points* on page 143

Modifying XML File Properties

To modify the properties of an XML file:

1. Right-click the **XML File** icon and select **Properties** from the shortcut menu. The Map Editor displays the **XML File Properties** dialog box (**Name** tab displayed by default).

2. Do you want to create an entity?

    ◆ If Yes, click the **Entities** tab to access entity options.

    ◆ If No, continue with step 3.

3. Do you want to modify the output options for the XML file?

    ◆ If Yes, click the **Output** tab and continue with step 4.

    ◆ If No, continue with step 5.

4. In the **Output** tab, specify the following:

    ◆ Whether the translator generates a prolog or document type declaration

    ◆ Public ID (if applicable)

- ◆ System ID (if applicable)

- ◆ How the XML elements are output to the file

5. Do you want to specify an extended rule for the XML file?

- ◆ If Yes, select the **Loop Extended Rules** tab, define the rule, and continue with step 6. See Chapter 9, *Using Extended Rules* for more information about extended rules.

- ◆ If No, continue with step 6.

6. Click **OK**. The Map Editor saves your changes and closes the XML File Properties dialog box.

Creating an Entity

An *entity* is an object that represents a string of characters.

To create an entity:

1. In the Map Editor, right-click the XML root element. From the shortcut menu, select **Properties**.

2. In the **XML File Properties** dialog box, click the **Entities** tab to access the entity options.

3. Click **New**.

4. In the **Entity Properties** dialog box, specify the properties as necessary.

   For information about properties, press F1 for Help or see Appendix B, *Map Editor Properties*.

Note:	In the **Tag** tab, the tag must match the entity tag in the XML document. XML tags *cannot* contain spaces.

5. Click **OK** to save the entity and close the Entity Properties dialog box.

6. Click **OK** again to close the XML File Properties dialog box.

Regarding Decimal Points

If you need to specify that each decimal point in your data is defined and generated as a something other than the default period (.), you can do so on the Decimal Point tab.

To change the default decimal point setting:

1. In the Map Editor, open each XML file.

2. Open the **XML File Properties** dialog box.

3. In the **Decimal Point** tab, select the **Define Decimal Point** check box.

4. In the **Decimal Point Character** box, type the value you want Gentran Integration Suite to use as a decimal point—for example, a comma (,). This resets the decimal point default to use what you specified instead of a period.

5. Click **OK**.

6. Save the map.

7. Compile the map.

    The decimal point setting is now correctly set to use your specification.

Managing XML Elements

An *XML element* contains related elements, content particles, or both. In addition, an element can contain one pcdata, one attribute or both. These objects repeat in sequence until either the element data ends or the maximum number of times that the loop is permitted to repeat is exhausted.

> **Note:**   The XML Element object cannot be referenced by standard rules or links.

For information about properties, press F1 for Help or see Appendix B, *Map Editor Properties*.

For more information, see:

✦ *Creating an Element* on page 144

✦ *Inserting an Element from a Schema or DTD* on page 145

Creating an Element

To create an element:

1. Right-click a map object and select either **Create Sub** or **Insert** from the shortcut menu.

2. From the shortcut menu, select **Element**. The Map Editor displays the **XML Element Properties** dialog box.

3. In the **Name** tab, specify the following:

    ◆ Unique element name

    ◆ Description (if applicable)

    ◆ Additional notes

4. If necessary, click the **Tag** tab and change the value in the Tag box.

> **Note:** This value must match the element in the XML document. XML tags *cannot* contain spaces.

---

5. Select the **Repeating** tab to access the occurrence options.

6. Select either the **Conditional** or **Mandatory** option to specify whether the element is required in the map.

7. Select the appropriate repeating option for the element.

8. If you must specify the number of times the element can repeat (loop), type that number in the **Maximum usage** box.

9. Did you specify that the element repeats (loops)?

    ◆ If Yes, continue with step 11.

    ◆ If No, continue with step 12.

10. Do you want to specify an extended rule for this element?

    ◆ If Yes, click the **Loop Extended Rules** tab, define the rule, and continue with step 12.

       See Chapter 9, *Using Extended Rules* for more information about extended rules.

    ◆ If No, continue with step 12.

11. Click **OK**. The Map Editor saves the element and closes the XML Element Properties dialog box.

Inserting an Element from a Schema or DTD

The Insert Element from Schema or DTD function enables you to add an element from a schema or DTD.

To insert an element from a schema or DTD:

1. Right-click the XML map component after which you want to insert the element and select **Insert > Element from schema/dtd**, or if you want the element to be the first element in the XML file format, right-click the XML File icon and select **Create Sub > Element from schema/dtd**.

2. From the **Customization File Type** list, select whether the element will be copied from a DTD or schema.

3. In the **File Location** box, type the file path and name of the DTD or schema or click **Browse** and select the file (then click **Open**).

4. Click **Next**.

5. Follow the instructions on the next wizard page and click **Next**.

6. If you chose to insert the element from a schema, you are prompted with another wizard page. Follow the instructions on that wizard page and click **Next**.

7. Click **Finish** to insert the element into your XML file format. The Map Editor displays the XML Element Properties dialog box so you can configure the element properties.

   If the inserted element is an abstract one, the displayed XML Element Properties dialog box enables you to provide a concrete element to replace the abstract one, by clearing the **Defined as Abstract** check box on the **Tag** tab (you *must* replace abstract elements with concrete elements by clearing this check box to avoid map compilation errors or, optionally, you can avoid the compilation errors by deactivating the abstract elements). For more information about abstract elements, see *Abstract Elements and Gentran Integration Suite* on page 133. For more information about deactivating map components, see *Activating Map Components* on page 71.

8. In the **Name** tab, specify the description (if applicable).

Note:	The **Additional notes** box is populated automatically with the XML element annotation when you import an XML schema.

9. If necessary, click the **Tag** tab and change the value in the Tag box.

Note:	This value must match the element in the XML document. XML tags *cannot* contain spaces.

10. If the element is an abstract type, select the **Tag** tab and clear the **Defined as abstract** check box. By clearing this check box, you change the abstract element to a concrete one. This check box must be cleared to enable the map to compile successfully (without compilation errors).

Note:	Optionally, you can avoid compilation errors from abstract elements by deactivating the abstract elements. For more information about deactivating map components, see *Activating Map Components* on page 71.

11. Select the **Repeating** tab to access the occurrence options.

12. Select either the **Conditional** or **Mandatory** option to specify whether the element is required in the map.

13. Select the appropriate repeating option for the element.

14. If you must specify the number of times the element can repeat (loop), type that number in the **Maximum usage** box.

15. Did you specify that the element repeats (loops)?

    ◆ If Yes, continue with step 11.

    ◆ If No, continue with step 12.

16. Do you want to specify an extended rule for this element?

    ◆ If Yes, click the **Loop Extended Rules** tab, define the rule, and continue with step 12.

    For more information about extended rules, see Chapter 9, *Using Extended Rules*.

◆   If No, continue with step 12.

17. Click **OK**. The Map Editor saves the element and closes the XML Element Properties dialog box.

Managing Content Particles

In Gentran Integration Suite, a *content particle* contains child objects that define either a choice or a sequence. A content particle can contain related elements, content particles, or both. In addition, a content particle can contain one pcdata. If specified, these objects repeat in sequence until either the content particle data ends, or the maximum number of times that the loop is permitted to repeat is exhausted.

Note: The Content Particle object cannot be referenced by standard rules or links.

For information about properties, press F1 for Help or see Appendix B, *Map Editor Properties*.

For more information, see *Creating a Content Particle* on page 147.

Creating a Content Particle

To create a content particle:

1. Right-click a map object and select either **Create Sub** or **Insert** from the shortcut menu.

2. From the shortcut menu, select **Content Particle**. The Map Editor displays the **Content Particle Properties** dialog box.

3. In the **Name** tab, specify the following:

    ◆   Unique content particle name

    ◆   Description (if applicable)

    ◆   Additional notes

Note: The Additional notes box is populated automatically with the XML content particle annotation when you import an XML schema.

---

4. Click the **Type** tab to access the content particle type options.

5. Select the appropriate option to define what the child objects of the content particle represent.

6. Click the **Repeating** tab to access the occurrence options.

7. Select **Conditional** or **Mandatory** to specify whether the content particle is required in the map.

8. Select the appropriate repeating option for the content particle.

9. In the **Maximum usage** box, type the number of times the content particle can repeat (loop).

10. Did you specify that the content particle repeats (loops)?

   ◆ If Yes, continue with step 11.

   ◆ If No, continue with step 12.

11. Do you want to specify an extended rule for this content particle?

   ◆ If Yes, select the **Loop Extended Rules** tab, define the rule, and continue with step 12.

     See Chapter 9, *Using Extended Rules* for more information about extended rules.

   ◆ If No, continue with step 12.

12. Click **OK**. The Map Editor saves the content particle and closes the Content Particle Properties dialog box.

Managing Pcdata

A pcdata object contains character data in an XML document. Only one pcdata object can be defined per element or content particle. Gentran Integration Suite automatically names the pcdata object with the name of the parent element or content particle.

Note: When a pcdata has an operation performed against it (link, standard rule, or as an extended rule storage field), the Map Editor displays a red check mark over the pcdata icon.

For information about properties, press F1 for Help or see Appendix B, *Map Editor Properties*.

Note: By default, the translator always trims "white space" from XML Pcdata fields during input. To control this behavior (to deactivate the trimming of Pcdata white space by the translator during input), add the line **trimPcdata=false** to the **translator.properties** file in the Properties directory.

For more information, see *Creating a Pcdata* on page 148.

Creating a Pcdata

To create a pcdata:

1. Right-click a map object and select either **Create Sub** or **Insert** from the shortcut menu.

2. From the shortcut menu, select **Pcdata**. The Map Editor displays the **Pcdata Properties** dialog box.

3. In the **Validation** tab, specify the following:

   - Whether the pcdata is required

   - Minimum length

   - Maximum length

   - Type of data

   - How the data is formatted

4. Do you want to specify an extended rule for this pcdata?

   - If Yes, click the **Extended Rule** tab, define the rule, and continue with step 5.

     For more information about extended rules, see Chapter 9, *Using Extended Rules*.

   - If No, continue with step 6.

5. Do you want to specify a standard rule for this pcdata?

   - If Yes, click the **Standard Rule** tab, define the rule, and continue with step 6.

     For more information about standard rules, see Chapter 8, *Using Standard Rules*.

   - If No, continue with step 6.

6. Click **OK**. The Map Editor saves the pcdata and closes the **Pcdata Properties** dialog box.

Managing Attributes

In Gentran Integration Suite, each attribute is contained in an attribute container. An element can only have one attribute container object, but the attribute container object can enclose many attribute objects.

Note: When an attribute has an operation performed against it (link, standard rule, or as an extended rule storage field), the Map Editor displays a red check mark over the attribute icon.

For information about properties, press F1 for Help or see Appendix B, *Map Editor Properties*.

For more information, see:

✦ *Attribute Container Object* on page 150

✦ *Attribute Object* on page 150

Attribute Container Object

The *attribute container object* does not correspond to an XML feature. Gentran Integration Suite uses attribute container objects to contain the attributes of an XML element, so attribute containers do not have properties. An attribute container object is automatically created when you create the first attribute of an XML element. Subsequent attribute objects are created in the existing attribute container object.

Attribute Object

The *attribute object* specifies information associated with an element that further defines the element.

Creating an Attribute

To create an attribute:

1. Right-click a map object and select either **Create Sub** or **Insert** from the shortcut menu.

2. From the shortcut menu, select **Attribute**. The Map Editor displays the Attribute Properties dialog box.

3. In the **Name** tab, specify the following:

    ◆   Unique attribute name

    ◆   Description (if applicable)

    ◆   Additional notes

    Note: The Additional notes box is populated automatically with the XML attribute annotation when you import an XML schema.

4. If necessary, click the **Tag** tab and change the value in the **Tag** box.

    Note: The value in the Tag box must match the attribute tag in the XML document. XML tags *cannot* contain spaces.

5. Click the **Type** tab to access the attribute type options.

6. In the **Type** tab, specify the following:

    ◆   Attribute type

    ◆   Default usage of the attribute

◆ Default value (only if you selected Default Exists or Fixed)

> **Note:** To use an enumerated attribute, you must also create a code list and use a code list standard rule with the attribute.

7. Click the **Validation** tab to access the validation options.

8. In the **Validation** tab, specify the following:

   ◆ Minimum length

   ◆ Maximum length

   ◆ Type of data

   ◆ How the data is formatted

9. Do you want to specify an extended rule for this attribute?

   ◆ If Yes, click the **Extended Rule** tab, define the rule, and continue with step 10.

     See the *Chapter 9, Using Extended Rules* for more information about extended rules.

   ◆ If No, continue with step 11.

10. Do you want to specify a standard rule for this attribute?

    ◆ If Yes, click the **Standard Rule** tab, define the rule, and continue with step 11.

      See the *Chapter 9, Using Extended Rules* for more information about extended rules.

    ◆ If No, continue with step 11.

11. Click **OK**. The Map Editor saves the attribute and closes the Attribute Properties dialog box.

About XML Namespaces

An *XML namespace* is a Uniform Resource Identifier (URI). A URI can be either a Uniform Resource Locator (URL) or a Uniform Resource Name (URN) that is uniquely associated with an element or attribute name.

Namespaces can:

✦ Uniquely identify element and attribute names so that there is no data confusion when the names are transmitted. If two or more organizations use the same element or attribute names in their XML documents, when they transmit those documents outside of the organization, the names can collide.

✦ Provide context for element and attribute names.

Namespaces are not required. However, for documents that are transmitted outside of the organization or in large organizations where similar names might be used by different departments, they are invaluable.

Here is an example of a namespace declaration for the element *order*:

```
<order xmlns='uuid.12345...67890' />
```

If a namespace is declared for an element, any subordinate elements or attributes inherit that namespace by default. However, a subordinate element or attribute can have a separate namespace defined for it.

Here are examples of namespace declarations for an element and a subordinate element:

```
<order xmlns='uuid.12345...67890' >
   <number xmlns='uuid.09876...54321, />
</order>
```

For more information, see *About Namespace Prefixes* on page 152.

About Namespace Prefixes

As a convenience, you can specify a prefix (an alias) for a namespace. A prefix is not an informational part of the element, attribute name, or namespace, and a prefix can be used with both element and attribute names. However, you can specify a default namespace and omit the prefix.

In the following example of a namespace declaration, *abc* is the prefix:

```
<order xmlns:abc='uuid.1234567890' >
```

If a namespace with prefixes is declared for an element, any subordinate elements or attributes inherit that namespace and the prefixes, as shown in this example:

```
<order xmlns:='uuid.1234567890'
      xmlns:aaa='uuid.09876...54321' >
  <aaa:number />
</order>
```

You can also write the namespace declaration without the prefix:

```
<order xmlns:='uuid.1234567890' >
  <number xmlns='uuid.09876...54321' />
</order>
```

For more information about XML namespaces, see the W3C document at www.w3.org/TR/1999/REC-xml-names-19990114/.

Using Namespaces in the XML File

If you enable namespaces at the root level of the XML file, you can specify namespaces using the **Namespace** tab.

For more information, see:

✦ *Using Namespaces on the Input Side of the XML Map* on page 153
✦ *Using Namespaces on the Output Side of the XML Map* on page 153

Using Namespaces on the Input Side of the XML Map

You must identify namespaces for incoming documents in envelopes so that Gentran Integration Suite can match the correct envelope with a document. The namespace and the tag must match for the translator to translate the data, and XML tags *cannot* contain spaces. You do not specify namespace prefixes on the input side of the map. They are defined in the document itself.

Using Namespaces on the Output Side of the XML Map

On the output side of the map, you must specify the prefixes that are used with the namespaces. You can define prefixes at the root level or for an element, but not for attributes. After the namespace prefix is defined, you must set up the elements that use the namespace

Defining a Namespace Prefix and URI

To define a namespace prefix and URI at the root level:

1. In the Map Editor, enable namespaces by right-clicking the **XML File** icon and selecting **Properties** from the shortcut menu.
2. Select the **Namespace** tab.
3. Select the **Enable Namespace Support** box.

    For more information about properties, press F1 for Help or see Appendix B, *Map Editor Properties*.

4. Select **New** to the right of **Define namespace prefixes**.
5. In the first box, type the namespace prefix (for example, SCI).

    Note: If a prefix is not required, leave the first box empty.

    ---

6. In the second box, type the namespace URI (for example, www.sterlingcommerce.com).
7. Repeat steps 4 through 6 for each namespace prefix required for the map.
8. Click **OK** to save the namespace information.

Setting Up Elements to Use a Namespace

After the namespace prefix is defined, you must set up the elements that use the namespace.

1. Right-click an **Element** icon and select **Properties** from the shortcut menu.

2. Select the **Namespace** tab.

3. Select **Use this namespace**.

> **Note:**   Do note type the namespace prefix in the box.

4. Click **OK** to save the namespace information.

5. If any of the elements also have the XML attribute created, define the namespace prefix and namespace URI for the element using the instructions for *Defining a Namespace Prefix and URI* on page 153.

Examples

In the following namespace declaration, the prefix is *SCI* and the namespace is www.sterlingcommerce.com:

```
<x xmlns:SCI='http://www.sterlingcommerce.com'>
</x>
```

In the following example of multiple namespace prefixes, one element defines two namespace prefixes, *bk* and *isbn*:

```
<bk:book xmlns:bk='urn:loc.gov:books'>
       xmlns:isbn='urn:ISBN:0-111-22222-3'>
   <bk:title>Numbers</bk:title>
   <isbn:number>0123456789</isbn:number>
</bk:book>
```

Using Standard Rules

The Map Editor provides you with standard rules that you can apply to fields. Standard Rules give you access to functions that are necessary for mapping operations more complex than simple linking but less complex than extended rules. Standard Rules are mutually exclusive (you can use only one at a time on a field). For information about extended rules, see Chapter 9, *Using Extended Rules*.

Note: If a standard or extended rule is applied to a map component, this is visually represented in the Map Editor by an asterisk (*) on the map icon for that map component.

This section covers the following topics:

✦ Using the System Variable Standard Rule

✦ Using the Use Constant Standard Rule

✦ Using the Loop Count Standard Rule

✦ Using the Use Accumulator Standard Rule

✦ Using the Use Code Standard Rule

✦ Using the Select Standard Rule

✦ Using the Update Standard Rule

Using the System Variable Standard Rule

The *System Variable standard rule* enables you to set a variable that maps the current date and time to the selected field. The selected map component must have a data type of Date/Time.

For more information, see *Standard Rule Tab: System Variable* on page 155.

Standard Rule Tab: System Variable

To configure the System Variable standard rule:

---

1. In the Map Editor, double-click an existing field or create a new one.

   The Field Properties dialog box opens.

2. Click the **Standard Rule** tab to access the standard rule options.

3. From the **standard rule** list, select **Use System Variable**.

4. From the **system variable** list, select **Current date and time** to map those variables to the field.

5. Click **OK** to save the standard rule.

Using the Use Constant Standard Rule

The *Use Constant standard rule* enables you to move a constant value to the specified field and indicate a qualifying relationship with another field.

Constants are used in maps to hold information that is needed later in the map, either for an output field or a conditional statement. Typically, constants move a constant value to the specified field or generate qualifiers (to indicate a qualifying relationship with another field).

Typically, constants are used to load common data into each positional record on a map. You create the constant and name it. You can hard code (type) a value that is loaded into the field that uses the constant. Or you can store information from a field in a constant by using an extended rule. However, to *use* a constant (the information that is stored in it), you must use a standard rule.

For more information, see:

✦ *Standard Rule Tab: Use Constant* on page 157
✦ *Configuring the Use Constant Standard Rule* on page 157
✦ *Creating and Editing Constants* on page 157
✦ *Deleting Constants* on page 158
✦ *Mapping Constants* on page 158
✦ *Generating Qualifiers* on page 158

Standard Rule Tab: Use Constant

The following table describes the fields and buttons on the Standard Rule tab (with the Use Constant standard rule selected):

Field or Button	Description
constant	Lists available constants. The selected constant is mapped to the current field. If the necessary constant is not present in the constant list, you must create it by using the **Map Constants** dialog box, which you can access by clicking the **Edit** button.
Edit	Accesses the **Map Constants** dialog box, which enables you to create, edit, and delete constants.
qualifier	Lists fields that you can use to qualify the selected map component in order to establish a qualifying relationship between the two map components. If the qualified field is not generated because of a lack of data, the constant is not moved to the current field.

Configuring the Use Constant Standard Rule

To configure the Use Constant standard rule:

1. In the Map Editor, double-click an existing field or create a new one.

2. In the **Field Properties** dialog box, click the **Standard Rule** tab.

3. From the standard rule list, select **Use Constant**.

4. Do you need to create or edit a constant to use for the current field?

    ◆ If Yes, click **Edit** to access the Map Constants dialog box.

      For more information about creating and editing constants, see *Creating and Editing Constants* on page 157.

    ◆ If No, go to the next step.

5. From the constants list, select the constant that you want to map to the current field.

6. Do you want to establish a qualifying relationship between the current map component and another component?

    ◆ If Yes, from the qualifiers list, select the field that Gentran Integration Suite uses to determine whether to run this standard rule (if the qualifying component contains data).

    ◆ If No, go to the next step.

7. Click **OK** to save the standard rule.

Creating and Editing Constants

To create or edit a constant so you can use it to store information:

1. From the Map Editor **Edit** menu, select **Constants**.

2. In the **Map Constants** dialog box, do you want to create a new constant?

    ◆   If Yes, click **New** and continue with step 3.

    ◆   If No (you want to edit an existing constant), select the constant and click **Edit** and continue with step 3.

    The Edit Constant dialog box opens.

3. In the **ID** field, type the constant identifier.

4. From the **Type** list, select the category of the constant.

5. In the **Value** field, type the constant expression.

6. Click **OK** to add the constant.

Deleting Constants

To delete a constant:

1. From the Map Editor **Edit** menu, select **Constants**.

2. In the **Map Constants** dialog box, select the constant you want to delete.

3. Click **Delete**. The constant is removed without a warning message.

Mapping Constants

To map a constant in which you previously stored data using an extended rule:

1. In the Map Editor, double-click the field in which you want to use the constant.

    The **Field Properties** dialog box opens.

2. Click the **Standard Rule** tab to access the 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** to load the data that was stored in the selected constant into the field.

Generating Qualifiers

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

To generate a qualifier:

1. In the Map Editor, double-click the field you want to use to further define (qualify) another field.

    The **Field Properties** dialog box opens.

2. Click the **Standard Rule** tab to access standard rule options.

3. From the standard rule list, select **Use Constant**.

4. From the qualifiers list, select the field that this field qualifies.

   This list contains only the other *active* fields in the same record or segment as the qualifying field.

5. Click **OK** to generate the qualifying relationship between the two fields.

Using the Loop Count Standard Rule

The *Loop Count standard rule* enables you to count the number of times a loop is repeated, if the field 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).

For more information, see *Configuring the Loop Count Standard Rule* on page 159.

Configuring the Loop Count Standard Rule

To configure the Loop Count standard rule:

1. In the Map Editor, double-click an existing field or create a new one.

   The Field Properties dialog box opens.

2. Click the **Standard Rule** tab to access the standard rule options.

3. From the standard rule list, select **Loop Count**.

4. Select the loop that you want to count. If the loop is a nested loop, you can track the current loop or the outer loop.

5. Click **OK** to save the standard rule.

Using the Use Accumulator Standard Rule

The *Use Accumulator standard rule* gives you access to a set of numeric variables that you can manipulate using numeric operations, and then transfer to and from fields. This rule enables you to add, change, or delete calculations for the field, including hash totals (used to accumulate numeric field values such as quantity and price). This rule also enables you to map the accumulated total into a control total field, and use accumulators. *Accumulators* count the occurrences of a specific field or generate increasing, sequential record, or line item numbers.

Note: Generally, the Accumulator Standard Rule must be placed on numeric elements or fields.

The accumulator has an start value of zero, unless you assign another value using the *accum* extended rule. For more information, see Chapter 9, *Using Extended Rules*.

For more information, see:

✦ *Standard Rule Tab: Use Accumulator* on page 160
✦ *Counting Line Items* on page 162
✦ *Calculating Hash Totals* on page 163
✦ *Resetting and Calculating a Value Total* on page 164

Standard Rule Tab: Use Accumulator

The following table describes the fields and buttons on the Standard Rule tab (with the Use Accumulator standard rule selected):

Field or Button	Description
Primary accumulator	Lists all existing calculations that were created for this field.
New	Accesses the **Edit Accumulator Entry** dialog box to create a new calculation for this field.
Change	Accesses the **Edit Accumulator Entry** dialog box to edit a calculation selected from the Primary accumulator field.
Delete	Deletes a calculation selected from the Primary accumulator field. **Note:** The selected calculation is deleted without a warning message.

For more information, see *Accumulator Operations* on page 160.

Accumulator Operations

For more information about the Edit Accumulator Entry dialog box, see Appendix B, *Map Editor Properties*.

The following table lists the available accumulator operations and their functions:

Operation	Function
Increment primary	Adds 1 (one) to the contents of the primary accumulator (Primary = Primary + 1).
Decrement primary	Subtracts 1 (one) 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).

Operation	Function
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 + 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). **Note:** The field must be a numeric field to display the accumulator value.
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 field with the contents of the primary accumulator, and stores the *remainder* of that operation in the primary accumulator (Primary = Primary % Field).
Modulo with field	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).
Negate primary	Makes the contents of the primary accumulator negative (Primary = Primary * -1). 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 field. 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 field, 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 field, 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 field, and stores the result in the primary accumulator (Primary = Primary / Alternate).
Modulo primary with alternate	Divides the contents of the primary accumulator with the contents of the Alternate Accum field, and stores the *remainder* of that operation in the primary accumulator (Primary = Primary % Alternate).

Counting Line Items

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

To count line items and generate a control total for a purchase order:

1. In the Map Editor, double-click the field that you typically use to count the line items.

    The Field Properties dialog box opens.

2. Click the **Standard Rule** tab to access the standard rule options.

3. From the standard rule list, select **Use Accumulator**.

4. Click **New**.

    The Edit Accumulator Entry dialog box opens to create a new calculation for this field.

5. From the **Primary Accumulator** list, select **0**.

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

6. In the **Name** box, type **Line Item Number**.

    The line item number is a descriptive alias that enables you to differentiate what the accumulators you create are used for.

7. From the **First** list, select **Increment primary**.

    Increment primary is the first operation performed. The value in the accumulator increments by one for each iteration of the current group.

8. From the **Second** list, select **Use primary**.

    Use primary is the second operation performed, after the first operation is completed. The current value of the accumulator is loaded into the Assigned Identification field.

9. In the **Edit Accumulator** dialog box, click **OK** to add the accumulator.

10. In the **Field Properties** dialog box, click **OK** to add the standard rule to the Line Items field.

11. Double-click the field that typically contains the total number of line items.

    The Field Properties dialog box opens.

12. Click the **Standard Rule** tab to access the standard rule options.

13. From the standard rule list, select **Use Accumulator**.

14. Click **New**.

    The Edit Accumulator Entry dialog box opens to create a new calculation for this field.

15. From the **Primary Accumulator** list, select **0**.

    This accumulator currently contains the total number of line items.

16. From the **First** list, select **Use primary**.

    Use primary specifies that the current value of the accumulator is loaded into the Number of Line Items Total field.

17. In the **Edit Accumulator** dialog box, click **OK** to add the accumulator.

18. In the **Field Properties** dialog box, click **OK** to add the standard rule to the Number of Line Items Total field.

    The Number of Line Items Total field now contains the total number of line items in the purchase order.

Calculating Hash Totals

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

To count the quantity ordered and generate a hash total for a purchase order:

1. In the Map Editor, double-click the field that you typically use to count the line items.

    The Field Properties dialog box opens.

2. Click the **Standard Rule** tab to access the standard rule options.

3. From the standard rule list, select **Use Accumulator**.

4. Click **New**.

    The Edit Accumulator Entry dialog box opens to create a new calculation for this field.

5. From the **Primary Accumulator** list, select **1**.

6. In the **Name** box, type **Total Quantity**.

7. From the **First** list, select **Sum in primary**.

    Sum in primary specifies that the numeric value of the field (taking the positive or negative sign of the numbers into account) is added to the contents of the primary accumulator for each iteration of the current group.

8. In the **Edit Accumulator** dialog box, click **OK** to add the accumulator.

9. In the **Field Properties** dialog box, click **OK** to add the standard rule to the Line Items field.

10. Double-click the field that typically contains the total quantity of the purchase order.

    The Field Properties dialog box opens.

11. Click the **Standard Rule** tab to access the standard rule options.

12. From the standard rule list, select **Use Accumulator**.

13. Click **New**.

    The Edit Accumulator Entry dialog box opens to create a new calculation for this field.

14. From the **Primary Accumulator** list, select **1**.

    This accumulator currently contains the total quantity.

15. From the **First** list, select **Use primary**.

    Use primary specifies that the current value of the accumulator is loaded into the Hash Total field.

16. In the **Edit Accumulator** dialog box, click **OK** to add the accumulator.

17. In the **Field Properties** dialog box, click **OK** to add the standard rule to the Hash Total field.

    The Hash Total field now contains the total quantity of the purchase order.

Resetting and Calculating a Value Total

In this example, you want to multiply the quantity invoiced for each line item by the unit price to obtain the extended price. Then, you want to generate a running total of extended price and load the final total into the Total Invoice Amount field.

Multiplying Quantity Invoiced by Unit Price

To multiply the quantity invoiced for each line item by the unit price to obtain the extended price for an invoice:

1. In the Map Editor, double-click the field that you typically use to count the quantity invoiced.

    The Field Properties dialog box opens.

2. Click the **Standard Rule** tab to access the standard rule options.

3. From the standard rule list, select **Use Accumulator**.

4. Click **New**.

    The Edit Accumulator Entry dialog box opens to create a new calculation for this field.

5. From the **Primary Accumulator** list, select **2**.

6. In the **Name** box, type **Extended Price**.

7. From the **First** list, select **Load primary**.

    Load primary specifies that the contents of the field are loaded into the primary accumulator for each iteration of the current group.

8. In the **Edit Accumulator** dialog box, click **OK** to add the accumulator.

9. In the **Field Properties** dialog box, click **OK** to add the standard rule to the Quantity Invoiced field.

10. Double-click the field that contains the unit price for each line item.

    The Field Properties dialog box opens.

11. Click the **Standard Rule** tab to access the standard rule options.

12. From the standard rule list, select **Use Accumulator**.

13. Click **New**.

    The Edit Accumulator Entry dialog box opens to create a new calculation for this field.

14. From the **Primary Accumulator** list, select **2**.

15. From the **First** list, select **Multiply with primary**.

    Multiply with primary specifies that the value of the Unit Price field is multiplied by the contents of the primary accumulator, and the result is stored in the primary accumulator for each iteration of the current group.

16. In the **Edit Accumulator** dialog box, click **OK** to add the accumulator.

17. In the **Field Properties** dialog box, click **OK** to add the standard rule to the Unit Price field.

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

Generating a Running Total of Extended Price

To generate a running total of the extended price:

1. In the Map Editor, double-click the field that contains the unit price for each line item.

    The Field Properties dialog box opens. You already established one accumulator that appears in the list on the Standard Rule tab.

2. Click **New**.

    The Edit Accumulator Entry dialog box opens to create a new calculation for this field.

3. From the **Primary Accumulator** list, select **3**.

4. In the **Name** box, type **Running Total**.

5. From the **First** list, select **Add primary to alternate**.

    Add primary to alternate specifies that the contents of the primary accumulator are added to the contents of the alternate accumulator, and the result is stored in the primary accumulator for each iteration of the current group.

6. From the **Alternate Accum** list, select **2**.

    The value of accumulator 2 (which contains the extended price for a line item) is added to the value of accumulator 3. The sum is stored in accumulator 3, which therefore contains a running total of the extended price with each iteration of the current group.

7. In the **Edit Accumulator** dialog box, click **OK** to add the accumulator.

8. In the **Field Properties** dialog box, click **OK** to add the standard rule to the Unit Price field.

Loading a Running Total of Extended Price

To load the running total of the extended price into the Total Invoice Amount field:

1. In the Map Editor, double-click the Unit Price field.

    The Field Properties dialog box opens.

2. Click the **Standard Rule** tab to access the standard rule options.

3. From the standard rule list, select **Use Accumulator**.

4. Click **New**.

    The Edit Accumulator Entry dialog box opens to create a new calculation for this field.

5. From the **Primary Accumulator** list, select **3**.

6. From the **First** list, select **Use primary**.

    Use primary specifies that the contents of the primary accumulator are loaded into the Total Invoice Amount field.

7. From the **Alternate Accum** list on the **Edit Accumulator** dialog box, select **2**.

    The value of accumulator 2 (which contains the extended price for a line item) is added to the value of accumulator 3. The sum is stored in accumulator 3, which therefore contains a running total of the extended price with each iteration of the current group.

8. In the **Edit Accumulator** dialog box, click **OK** to add the accumulator.

9. In the **Field Properties** dialog box, click **OK** to add the standard rule to the Total Invoice Amount field.

Using the Use Code Standard Rule

The *Use Code standard rule* enables you to use code lists to validate the contents of a field and use as a reference to look up an associated description for a field. A *code list* is a list of values and their corresponding descriptions. Code lists seldom change and are stored within the map itself. A field with a Use Code standard rule enables values to be either checked against or selected from the codes in a specified code list. Code lists are typically used to qualify another field.

You create code lists in the Map Editor as part of a map and manage them through the Map Editor. You can import and export code lists and copy and paste code lists between maps.

Code lists differ from the trading partner code lists used by the Select standard rule in that code lists are generally static and stored within the map file.

If you must attach a generic code list to a Gentran Integration Suite map, you have two options:

✦ Add the code list to the map, using the instructions in *Defining or Modifying a Code List* on page 168.

✦ Create a trading partner code list but do not specify the sender identity and receiver identity. Because the code list is generic, this ensures that it is "global," as it applies to all partners.

For more information, see:

✦ *Standard Rule Tab: Use Code* on page 167

✦ *Defining or Modifying a Code List* on page 168

✦ *Deleting a Code List or Code List Entry* on page 168

✦ *Importing a Code List* on page 169

✦ *Exporting a Code List* on page 169

✦ *Copying and Pasting Code Lists* on page 169

✦ *Validating Data Against Code Lists* on page 170

✦ *Mapping Code List Entry Descriptions* on page 170

Standard Rule Tab: Use Code

The following table describes the fields and buttons on the Standard Rule tab (with the Use Code standard rule selected):

Field or Button	Description
Code list	Lists all the code lists. If the necessary code list is not listed, click **Edit** to load or create a code list.
Edit	Accesses the **Edit Code List** dialog box, which enables you to create and edit code lists.
Raise compliance error	For compliance reasons, the field must contain one of the code list entries from the specified code list (nothing else is valid for that field). For example, if a field is defined as containing only YES or NO, you can create an exclusive code list that contains only YES and NO. Then if you receive a MAYBE in that field, Gentran Integration Suite flags it as an error.
Code description	Description of the code list entry (that is used) to appear when the selection is made. For example, if the code list entry is SU, it is much more useful to view the description of the code list entry (Supplier's Address). If you selected field XX from the store description list, the description for the code list entry used is mapped to field XX.

Defining or Modifying a Code List

To define or modify a code list:

1.   From the Map Editor **Edit** menu, select **Code Lists**.

2.   In the **Code Lists** dialog box, do you want to create a new code list?

     ◆   If Yes, click **New**.

     ◆   If No (you want to edit a code list), select a code list and click **Change**.

     The Edit Code List dialog box opens.

3.   In the **Table ID** box, type the name of the field for which this code list is used.

4.   In the **Description** box, type the description of the field for which this code list is used.

5.   Do you want to create a new code list entry?

     ◆   If Yes, click **New** and continue with step 6.

     ◆   If No (you want to edit a code list entry), select a code list entry and click **Change** and continue with step 6.

     The Edit Code List Entry dialog box opens.

6.   In the **Value** box, type the value of the code list entry.

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 - 8 to add more code list entries to the code list.

Deleting a Code List or Code List Entry

Caution: Be certain that you want to delete the code list or code list entry. When you click **Delete**, the selected code list (or code list entry) is deleted without a warning message.

To delete a code list:

1.   From the Map Editor **Edit** menu, select **Code Lists**.

2.   In the **Code Lists** dialog box, select the code list you want to delete.

3.   Click **Delete** to delete the code list.

To delete a code list entry:

1.   From the Map Editor **Edit** menu, select **Code Lists**.

2.   In the **Code Lists** dialog box, select the code list from which you want to delete a code list entry and click **Change**.

3.   In the **Edit Code List Entry** dialog box, select the code list entry and click **Delete**.

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

Importing a Code List

To import a code list:

1. From the Map Editor **Edit** menu, select **Code Lists**.

2. In the **Code Lists** dialog box, click **Import**.

3. In the **Open** dialog box, select the location of the code list.

   The default location is the Map Editor installation folder (the default location is Program Files\Sterling Commerce\Map Editor\Source Maps). The default file name extension for a code list is .cde.

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

Exporting a Code List

The Code List Export function enables you to export code lists to a file. You can define a code list for one map and use that code list in another map. This function also enables you to share code lists with other users of Gentran Integration Suite.

To export a code list:

1. From the Map Editor **Edit** menu, select **Code Lists**.

2. In the **Code Lists** dialog box, select a code list and click **Export**.

3. In the **Save As** dialog box, do you want to change the file name?

   ◆ If Yes, type the export file name in the **File name** box.

      The file name defaults to the table ID with a .cde file extension. The default location is the Map Editor installation folder (the default location is Program Files\Sterling Commerce\Map Editor\Source Maps).

   ◆ If No, go to the next step.

4. Click **Save** to export the code list.

Copying and Pasting Code Lists

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

To copy and paste a code list:

1. From the Map Editor **Edit** menu, select **Code Lists**.

2. In the **Code Lists** dialog box, select a code list and click **Copy**.

   The Map Editor copies the code list to the Clipboard.

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

4. Open the map in which you want to use the code list, if the map is not already open.

5. From the **Edit** menu, select **Code Lists**.

6. In the **Code Lists** dialog box, click **Paste**.

   The copied code list is added to the map.

Validating Data Against Code Lists

To validate data against a code list:

1. In the Map Editor, double-click the field for which you must validate data against a code table.

   The Field Properties dialog box opens.

2. Click the **Standard Rule** tab to access the standard rule options.

3. From the standard rule list, select **Use Code**.

4. Select the code list that the data in this field is validated against.

   If the code list is empty, you must create or load a code list.

5. If you must specify that the field *must* contain one of the code list entries from the specified code list (nothing else is valid for that field), select the **Raise compliance error** check box.

6. Click **OK**.

Mapping Code List Entry Descriptions

To map a code list entry description:

1. In the Map Editor, double-click the field for which you must map a code list entry description.

   The Field Propertie**s** dialog box opens.

2. Click the **Standard Rule** tab to access standard rule options.

3. From the standard rule list, select **Use Code**.

4. Select the code list that the data in this field is validated against.

   If this list is empty, you must create or load a code list.

5. If you must specify that the field *must* contain one of the code list entries from the specified code list (nothing else is valid for that field), select the **Raise compliance error** check box.

6. From the store description list, select the field to which you want the description of the code list entry to be mapped to when the selection is made.

7. Click **OK** to add the Use Code standard rule to the field.

Using the Select Standard Rule

The *Select standard rule* enables you to associate a field with a trading partner code list, a section in process data, a document envelope, the Gentran:Server for UNIX synonym table by in and out values, and use the transaction register function.

A description of the valid choices for the Select Standard Rule follow:

✦ Document Envelope – Gentran Integration Suite selects the specified TRADACOMS envelope value definitions and assigns them to fields in the map.

✦ Trading Partner Code List by Receiver Code – Gentran Integration Suite selects the receiver code in the trading partner code list specified in the **Please enter the name of the code list to use** box.

✦ Trading Partner Code List by Sender Code – Gentran Integration Suite selects the sender code in the trading partner code list specified in the **Please enter the name of the code list to use** box.

✦ Process Data – Gentran Integration Suite selects data from process data as specified in an XPath expression in the **Please enter the XPath to evaluate** box.

✦ Synonym Table by In Value – Gentran Integration Suite selects the data from the field on which the standard rule operates and queries the synonym table. If the data from the field matches data in the selected synonym table column, the translator returns that entire row from the synonym table. This rule is not supported in Gentran Integration Suite—this function enables easier migration of maps from Gentran:Server for UNIX to Gentran Integration Suite by enabling you to view the rule in your maps.

✦ Synonym Table by Out Value – Gentran Integration Suite selects the data from the field on which the standard rule operates and queries the synonym table. If the data from the field matches data in the selected synonym table column, the translator returns that entire row from the synonym table. This rule is not supported in Gentran Integration Suite—this function enables easier migration of maps from Gentran:Server for UNIX to Gentran Integration Suite by enabling you to view the rule in your maps.

✦ Transaction Register – Gentran Integration Suite invokes the check against the specified data (held in memory in Field1 through Field 6) to determine if the data is duplicate.

For more information, see:

Document Envelope (Select Standard Rule)

The *document envelope* function is used to support TRADACOMS enveloping to enable the map to extract values from the envelope definition and use them in the map. The TRADACOMS enveloping maps provided with Gentran Integration Suite use the document envelope function to build STX/END and MHD/MTR TRADACOMS envelope segments.

When the document envelope function is used, fields such as SenderID, ReceiverID, ControlNumber, and MessageType are extracted from the envelope definition. These values are then assigned to fields in the map and used to build the envelope segments according to the specified TRADACOMS enveloping map in the envelope definition.

The following table describes the fields on the Standard Rule tab with the Select standard rule selected and Document Envelope selected from the Table and key (or group) list:

Field	Description
Table and key (or group)	Document Envelope. Gentran Integration Suite selects the specified TRADACOMS envelope value definitions and assigns them to fields in the map.
Parameter	Type (exactly) the name of the parameter to select from the TRADACOMS document envelope.
Raise compliance error	For compliance reasons, the field must contain one of the values from the document envelope (nothing else is valid for the field for which this standard rule is located).
Map from	The value from the specified document envelope from which you want to map the contents. **Note:** Items are displayed in the map-from list only after you make a selection from the table and key list.
Map to	The field to which you want to map the document envelope value. **Note:** If you are working in the input side of the map, you can access all the fields in the current group. If you are working in the output side, you can access all the fields in the current record. A total of eight fields can be mapped using one Select standard rule. Entries are displayed in the map-to list after you select a table and key. These entries are activated after you select an entry in the map-from box.

Trading Partner Code List (Select Standard Rule)

A *trading partner code list* is a list of codes that are dynamic, related to trading partners, and stored in the Gentran Integration Suite database. The codes contained in the trading

partner code list consist of two code values that cross-reference an item between two trading partners, a description field, and four optional fields.

Trading partner code lists are created and managed in the Gentran Integration Suite interface.

Trading partner code lists differ from the code lists used in the Use Code standard rule. Those code lists are created and maintained in the Map Editor and stored in the map file.

The Select standard rule enables you to create one map that can be used with many trading partners, even if they use different code lists. It also enables you to change the content of a field according to a reference in the code list. For example, if a company trades the same item with several companies, and some reference the item by different names, the company can use one map, using the Select standard rule, for each transaction. This correctly names the item according to each company's preference.

The following table describes the fields on the Standard Rule tab with the Select standard rule selected and Trading Partner Code (List by Receiver Code or List by Sender Code) selected from the Table and key (or group) list:

Field	Description
Table and key (or group)	Trading Partner Code List (by Receiver Code or by Sender Code). The table and key from which Gentran Integration Suite selects data. Valid values are:
	◆ Trading Partner Code List by Receiver Code – Gentran Integration Suite selects the receiver code in the trading partner code list specified in the following field.
	◆ Trading Partner Code List by Sender Code – Gentran Integration Suite selects the sender code in the trading partner code list specified in the following field.
Code List	Type (exactly) the name of the trading partner code list to use (the name of the one you created in the Gentran Integration Suite interface).
Raise compliance error	For compliance reasons, the field must contain one of the codes from the specified trading partner code list (nothing else is valid for that field).
	For example, if a field is defined as containing only YES or NO, you can create a trading partner code list that contains only YES and NO. Then if you receive a MAYBE in that field, Gentran Integration Suite flags it as an error.
Map from	The code list entry from the specified code list from which you want to map the contents.
	Note: Items are displayed in the map-from list only after you make a selection from the table and key list.

Field	Description
Map to	The field to which you want to map the trading partner code list entries.
	Note: If you are working in the input side of the map, you can access all the fields in the current group. If you are working in the output side, you can access all the fields in the current record. A total of eight fields can be mapped using one Select standard rule. Entries are displayed in the map-to list after you select a table and key. These entries are activated after you select an entry in the map-from box.

Process Data (Select Standard Rule)

Process data is the service-independent XML document object model (DOM) associated with a business process that holds data associated with the business process, which you interact with using XPath. *XPath* is a World Wide Web Consortium (W3C) recommendation.

The following table describes the fields on the Standard Rule tab with the Select standard rule selected and Process Data selected from the Table and key (or group) list:

Field	Description
Table and key (or group)	Process Data. Gentran Integration Suite selects data from process data as specified in an XPath expression in the following field.
Please enter the XPath to evaluate	Type an XPath expression that identifies where in process data to access the data.
	Note: To access a field named trackingIdentifier found in ProcessData, the XPath would look like the following: **trackingIdentifier/text()**
Raise compliance error	For compliance reasons, the field must contain one of the codes from process data (nothing else is valid for that field).
Map from	The code list entry from the specified process data from which you want to map the contents.
	Note: Items are displayed in the map-from list only after you make a selection from the table and key list.
Map to	The field to which you want to map the data in process data.
	Note: If you are working in the input side of the map, you can access all the fields in the current group. If you are working in the output side, you can access all the fields in the current record. A total of eight fields can be mapped using one Select standard rule. Entries are displayed in the map-to list after you select a table and key. These entries are activated after you select an entry in the map-from box.

Synonym Table by In Value (Select Standard Rule)

The *synonym table by in value* function is used to support Gentran:Server for UNIX maps. This function enables you access the Gentran:Server for UNIX synonym table and search

a specified table column (In or Out) to match the data in the current map component. When a match is found, the translator returns the contents of that synonym table row into the field you specify.

In Gentran:Server for UNIX, this function enables you to configure cross-referencing so you can convert your values (for example, a part number) to your trading partner's values during outbound processing, and to convert your trading partner's values to your values during inbound processing.

> **Note:** This rule is not supported in Gentran Integration Suite—this function enables easier migration of maps from Gentran:Server for UNIX to Gentran Integration Suite by enabling you to view the rule in your maps.

The following table describes the fields on the Standard Rule tab with the Select standard rule selected and Synonym Table by In Value selected from the Table and key (or group) list:

Field	Description
Table and key (or group)	Synonym Table by In Value. Gentran Integration Suite selects the data from the field on which the standard rule operates and queries the synonym table. If the data from the field matches data in the selected synonym table column, the translator returns that entire row from the synonym table.
Sub-table to use	The name of the table to query.
Raise compliance error	For compliance reasons, the field must contain one of the values from the selected synonym table column (nothing else is valid for the field for which this standard rule is located).
Map from	The table column for which you want to query (In or Out) or the table name (Name). **Note:** Items are displayed in the map-from list only after you make a selection from the table and key list.
Map to	The field to which you want to map the synonym table row that contains data that matches the data in the current map component (the map component on which the standard rule operates). **Note:** If you are working in the input side of the map, you can access all the fields in the current group. If you are working in the output side, you can access all the fields in the current record. A total of eight fields can be mapped using one Select standard rule. Entries are displayed in the map-to list after you select a table and key. These entries are activated after you select an entry in the map-from box.

Synonym Table by Out Value (Select Standard Rule)

The *synonym table by out value* function is used to support Gentran:Server for UNIX maps. This function enables you access the Gentran:Server for UNIX synonym table and search a specified table column (In or Out) to match the data in the current map component. When

a match is found, the translator returns the contents of that synonym table row into the field you specify.

In Gentran:Server for UNIX, this function enables you to configure cross-referencing so you can convert your values (for example, a part number) to your trading partner's values during outbound processing, and to convert your trading partner's values to your values during inbound processing.

Note: This rule is not supported in Gentran Integration Suite—this function enables easier migration of maps from Gentran:Server for UNIX to Gentran Integration Suite by enabling you to view the rule in your maps.

The following table describes the fields on the Standard Rule tab with the Select standard rule selected and Synonym Table by Out Value selected from the Table and key (or group) list:

Field	Description
Table and key (or group)	Synonym Table by Out Value. Gentran Integration Suite selects the data from the field on which the standard rule operates and queries the synonym table. If the data from the field matches data in the selected synonym table column, the translator returns that entire row from the synonym table.
Sub-table to use	The name of the table to query.
Raise compliance error	For compliance reasons, the field must contain one of the values from the selected synonym table column (nothing else is valid for the field for which this standard rule is located).
Map from	The table column for which you want to query (In or Out) or the table name (Name). **Note:** Items are displayed in the map-from list only after you make a selection from the table and key list.
Map to	The field to which you want to map the synonym table row that contains data that matches the data in the current map component (the map component on which the standard rule operates). **Note:** If you are working in the input side of the map, you can access all the fields in the current group. If you are working in the output side, you can access all the fields in the current record. A total of eight fields can be mapped using one Select standard rule. Entries are displayed in the map-to list after you select a table and key. These entries are activated after you select an entry in the map-from box.

Transaction Register (Select Standard Rule)

Transaction register enables you to specify a field for which you want to see if it contains duplicate data (using the Update standard rule Transaction Register function) and then invoke it (using the Select standard rule Transaction Register function) to verify whether

the field contains duplicate data. Using the Update standard rule you can load document data into up to six fields in memory (Field 1 through Field 6).

> **Note:** You must define the data for which the translator will check by adding a Transaction Register Update standard rule prior to the point in the map where you invoke the Transaction Register Select standard rule to check for duplicate data. 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.

Then, using the Select standard rule Transaction Register function, you can invoke the check against the Transaction Register database table. If the data is validated as duplicate data, the translator notes the error in the translator report.

If there is no matching data already in the Transaction Register table, the content of Field1 through Field 6 is inserted as a row in the table.

> **Note:** If the document translation does not succeed, the field information (for which you have specified the translator should verify whether it is duplicate) is not added to the Transaction Register database table.

Purging the Transaction Register Database Table

By default, data in the Transaction Register table is deleted after thirty days (from the date the data was added to the table), so that the table does not continue to grow unchecked. If you wish, you can change the number of days after which data is deleted or remove this control altogether. See *Changing the Transaction Register Table Purge Control* on page 179 for more information about changing or removing the Transaction Register purge control.

> **Note:** You can use the Transaction Register function to view and delete data in the Transaction Register table.

The following table describes the fields on the Standard Rule tab with the Select standard rule selected and Transaction Register selected from the Table and key (or group) list:

Field	Description
Table and key (or group)	Transaction Register. Gentran Integration Suite invokes the check against the specified data (held in memory in Field1 through Field 6) to determine if the data is duplicate.
Raise compliance error	This value is not used for the Transaction Register function.
Map from	This value is not used for the Transaction Register function.
Map to	This value is not used for the Transaction Register function.

Mapping Trading Partner Code List Items

To map trading partner code list items:

1. In the Map Editor in the **Standard Rule** tab, verify that **Select** is selected as the standard rule and that either a **Trading Partner Code List** or **Process Data** is selected for the table and key to use.

2. Click an item in the list on the lower left.

3. Click a field in the list on the lower right.

4. Repeat as necessary.

Unmapping Trading Partner Code List Items

To unmap trading partner code list items, with the mapped pair of items selected, click the field on the right to clear and unmap it.

Cross-Referencing with the Gentran:Server for UNIX Synonym Table

To configure cross-referencing using the Gentran:Server for UNIX Synonym Table:

1. In the Map Editor in the **Standard Rule** tab, verify that **Select** is selected as the standard rule and that **Synonym Table by In Value** or **Synonym Table by Out Value** is selected for the table and key.

   > **Note:**   These rules are not supported in Gentran Integration Suite—this function enables easier migration of maps from Gentran:Server for UNIX to Gentran Integration Suite by enabling you to view the rule in your maps.

2. In the sub-table name box, type the exact name of the synonym table.

3. In the list on the lower left, select the column you want to query (select Out if you are using Synonym Table by In Value, select In if you are using Synonym Table by Out Value, or select Name if you want to return the database table name).

4. Click a field in the list on the lower right into which the row containing the matched data will be returned.

5. Repeat as necessary and click **OK** to add the standard rule.

Checking for Duplicate Data

To check for duplicate data:

1. In the Map Editor in the **Standard Rule** tab, verify that **Update** is selected as the standard rule and that **Transaction Register** is selected for the table and key to use.

2. Select the internal storage field (Field1 through Field 6) for which you want to assign the contents of the current field.

Note: You do not need to assign values to the storage fields in order (that is, assigning a value to Field1 first, then Field2, and so forth), and you can assign multiple fields to the same internal storage field (for example, the values of NAME, ADDRESS, and CITY may all be assigned to Field1). However only the last field for which the Update standard rule is executed will be stored.

---

3. Click **OK** to add the standard rule.

4. Repeat steps 1 through 3 to add as many Update standard rules as necessary.

5. Following the sequence of Update standard rules, select a map component and choose **Select** is as the standard rule.

6. Select **Transaction Register** is selected for the table and key to use.

Note: If the translator finds that the map component (for which the Update standard rule is in operation) contains data that is the same as that for which the matching Select standard rule is operating, it adds an error to the translator report noting the duplicate data.

---

7. Click **OK** to add the standard rule.

Changing the Transaction Register Table Purge Control

To change the number of days that data will be kept in the Transaction Register database table before being purged, or to remove the purge control altogether:

1. Using a text editor, open the **translator.properties.in** file, which can be found in *<install dir>*/properties in your Gentran Integration Suite installation.

2. To change the default number of days, specify the number of days you want the Transaction Register table to retain data in the mapper.maximumTransactionRegisterAge value (highlight the default 30 and type the number of days).

3. To remove the purge control, delete this entire line from the **translator.properties.in** file:

```
mapper.maximumTransactionRegisterAge=30
```

4. After you complete the edit, execute *<install_dir>***/bin/setupfiles.sh** and then stop and restart Gentran Integration Suite to verify the changes take effect.

Caution: If you edit the translator.properties.in file and **setupfiles.sh / .cmd** is subsequently executed for any reason, the changes you made *will be lost*. This is true for every properties file that has an ".in" version.

Using the Update Standard Rule

The *Update standard rule* enables you to move data from a field in the map into process data, to update process data so that the data can be used elsewhere in the business process, to record document-specific parameters during translation, and to check for duplicate data.

The specific functions that can be invoked follow:

✦   Document Extraction

✦   Process Data

✦   Correlation Data

✦   Transaction Register

✦   Transaction XREF

Document Extraction (Update Standard Rule)

In Gentran Integration Suite, the Update standard rule *document extraction* function is used to load values into the Sender ID, Receiver ID, and Accepter Lookup Alias field values in the Document Extraction table, so that the Document Extraction service can put these values in process data, along with an associated output document. The Document Extraction service is used to split individual documents out of a batch file to make each one a separate document. This service relies on:

✦   One or more translation maps to perform the extraction of each separate document, using:

   ◆   Extended rule string comparisons

   ◆   The *readblock* and *writeblock* extended rules

   to find the start of a single document and write out the blocks of data as the separate document until the end of the document is found)

✦   An Update standard rule to set Sender ID, Receiver ID, and Accepter Lookup Alias values from that document.

> **Note:**   An option is available in the Document Extraction service to batch together similar documents during the extraction process. If this option is specified, all the extracted documents that have the same Sender ID, Receiver ID, and Accepter Lookup Alias are batched into a single document.

The first step in using the Document Extraction service is creating a translation map to define how a single document *looks* and where the translator will find the Sender ID, Receiver ID, and Accepter Lookup Alias values. Defining how a document looks is defining where a document starts and ends, or where the next document begins; you define how a document looks with readblock and writeblock extended rules. See *readblock* on page 234 and *writeblock* on page 248 for more information about these extended rules.

Then, if you are using an EDI or Positional map format, you can use the Update standard rule Document Extraction function to set the Sender ID, Receiver ID, Accepter Lookup Alias, Receiver ID Qualifier (which supports qualifier lookups for EDI), and Sender ID

Qualifier (which also supports qualifier lookups for EDI) values for each document. The Update standard rule updates the Document Extraction table with this information. You can also use the Document Extraction function to set the EDI Standard value (expected values are X12, EDIFACT, TRADACOMS, and CII).

The following table describes the fields on the Standard Rule tab with the Update standard rule selected and Document Extraction selected from the Table (or group) list:

Field	Description
Table (or group)	Specifies the Document Extraction function.
Column (or field)	Column (or field) that Gentran Integration Suite updates with the contents of the current field. Valid values are SenderID, ReceiverID, AccepterLookupAlias, EDIStandard, ReceiverIdQualifier, and SenderIdQualifier. **Note:** The AccepterLookupAlias value will not permit these special characters in the field: ! @ # % ^ * ( ) + ? , < > { } [ ] | ; " '

Process Data (Update Standard Rule)

In Gentran Integration Suite, *process data* is the service-independent XML document object model (DOM) associated with a business process that holds data associated with the business process, which you interact with using XPath. *XPath* is a World Wide Web Consortium (W3C) recommendation.

The following table describes the fields on the Standard Rule tab with the Update standard rule selected and Process Data selected from the Table (or group) list:

Field	Description
Table (or group)	Specifies the Process Data function.
XPath to evaluate	XPath expression that directs the translator where in process data to perform the update.
Column (or field)	Column (or field) that Gentran Integration Suite updates with the contents of the current field. Valid value is XPath Result.

Correlation Data (Update Standard Rule)

The Update standard rule *correlation data* function enables you to record document-specific correlation parameters during translation. These correlation parameters are attached to the translated document. You can then use the Correlation Search user interface to locate the translated document using the criteria you specified in the map through the Update standard rule. This function saves you time, because you would otherwise need to locate the translated document by reviewing the results from a Central search query.

Using the Correlation Data Update standard rule, the Correlation service in a business process, or the Correlation Search interface you can, for example, retrieve all MaxxMart invoices or find out what happened to purchase order number 12345.

The Correlation service, which enables you to specify detailed search criteria that Gentran Integration Suite then uses to search for and display the documents that match your criteria, is used to create business process-level correlations from information in Process Data.

The following table describes the fields on the Standard Rule tab with the Update standard rule selected and Correlation Data selected from the Table (or group) list:

Field	Description
Table (or group)	Specifies the Correlation Data function.
correlation parameter name	The unique name of the correlation name-value pair.
Column (or field)	Correlation Value.

Transaction Register (Update Standard Rule)

Transaction register enables you to specify a field for which you want to check for duplicate data (using the Update standard rule Transaction Register function) and then invoke it (using the Select standard rule Transaction Register function) to verify whether the field contains duplicate data. Using the Update standard rule you can load document data into up to six fields in memory (Field1 through Field6).

Note: You must define the data for which the translator will check by adding a Transaction Register Update standard rule prior to the point in the map where you invoke the Transaction Register Select standard rule to check for duplicate data. 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.

Then, using the Select standard rule Transaction Register function, you can invoke the check against the Transaction Register database table. If the data is validated as duplicate data, the translator notes the error in the translator report.

If there is no matching data already in the Transaction Register table, the content of Field1 through Field6 is inserted as a row in the table.

Note: If the document translation does not succeed, the field information (specified for duplicate verification) is not added to the Transaction Register database table.

The following table describes the fields on the Standard Rule tab with the Update standard rule selected and Transaction Register selected from the Table and key (or group) list:

Field	Description
Table (or group)	Transaction Register. Gentran Integration Suite updates the selected field repository with the data from a specified field (in the Column (or field) list) so that other fields may be later validated against that data to prevent duplicate items.
Column (or field)	Column (or field) the translator will update with the information from the current map component. Valid values are Field1 through Field 6.

Field Size Limits

The following table describes the limits on the maximum size of the data that may be inserted in Field1 through Field6 (any data over the maximum limit is truncated—for example if data mapped to Field2 contains 40 characters, only the first 35 characters is stored in the database and used for duplicate checking):

Field	Maximum Size Limit
Field1	150 characters
Field2	35 characters
Field3	35 characters
Field4	35 characters
Field5	35 characters
Field6	30 characters

Transaction XREF (Update Standard Rule)

Transaction XREF is a function that enables you cross-reference the application data to generated transaction. ExternalDataImpl populates these values during translation, and after each round of translation completes, the ACH Enveloping service can query the harness for these values. Thus, when the ACH Enveloping service performs the enveloping, it knows what the standard specific values are for the four parameters specified.

Based on these two sets of values for identifiers (application specific and standard specific), the ACH Enveloping service populates a TransactionCrossReferenceTable in the database. You can also build a custom application to query this table to uniquely link an application file and generated ACH transaction.

Note: This rule only functions when used with the ACH Enveloping service. Transaction XREF rules, if used, should be defined in the Entry Detail Map.

The following table describes the fields on the Standard Rule tab with the Update standard rule selected and Transaction XREF selected from the Table and key (or group) list:

Field	Description
Table (or group)	Transaction XREF. Gentran Integration Suite cross-references the selected application data with the data from a specified transaction (in the Column (or field) list).
Column (or field)	Column (or field) the translator will update with the information from the current map component. Valid values are SenderID, ReceiverID, MessageType, Identifier.

Setting an Update Standard Rule as Part of Document Extraction

To use the Update standard rule as part of document extraction:

1. Select the element (on the input side of the document) that contains the Sender ID value.

2. Right-click the appropriate element and select **Properties** from the shortcut menu. The **Element Properties** dialog box (Name tab) opens.

3. Click the **Standard Rule** tab to access the standard rule options.

4. From the standard rule list, select **Update**.

5. From the table (or group) to update list, select **Document Extraction**.

6. From the column (or field) to update list, select **SenderID**.

7. Click **OK** to save the Sender ID update rule.

8. Select the element (on the input side of the document) that contains the Receiver ID value.

9. Right-click the appropriate element and select **Properties** from the shortcut menu. The **Element Properties** dialog box (Name tab) opens.

10. Click the **Standard Rule** tab to access the standard rule options.

11. From the standard rule list, select **Update**.

12. From the table (or group) to update list, select **Document Extraction**.

13. From the column (or field) to update list, select **ReceiverID**.

14. Click **OK** to save the Receiver ID update rule.

15. Select the element (on the input side of the document) that contains the Accepter Lookup Alias value.

16. Right-click the appropriate element and select **Properties** from the shortcut menu. The Map Editor displays the **Element Properties** dialog box (Name tab).

17. Click the **Standard Rule** tab to access the standard rule options.

18. From the standard rule list, select **Update**.

19. From the table (or group) to update list, select **Document Extraction**.

20. From the column (or field) to update list, select **AccepterLookupAlias**.

21. Click **OK** to save the Accepter Lookup Alias update rule.

22. See *readblock* on page 234 and *writeblock* on page 248 for more information about configuring the appropriate standard rules.

23. Right-click the appropriate element that you want to contain the EDI Standard and select **Properties** from the shortcut menu. The **Element Properties** dialog box (Name tab) opens.

24. Click the **Standard Rule** tab to access the standard rule options.

25. From the standard rule list, select **Update**.

26. From the table (or group) to update list, select **Document Extraction**.

27. From the column (or field) to update list, select **EDIStandard**.

28. Click **OK** to save the update rule.

Setting an Update Standard Rule as Part of Using Correlations

To use the Update standard rule as part of using correlations:

1. Select the field that contains the information with which you want to update the correlation data.

2. Right-click the appropriate field and select **Properties** from the shortcut menu. The Map Editor displays the **Element Properties** dialog box (Name tab).

3. Click the **Standard Rule** tab to access the standard rule options.

4. From the standard rule list, select **Update**.

5. From the table (or group) to update list, select **Correlation Data**.

6. In the correlation parameter name box, type the name of the correlation parameter.

7. From the column (or field) to update list, select **Correlation Value**.

8. Click **OK** to save the update rule.

Checking for Duplicate Data

To check for duplicate data:

1. In the Map Editor in the **Standard Rule** tab, verify that **Update** is selected as the standard rule and that **Transaction Register** is selected for the table and key to use.

2. Select the internal storage field (Field1 through Field 6) for which you want to assign the contents of the current field.

Note: You do not need to assign values to the storage fields in order (that is, assigning a value to Field1 first, then Field2, and so forth), and you can assign multiple fields to the same internal storage field (for example, the values of NAME, ADDRESS, and CITY may all be assigned to Field1). However only the last field for which the Update standard rule is executed will be stored.

For more information about the size limitations of Field1 through Field6, see *Field Size Limits* on page 183.

---

3. Click **OK** to add the standard rule.

4. Repeat steps 1 through 3 to add as many Update standard rules as necessary.

5. Following the sequence of Update standard rules, select a map component and choose **Select** as the standard rule.

6. Select **Transaction Register** is selected for the table and key to use.

Note: If the translator finds that the map component (for which the Update standard rule is in operation) contains data that is the same as that for which the matching Select standard rule is operating, it adds an error to the translator report noting the duplicate data.

---

7. Click **OK** to add the standard rule.

Using Extended Rules

You can use extended rules to define more complex translations than are available through the link function (simple mapping) and standard rules. You can use extended rules to define operations that are not possible using standard rules.

This section covers the following topics:

✦ About Extended Rules

✦ About Extended Rule Processing

✦ Defining Extended Rules

✦ Extended Rule Keywords and Commands

✦ Common Statements and Examples

✦ Alphabetical Language Reference

✦ Select and Update Available Options

About Extended Rules

> **Note:** If a standard or extended rule is applied to a map component, this is visually represented in the Map Editor by an asterisk (*) on the map icon for that map component.

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

The *declarations* section is required only if you use additional variables. The declarations section is where you declare the names and types of any variables you use either in the extended rule

The *statements* section is where you define the actions that you want the extended rule to run.

You must declare any variables that are not already defined as part of the input or output specification of the map before you use those variables in an extended rule.

For more information, see:

✦ *Declarations Section* on page 188

✦ *Statements Section* on page 189

Declarations Section

Variables consist of a name and a data type. Variable names can include alphanumeric characters and the colon (:) and underscore (_) characters. The first character in a variable name cannot be numeric. All variable names are case-sensitive.

Note: A declaration must be terminated with a semicolon (;). To improve readability, you typically include a blank line between the declaration and statements sections.

The translator does not initialize any variables. After you declare a variable, initialize it to prevent unexpected mapping results.

Supported Data Types

Extended rules support these data types:

✦ Integer – A whole number with no decimal component

✦ Real – A whole number with a decimal component

✦ String – One or more printable characters

✦ Datetime – A date or time

✦ Array – A table of multiple occurrences of a single data type

---

Note: Single-character strings are treated as a String; the "char" datatype is not supported by Map Editor extended rules.

Variable Declaration and Initialization Examples

This list contains a sample declaration and initialization of variables of each data type:

✦ Declare **i** as an integer and initialize **i**

integer i;

i = 0;

✦ Declare **r** as a real number and initialize **r**

real r;

r = 0;

✦ Declare **d** as a date or time and initialize **d**

datetime d;

d = date(0,0,0);

✦ Declare **s** as a 20-character string and initialize **s**

string[20] s;

s = "";

✦ Declare **a** as an array of 10 integers and initialize **each occurrence**

integer x;

x = 0;

while x < 11 do

  a[x] = 0;

✦ Declare **p** as an array of 50 10-character strings and initialize **each occurrence**

string[10] p[50];

integer x;

x = 0;

while x < 50 do

  p[x] = "";

Statements Section

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

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

An *expression* is a logical unit (for example, A = B or A + B) evaluated by Gentran Integration Suite. The statements section contains a sensible combination of expressions, keywords, commands, operators, and symbols. Each of these components is explained in detail in *Extended Rule Keywords and Commands* on page 194.

About Extended Rule Processing

You can specify pre- and post-session rules in the Session Rules dialog box. Pre-session extended rules are processed before the translation object is processed, and are applied to every extended rule defined in the map. Post-session rules are run after the translation object is processed.

The scope of an extended rule determines which variables are accessible from within the extended rule. The scope varies according to the following criteria:

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

✦ On Begin extended rules (defined in the Loop Extended Rules tab) apply until the conclusion of companion On End rules (also defined in the Loop Extended Rules tab).

> **Note:** For looping records or groups, On Begin and On End rules process once for each occurrence of the loop, at the beginning and end of each loop respectively.

✦ Field-level extended rules apply only for the duration of the field.

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

For example, if a current group to which the rule is attached is subordinate to another group, the parent group is automatically in scope for the duration of the child group, and the current hierarchical structure is also in scope for the duration of the child by using an addressing method. For more information about syntax and addressing methods, see *Symbol Syntax* on page 197.

An extended rule that is attached to a field applies only for the duration of the field. Field-level extended rules are always processed after standard rules.

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

The translator creates the data storage area for a map according to the structure of the input side (the source side) of the map. The data storage area (Internal Storage) is created in memory on the system performing the translation. Therefore, extended rules address the map according to the hierarchy of the input side. When you use extended rules, you must be careful to address the input side of the map so the translator can locate the map component that the rule accesses. On the output side of a map, extended rules can access only the current record and the input side of the map. However, from the input side of a map, extended rules can access the entire file structure. As long as you address the input side of a link, you can write to any field on the output side.

For more information, see:

✦ *Input Rule Processing* on page 190
✦ *Output Rule Processing* on page 191
✦ *Overview of Rule Processing* on page 192

Input Rule Processing

The translator processes the input side of the map first and then the output side.

> **Note:** Rules are invoked on *all* fields in a record *only* if there is data present for any field in that record.

The translator processes rules on the input side of a map in the following sequence:

1. In the Map Editor, load the input definition.

2. Read the input file.

3. Determine whether data is present for the first/next group and then run the On Begin rule, if present. If data is not present, continue with the next group in the map hierarchy.

4. Load each field in the group and run field-level rules in the following sequence:

   ◆ Run standard rules.

   ◆ Run extended rules.

5. At the end of the group, run the On End rule, if present.

6. Repeat steps 2 - 5 for each group in the input file.

Output Rule Processing

The translator processes rules on the output side of a map in the following sequence:

1. Determine whether data exists for the first/next record. If data is not present, continue with the next group in the map hierarchy.

2. If the record is the first record of a group, run the On Begin rule, if present.

3. For each field in the record, run field-level rules in the following sequence:

   ◆ Run standard rules.

   ◆ Run extended rules.

4. In the **Field Properties** dialog box, format data according to the specified field properties.

5. Write the record to the output file.

6. At the end of the group, run the On End rule, if present.

7. Repeat steps 1 - 6 for each record in the output file.

Overview of Rule Processing

The following figure shows when loop-level (On Begin and On End) extended rules and field-level extended rules are processed in relation to the overall business process flow:
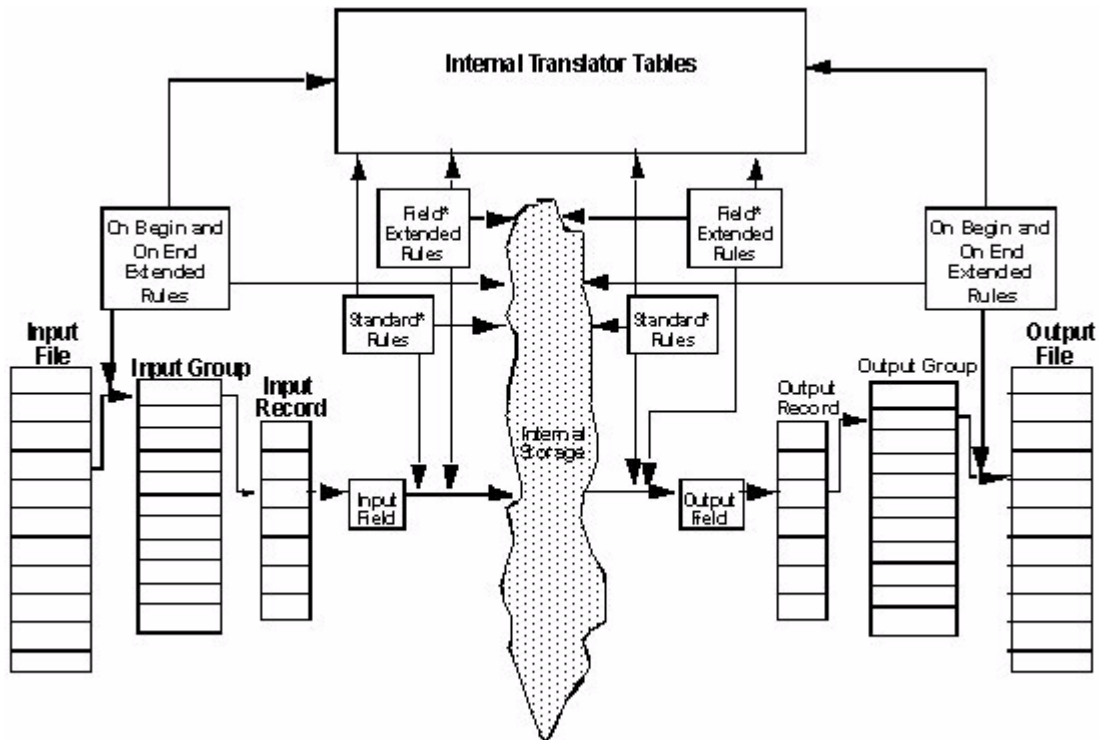
Defining Extended Rules

The map component an extended rule accesses depends on what you want the scope of the rule to be (that is, whether you want the rule to be in scope for the entire translation session or just when a specific map component is processed). It also depends on when you want the rule to be run (for example, before or after the map component is processed).

The process 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 map components:

✦   Entire session (input and output sides of the map)

✦   Looping map components (groups, records)

✦   Fields

For more information, see:

✦   *Defining a Session Rule* on page 193

✦ *Defining a Map Component Rule* on page 193

Defining a Session Rule

Pre-session rules define variables that have global scope (can be accessed from any other extended rule in the map). Pre-session extended rules are processed before the translation object is processed, and apply to every extended rule defined in the map.

Post-session rules are run after the translation object is processed and thus have no permanent scope.

To define a session rule:

1. From the Map Editor **Edit** menu, select **Session Rules**.

2. In the **Session Level Extended Rules** dialog box, select either **Pre-session** or **Post-session**.

3. In the **Editor** area, type the extended rule.

> **Note:** Click **Full Screen** to enlarge the area available for typing the extended rule.

4. Click **Compile** to compile the extended rule. Any warnings or errors are displayed in the Errors list.

> **Note:** Double-click an error to instantly navigate to the line containing the error.

The Compile function gives you immediate feedback about the accuracy of your rule. The rule is compiled when you compile the entire map.

5. Correct any errors that were flagged and click **Compile** again. Repeat this process until no errors are generated.

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

Defining a Map Component Rule

To define an extended rule for a map component:

1. In the Map Editor, right-click the map component. From the shortcut menu, select **Extended Rule(s)**.

Here are some possible responses:

- The Loop Extended Rules tab in the dialog box is displayed if the map component is a positional file, group, subgroup, or repeating record.

- The Extended Rule tab in the dialog box is displayed if the map component is a field.

2. Do you want the extended rule to be run before the map component is processed?

◆   If Yes, select the **On Begin** option.

◆   If No (you want the rule to be run when that map component is finished processing), select the **On End** option. The Map Editor processes On End rules at the end of each loop occurrence, not at the end of all loops.

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

3.   Type the extended rule.

For more information about rule syntax, see the *Alphabetical Language Reference* on page 210.

Note: Click **Full Screen** to enlarge the area available for typing the extended rule.

---

4.   Click **Compile** to compile the extended rule. Any warnings or errors are displayed in the Errors list.

Note: Double-click an error to instantly navigate to the line containing the error.

The Compile function gives you immediate feedback about the accuracy of your rule. The rule is compiled when you compile the map.

5.   Correct any errors that were flagged and click **Compile** again. Repeat this process until no errors are generated.

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

Extended Rule Keywords and Commands

Generally, a *keyword* is a fixed use of a word that indicates how the programming language must be interpreted. In Gentran Integration Suite, there are two types of keywords—execution control keywords and commands. *Execution control keywords* control the flow of execution of the defined rule. These keywords 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.

For information about specific keywords and commands, see *Alphabetical Language Reference* on page 210.

For more information about topics in this section, see:

✦   *Keywords* on page 194

✦   *Commands and Functions* on page 195

Keywords

The Gentran Integration Suite execution control keywords are:

- begin
- break
- continue
- end
- if...then...else
- while...do

Commands and Functions

The Gentran Integration Suite extended rule commands and functions are:

- accum
- atoi
- aton
- cerror
- collate
- concat
- continue
- count
- date
- delete
- empty
- eof
- exist
- get
- index
- left
- len
- messagebox
- mid
- ntoa
- numerrors
- readblock
- right
- select
- set
- sort
- strdate

✦ strstr
✦ trim
✦ trimleft
✦ trimright
✦ unreadblock
✦ update
✦ writeblock

Extended Rule Operators and Symbols

Operators define the simplest operation in an expression. You use operators and spaces to separate keywords and symbols. You cannot string two keywords together sequentially without an operator.

Operations are performed on symbols. In Gentran Integration Suite, you can use the following symbols in extended rules: variables, constants, map components/internal storage, arrays, and accumulators.

For more information, see:

✦ *Operators* on page 196
✦ *Symbols* on page 197

Operators

The following table lists the operators and their functions in extended rules:

Operator	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

Operator	Function
&	Logical and
|	Logical or
<<	Date modification

Symbols

Operations are performed on symbols. In Gentran Integration Suite, you can use the following symbols in extended rules:

✦ Variables

✦ Constants

✦ Map components/Internal storage

✦ Arrays

✦ Accumulators

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

You can use symbols to create extra line items when one line item field is already defined in internal storage.

Symbol Syntax

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

String Constant

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

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

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;
```

2 is a numeric constant value.

Addressing or Creating a Repeating Field in Internal Storage

To address a repeating field or create a repeating field in internal storage, within the scope of the current mapping action, the syntax is #FIELD_NAME[index]:

```
#field_1[3] = 2;
```

2 is a numeric constant value.

For more information about using indexes in maps, see Appendix D, *Using Indexes in the Map Editor and Translator*.

For more detailed information about using indexes with XML in the Map Editor, see Using Indexes in Common Mapper.doc from the Sterling Commerce Web site (www.sterlingcommerce.com and then select **Customer Support>Support on Demand**).

Note: Support on Demand requires a user name and password, so you must register if you have not completed this step already.

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.

Addressing or Creating a Repeating Field in a Group

To address a repeating field within a group or create a repeating field within a group in internal storage within the scope of the current hierarchy, the syntax is $GROUP.#FIELD_NAME[index].

For more information about using indexes in map, see Appendix D, *Using Indexes in the Map Editor and Translator*.

For more detailed information about using indexes with XML in the Map Editor, see Using Indexes in Common Mapper.doc from the Sterling Commerce Web site (www.sterlingcommerce.com and then select **Customer Support>Support on Demand**).

Note: Support on Demand requires a user name and password, so you must register if you have not completed this step already.

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
```

You are specifying the second instance of Group_C within the third instance of Group_B.

The following figure shows this relationship:
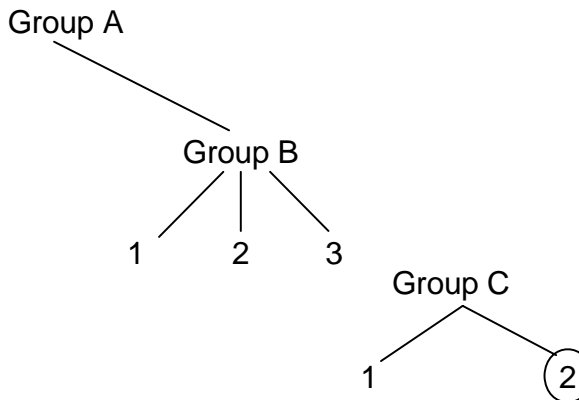
Group A

Group B

1   2   3

Group C

1   2

For more information about using indexes in maps, see Appendix D, *Using Indexes in the Map Editor and Translator*.

For more detailed information about using indexes with XML in the Map Editor, see Using Indexes in Common Mapper.doc from the Sterling Commerce Web site (www.sterlingcommerce.com and then select **Customer Support > Support on Demand**).

Note: Support on Demand requires a user name and password, so you must register if you have not completed this step already.

Addressing or Creating a Repeating Field in Internal Storage

Addressing or creating a repeating field in internal storage is similar to how you address or create a group in internal storage; however, for a repeating field, you can specify an instance of the field:

```
$Group_C[3][2].#Field_2[5]
```

You are specifying the fifth instance of field two within 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 field 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 field of the array is addressed individually:

```
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 assign a value to a specific accumulator (to write data into the accumulator), use the syntax accum(*n*), where *n* is the number (not the name) of the accumulator:

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

To assign the value of a field to equal a specific accumulator (to read data out of an accumulator), use the syntax accum(*n*), where *n* is the number (not the name) of the accumulator:

```
#field = accum(3)
```

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= integer variable that indicates the specific
                            occurrence of a repeating field or field within a
                            repeating composite.
```

This 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, #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 with Symbols

A statement is a single operation that consists of a combination of expressions, keywords, commands, operators, and symbols. The following figure shows an example of a simple statement, including symbols:
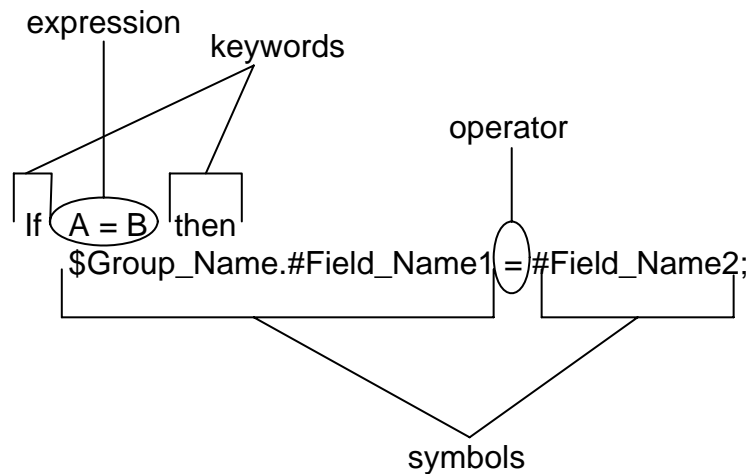
Common Statements and Examples

This section lists examples of the following common statements:

✦ Assignment

✦ Datetime Expressions

✦ Conditional Logic

✦ String Conditions and Functions

✦ Numeric Functions

✦ Raise Compliance Error Function (cerror)

✦ Number of Errors Function (numerrors)

✦ Remove Field Value Function (empty)

✦ Existence of Data Function (exist)

✦ Count Function (count)

✦ Delete Function (delete)

✦ Data Block Functions (readblock, unreadblock, writeblock)

Assignment

The assignment statement is the most powerful and most often used extended rule statement. In its simplest form, it has the following syntax:

```
variable=expression
```

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

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

Numeric Expression

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

String Expression

A string expression can consist of string constants, string fields, string variables, and string functions concatenated with the plus sign (+).

Datetime Expression

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

Examples

Here are some examples of assignment expressions:

```
a = 5;
a = b + c;
```

```
s = "hello";
s = s + "world";
```

Gentran Integration Suite Map Editor Guide

Datetime Expressions

Datetime expressions consist of a datetime variable and (optionally) datetime modifiers. If you are using the standard syntax, you can use datetime constants to write datetime expressions:

✦ Year/month/day

✦ Hour:minute:second

✦ Year/month/day/hour:minute:second

You can also write datetime expressions using datetime fields, variables, or date and time functions.

Date Syntax

Date functions have the following syntax (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 type into a datetime type by indicating a format mask (**"%y/%m/%d"**) along with the string (**"95/4/6"**) you want to convert. Use this function if you are using nonstandard syntax and must specify the syntax you are using.

For more information about date syntax, see *date* on page 223.

For example, if you want to add five days to a date in a field, you must:

1. Verify that you are using a standard rule that loads the current date into a field.

   For more information about loading a date into a field, see *Using the System Variable Standard Rule* on page 155.

2. Use the following extended rule logic to add five days to the date in that field:

```
#TheField = #TheField << days(5);
//The field is called #TheField
```

<< Operator

You can use the << operator to modify your datetime variable by adding time increments. Valid modifiers are:

✦ years

✦ months

✦ weeks

✦ days

✦ hours

✦ minutes

✦ seconds

For example:

```
datetime d;
d=#Date_Field <<weeks(2);
//This adds 2 weeks to the date in the date field named Date_Field
```

Time Syntax

Time functions have the following syntax:

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

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

```
datetime d;
d=#Time_Field <<seconds(1);
//This adds 1 second to the time field named Time_Field
```

get and set Syntax

The get and set functions enable you to access (get) or modify (set) individual components of a datetime type. For more information about the get and set functions, see *get* on page 227 and *set* on page 238.

Here is the syntax to use:

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

Conditional Logic

The Map Editor uses conditional logic to test conditions and then, depending on the results of the test, perform various operations. Conditions can be nested to any level. Do not end conditions with a semicolon (;). This terminating syntax is necessary for statements only.

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.

Gentran Integration Suite interprets the value as either true or false—a zero value as false, and a nonzero value as true.

> **Note:** 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 BEGIN and END.

The Map Editor evaluates the IF/THEN condition and, if it is true, runs all the statements that follow the THEN keyword. If the condition is false, it does not run any of the statements following THEN.

Use the ELSE keyword with IF/THEN to define several blocks of statements, one of which is run. The Map Editor tests the first IF/THEN condition. If the condition is false, The Map Editor proceeds to test each sequential condition until it finds one that is true. The Map Editor runs the corresponding block of statements for the true condition. If none of the IF/THEN conditions are true, The Map Editor runs the statements following the ELSE keyword. For more information about using the if, then, and else functions, see *if...then...else* on page 228.

Use this syntax:

```
IF condition THEN
BEGIN
statement1;
     statement2;
END
ELSE
BEGIN
     statement3;
     statement4;
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
```

You can also use the following string functions:

✦ left
✦ right
✦ mid
✦ strdate
✦ concat
✦ strstr
✦ trim

- ✦ trimleft
- ✦ trimright

left, right, mid

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 characters 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. For more information about the left function, see *left* on page 230. For more information about the right function, see *right* on page 237. For more information about the mid function, see *mid* on page 232. The following example shows 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

The strdate function converts a datetime type into a string using a format that you specify. This function enables you to include static characters such as a slash (/), which gives you access to full date support. For more information about the strdate function, see *strdate* on page 239.

Use this syntax:

```
datetime d;
string[8] s;

strdate(d,"%y/%m/%d",s);
```

concat

The concat function concatenates a specified number of characters from one string onto the end of another string. For more information about the concat function, see *concat* on page 221. In the following example, five characters from string s2 are concatenated onto the end of string s1:

```
string[10] s1,s2;
concat(s1,s2,5);
```

strstr

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. For more information about the strstr function, see *strstr* on page 241.

Use this syntax:

```
integer d;

d = strstr("hello", "el");
```

trim, trimleft, trimright

The trim, trimleft, and trimright functions enable you to remove white (unused) space or a specific character from a string value. By default these functions remove white space. If you want to remove a specific character, you must indicate that character in the function. For more information about the trim function, see *trim* on page 243. For more information about the trimleft function, see *trimleft* on page 244. For more information about the trimright function, see *trimright* on page 245.

Here is the syntax to follow to remove white space:

```
string = function(string_to_trim)

where: string = string field that contains the end result
       function = either trim, trimleft, or trimright
       string_to_trim = string field containing the value
                        to be trimmed
```

Here is the syntax to follow to remove a character:

```
string = function(string_to_trim, character_to_trim)

where: string = string field that contains the end result
       function = either trim, trimleft, or trimright
       string_to_trim = string field containing the value
                        to be trimmed
       character_to_trim = character to trim from string (no empty spaces)
```

Numeric Functions

Gentran Integration Suite provides the following numeric functions, which enable you to convert one data type to another.

✦ len

✦ atoi

✦ aton

✦ ntoa

len

The len function counts and returns the number of characters in a string. For more information about the len function, see *len* on page 230.

Here is an example of the syntax:

```
integer a;
a = len("hello");
```

atoi, aton, ntoa

The atoi function converts strings into integers. For more information about the atoi function, see *atoi* on page 212.

The aton function converts strings into real numbers. For more information about the aton function, see *aton* on page 213.

The ntoa function converts real numbers into strings. For more information about the ntoa function, see *ntoa* on page 232.

Here is an example of the syntax:

```
integer a;
real b;
string[8] s;
      a = atoi("5");
      b = aton("5.5");
      ntoa(5.5, s);
```

Raise Compliance Error Function (cerror)

The cerror function raises a compliance error. 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 which is written to the translator report as part of the entry for the compliance error you are raising. For more information about the cerror function, see *cerror* on page 214.

The following example raises compliance error 100 on the FIELDNAME field of the specified instance of the GROUPNAME group:

```
cerror(100,$GROUPNAME[1][1][1].#FIELDNAME, "Extra error information can be
supplied here");
```

Number of Errors Function (numerrors)

The numerrors function returns an integer which contains the current count of errors in the translation report (the count at the time the rule is executed). For more information about the numerrors function, see *numerrors* on page 233.

Remove Field Value Function (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. For more information about the empty function, see *empty* on page 225.

The following example sets the value of the specified instance of the FIELDNAME field to null:

```
empty($GROUPNAME[1][1][1].#FIELDNAME);
```

Existence of Data Function (exist)

The exist function returns a nonzero (true) value if there is data in a specified field in internal storage. If there is not data in the specified field, this function returns a zero (false) value. This function is typically used as a part of a condition. For more information about the exist function, see *exist* on page 226.

The following example returns a nonzero value if the condition is true (data is present in the specified instance of the FIELDNAME 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($GROUPNAME[1][1][1].#FIELDNAME) THEN
```

Count Function (count)

The count function returns the number of iterations of a group (or repeating record). For more information about the count function, see *count* on page 222.

Note: When a count extended rule is performed on an empty group, the value of -1 is returned from count($GROUPNAME[*]).

The following example returns the total iterations of the GROUPNAME group within the third iteration of the parent group:

```
count(GROUPNAME[3][*]);
```

Delete Function (delete)

The delete function deletes the specified occurrence of a repeating record or group. For more information about the delete function, see *delete* on page 224.

Use this syntax:

```
delete(GROUPNAME[occurrence]);
```

Data Block Functions (readblock, unreadblock, writeblock)

The readblock function reads a data block (record) from the input file and places it into the argument of a string variable. The unreadblock function resets the input file pointer to the place it was previous to the prior readblock. The writeblock function writes the data contained in the argument of a string variable to the output file.

These functions are commonly used together to pass data blocks in bulk from the input file to the output file. They are useful for maps that are designed to envelope data.

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.

For more information about using the corollary Update standard rule to complete the document extraction, see *Setting an Update Standard Rule as Part of Document Extraction* on page 184.

These functions enable data to be parsed faster than it would be if each component was translated separately. Parsing with data block function is faster because it avoids parsing fields out of blocks.

For more information about the readblock function, see *readblock* on page 234. For more information about the unreadblock function, see *unreadblock* on page 246. For more information about the writeblock function, see *writeblock* on page 248.

Use this syntax:

```
readblock(string_variable);
unreadblock();
writeblock(string_variable);
```

Alphabetical Language Reference

This alphabetical language reference enables you to refer quickly and easily to information about specific keywords or commands.

The following table describes the typeface conventions used in this section:

Typeface	Description
Regular	In syntax, required information.
Brackets [ ]	In syntax, optional information.

This section uses the following programming guidelines:

✦ Keywords and commands are shown in all lowercase letters.

✦ Map components are shown in all uppercase letters.

✦ Two slashes (//) introduce comments.

This topic covers the following information:

- ✦ accum
- ✦ atoi
- ✦ aton
- ✦ begin
- ✦ break
- ✦ cerror
- ✦ collate
- ✦ concat
- ✦ continue
- ✦ count
- ✦ date
- ✦ delete
- ✦ empty
- ✦ end
- ✦ eof
- ✦ exist
- ✦ get
- ✦ if...then...else
- ✦ index
- ✦ left

- ✦ len
- ✦ messagebox
- ✦ mid
- ✦ ntoa
- ✦ numerrors
- ✦ occurrencetotal
- ✦ readblock
- ✦ resetoccurrencetotal
- ✦ right
- ✦ select
- ✦ set
- ✦ sort
- ✦ strdate
- ✦ strstr
- ✦ sum
- ✦ sumtotal
- ✦ trim
- ✦ trimleft
- ✦ trimright
- ✦ unreadblock
- ✦ update
- ✦ while...do
- ✦ writeblock

accum

The accum function is a numeric function that uses extended rule logic to reference an accumulator variable established in the accumulator standard rule.

Syntax

Use this syntax:

```
#numericfield = accum(number);
where: numericfield = numeric field, the accum function references the
accumulator, specified by its number
```

```
accum(accumulator_number) = numeric_value;
where: the accumulator specified by accumulator_number is initialized with the
numeric value given.
```

Examples

Examples of this function follow:

```
real c;
c = accum(1);
// c contains the value of accumulator one (1).
```

```
accum(3) = 0
// Accumulator 3 is initialized with a value zero (0).
```

atoi

The atoi function is a numeric function that converts strings into integers. The numeric functions enable you to convert one data type to another.

Syntax

Use this syntax:

```
int = atoi(string);
where:   int    = integer variable
         string =  string variable
```

Example

An example of this function follows:

```
integer a;
string[20] s;
s = "5";
a = atoi(s);
// "a" contains the value 5
```

Gentran Integration Suite Map Editor Guide

aton

The aton function is a numeric function that converts strings into real numbers. The numeric functions enable you to convert one data type to another.

Syntax

Use this syntax:

```
real = aton(string);
where:  real    = real number variable
        string  = string variable
```

Example

An example of this function follows:

```
real b;
string[20] s;
s = "5.5";
b = aton(s);
// "b" contains the value 5.5
```

begin

The begin function encloses a group of statements that form the body of an if/then/else statement or a while loop.

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 and end.

Note: Do not end conditions with a semicolon (;). This terminating syntax is necessary for statements *only.*

Syntax

Use this syntax:

```
if condition then
begin
    statement1;
    statement2;
end
```

break

The break function terminates the execution of the nearest enclosing while loop, and passes control to the statement that follows the end keyword. This function is generally used in complex loops to terminate a loop before several statements have been run.

Example

An example of this function follows:

```
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.
```

cerror

The cerror function raises a compliance error. 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 which is written to the translator report as part of the entry for the compliance error you are raising.

Syntax

Use this syntax:

```
cerror(100,$GROUPNAME[0][1][1].#FIELDNAME, "Extra error information can be
supplied here");
```

Example

An example of this function follows:

```
cerror(100,$GROUPNAME[0][1][1].#FIELDNAME);
//This raises compliance error 100 on the FIELDNAME field of the
//specified instance of the GROUPNAME group
```

Compliance Codes

The following tables list compliance error codes: general messages first, then SQL, EDI, and XML specific messages.

General Messages

The following table lists compliance errors for general messages:

Message Number	Message Type	Message Generated
12	Information	StartTime
13	Information	EndTime
14	Information	BlocksRead
15	Information	BlocksWritten
19	Information	ExecutionTimeMillis
20	Information	TranslationObjectName
21	Information	TranslationIsLightweight
25	Warning	BlockDataUnknown
100	Error	MandatoryDataMissing
101	Error	InsufficientRepeats
102	Error	TooManyRepeats
110	Error	IncorrectDataFormat
111	Error	DataNotMinLength
112	Error	DataExceedsMaxLength
113	Error	InvalidDate
120	Error	TooManyComponents
121	Error	TooManyCompositeElements
122	Error	UnsupportedDataType
123	Error	DataConversionError
140	Error	StandardRuleFailure
142	Error	StandardRuleUseCodeDataMissing
143	Error	StandardRuleDataConversionError
170	Error	ExtendedRuleFailure
171	Error	ExtendedRuleDataConversionError
300	Error	MandatoryBlockMissing

Message Number	Message Type	Message Generated
301	Error	MandatoryGroupMissing
316	Error	MaxUsageExceeded
400	Error	BlockProcessorInitializationFailure
401	Error	FieldProcessorInitializationFailure
10001	Information	BlockSignature
10002	Information	BlockCount **Note:** This message will only be written if you create the cerror extended rule at the field level. If you call the cerror extended rule at any other level in your map, it will not be written because it is only applicable at the field level.
10003	Information	BlockName **Note:** This message will only be written if you create the cerror extended rule at the field level. If you call the cerror extended rule at any other level in your map, it will not be written because it is only applicable at the field level.
10004	Information	FieldName **Note:** This message will only be written if you create the cerror extended rule at the field level. If you call the cerror extended rule at any other level in your map, it will not be written because it is only applicable at the field level.
10005	Information	FieldData
10006	Information	Exception
10007	Information	GroupName
10008	Information	FieldId
10009	Information	FieldNumber
10010	Information	Instance
10011	Information	RuleType
10012	Information	OnBeginRule
10013	Information	OnEndRule
10014	Information	RepeatCount

Message Number	Message Type	Message Generated
10015	Information	BlockData
10016	Information	BlockSignatureIDTag **Note:** This message will only be written if you create the cerror extended rule at the field level. If you call the cerror extended rule at any other level in your map, it will not be written because it is only applicable at the field level.
10017	Information	MapIterationCount **Note:** This message will only be written if you create the cerror extended rule at the field level. If you call the cerror extended rule at any other level in your map, it will not be written because it is only applicable at the field level.
10018	Information	AdditionalInformation

SQL Messages

The following table lists compliance errors for SQL-specific messages:

Message Number	Message Type	Message Generated
700	Error	SQLDataSourceOpenError
701	Error	SQLDataSourceRollback
702	Error	SQLDataSourceCommitError
703	Error	SQLDataSourceRollbackError
710	Error	SQLQueryOpenError
711	Error	SQLCommandError
712	Error	SQLCursorError
713	Error	SQLGetFieldError
721	Error	SQLOutputOperationError
722	Error	SQLPreparedStatementError
724	Information	SQLUpdateEffected0Rows
725	Information	SQLRetryingAsInsert
726	Information	SQLRetryingAsUpdate
10700	Information	DataSourceName
10701	Information	DataSourcePool
10702	Information	QueryName
10703	Information	SQLStatement

Message Number	Message Type	Message Generated
10704	Information	CursorOperation
10705	Information	ColumnId

EDI Messages

The following table lists compliance errors for EDI-specific messages:

Message Number	Message Type	Message Generated
103	Information	IllegalRepeatingDelimiter
104	Information	IllegalSubElementDelimiter
105	Information	ElementPosition
106	Information	SubElementPosition

XML Messages

The following table lists compliance errors for XML-specific messages:

Message Number	Message Type	Message Generated
610	Error	XMLParticleORGroupError
690	Error	XMLParserError
691	Error	XMLElementUnknown
692	Error	XMLPcDataUnknown
693	Error	XMLAttributeUnknown
10060	Information	PublicId
10061	Information	SystemId
10062	Information	LineNumber
10063	Information	ColumnNumber
10064	Information	Message
10065	Information	XMLTagName
10066	Information	XMLNamespaceURI

The cerror function can also be used with SWIFTNet FIN to allow it to be called with only a code and description string (instead of code, field reference, option description string). 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 which is written to the translator report as part of the entry for the compliance error you are raising.

Syntax 1

```
cerror(code,$GROUPNAME[0][1][1].#FIELDNAME, "Optional string with error
information can be supplied here");
```

Example 1

An example of this function in this syntax follows:

```
cerror(100,$GROUPNAME[0][1][1].#FIELDNAME);
//This raises compliance error 100 on the FIELDNAME field of the
//specified instance of the GROUPNAME group. There is no optional error text
//given.
```

Syntax 2

```
cerror(code, "String with error information supplied here");
```

Example 2

An example of this function in this syntax follows:

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

collate

The collate function is used to construct a hierarchical relationship between two groups that are not currently related hierarchically but which are instead related by "foreign key" fields. One of the groups is considered the "master" group and the other is considered the "detail" group. Instances of the detail group are matched against the master group and moved to a "result" group which must be a child of the master group. The result group must have an identical field structure to the detail group. The master and detail groups must both be sorted into the same order using the sort function prior to performing the collation. After the collate function is performed, the data in the detail group is no longer available because it has been moved to the result group."

Caution: This is an advanced function which requires understanding of how field data is accessed in translator storage and how indices are used to reference specific instances of groups.

In the command syntax, "ordering" is either the keyword ASC or DESC to indicate that the groups are sorted in ascending or descending order, respectively. If no ordering is given for

---

a particular field, ascending order is assumed. Any number of fields may be specified, but the number of master and detail fields specified must match.

Note: If the collated data is too large, you may receive an out of memory error.

Syntax

Use this syntax:

```
collate (MASTERGROUPNAME[iteration], MASTERFIELDNAME ordering,
MASTERFIELDNAME ordering, …, DETAILGROUPNAME[iteration], DETAILFIELDNAME,
DETAILFIELDNAME, …, RESULTGROUPNAME))
```

Example

An example of this function follows:

```
collate($POHeader, #PONumber, $PODetail, #POHeaderNumber, $PODetailResult);
```

Detailed Example

In this example, you need to process a flat file that contains information for Stores and Sales. All of the Stores information is listed first in the file, in no particular order, and all of the Sales information follows the Stores information, in no particular order. You need to sort the information and associate the Sales information to the appropriate Store information. The Stores and Sales records contain a common field: Stor_id.

Map Layout

The input side of the map contains two repeating groups that are children to the input level: Stores_Group and Sales_Group. The Stores_Group group contains a single occurring record for the Stores records in the input file, and the Sales_Group group contains a single occurring record for the Sales records in the input file.

The Stores_Group is the Master group and the Sales_Group is the Detail Group when you write the Collate extended rule. Since the Stores_Group is the master group, the result group must be contained within this group.

Therefore, the Stores_Group structure contains the single occurring record for the Stores records in the input file and a repeating Result_Group. The Result_Group contains a temporary Sales record that is an exact copy of the Sales_Record in the Sales_Group.

Collate Extended Rule

To sort and collate the data, place the following rule on the Input On-End Extended Rules:

```
SORT($Stores_Group,#Stores_Stor_ID ASC);
SORT($Sales_Group,#Sales_Stor_ID ASC);

COLLATE ($Stores_Group, #Stores_Stor_ID ASC,$Sales_Group, #Sales_Stor_ID ASC,$
Result_Group);
```

Sample Input File

The following is the sample input file:

```
STORES8042   Bookbeat              679 Carson St.       Portland   OR   890762
STORES7066   Barnum's              567 Pasadena Ave.    Tustin     CA   927892
STORES7896   Fricative Bookshop  89 Madison St.         Fremont    CA   900192
SALES7896QQ2299           1993-10-28 00:00:00    15    Net 60     BU78322
SALES8042423LL922         1994-09-14 00:00:00    15    ON invoice MC30212
SALES8042P723             1993-03-11 00:00:00    25    Net 30     BU11112
SALES7066QA7442.3         1994-09-13 00:00:00    75    ON invoice PS20912
SALES7896TQ456            1993-12-12 00:00:00    10    Net 60     MC22222
SALES7896X999             1993-02-21 00:00:00    35    ON invoice BU20752
SALES8042423LL930         1994-09-14 00:00:00    10    ON invoice BU10322
SALES8042QA879.1          1993-05-22 00:00:00    30    Net 30     PC10352
SALES7066A2976            1993-05-24 00:00:00    50    Net 30     PC88882
```

Output File

The following is an example of the output file after translation:

```
STORES7066   Barnum's              567 Pasadena Ave.    Tustin     CA   927892
SALES7066QA7442.3         1994-09-13 00:00:00    75    ON invoice PS20912
SALES7066A2976            1993-05-24 00:00:00    50    Net 30     PC88882
STORES7896   Fricative Bookshop  89 Madison St.         Fremont    CA   900192
SALES7896QQ2299           1993-10-28 00:00:00    15    Net 60     BU78322
SALES7896TQ456            1993-12-12 00:00:00    10    Net 60     MC22222
SALES7896X999             1993-02-21 00:00:00    35    ON invoice BU20752
STORES8042   Bookbeat              679 Carson St.       Portland   OR   890762
SALES8042423LL922         1994-09-14 00:00:00    15    ON invoice MC30212
SALES8042P723             1993-03-11 00:00:00    25    Net 30     BU11112
SALES8042423LL930         1994-09-14 00:00:00    10    ON invoice BU10322
SALES8042QA879.1          1993-05-22 00:00:00    30    Net 30     PC10352
```

concat

The concat function concatenates a specified number of characters from one string onto the end of another string.

Syntax

Use this 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
```

Example

An example of this function follows:

```
string[10] s1,s2;
concat(s1,s2,5);
//Concatenate five characters from string "s2" onto the end of string
//"s1"
```

continue

The continue function continues the execution of the innermost loop without processing the statements in the loop that follow the continue statement.

Example

An example of this function follows:

```
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 "i", 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.

Note: When a count extended rule is performed on an empty group, the value of -1 is returned from count($GROUPNAME[*]).

Example

An example of this function follows:

```
integer i;
i = count($GROUPNAME[*]);
//The [*] is a wildcard that counts the number of iterations of the
//GROUPNAME group.
```

date

The date function converts a string type into a datetime type using a format that you specify. This function enables you to include static characters such as a slash (/), which gives you access to full date support.

Syntax

Use this syntax:

```
Datetime = date("format",string);
where:   datetime=  datetime variable (month specified as 1-12)
         format  =  date format
         string  =  string variable
```

Example

An example of this function follows:

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

Format Specifiers

This table lists the format specifiers:

Format Specifier	Description
%8	ISO-8601 date format. Valid format is *YYYYMMDD'T'HHMMSS'.'sssZ* **Note** This date format cannot be combined with any other format specifier.
%a	Abbreviated weekday name.
%A	Full weekday name.

Format Specifier	Description
%b	Abbreviated month name.
%B	Full month name.
%d	Day of the month as a decimal number (01 - 31).
%H	Hour in 24-hour format (00 - 23).
%I	Hour in 12-hour format (01 - 12).
%j	Day of the year as a decimal number (001 - 366).
%m	Month as a decimal number (01 - 12).
%M	Minute as a decimal number (00 - 59).
%S	Second as a decimal number (00 - 59)
%U	Week of the year as a decimal number, with Sunday as the first day of the week (00 - 51).
%w	Weekday as a decimal number (0 - 6, with Sunday as "0").
%W	Week of the year as a decimal number, with Monday as the first day of the week (00 - 51).
%y	Year without the century as a decimal number (00 - 99).
%Y	Year with the century as a decimal number.
%%	Percent sign.

delete

The delete function deletes a specified occurrence of a group.

Syntax

Use this syntax:

```
delete($GROUP_NAME[N]);
where [n] is the occurrence of the group that you want to delete.
```

Example

An example of this function follows:

```
delete($GROUPNAME[2]);
//Deletes the second occurrence of the GROUPNAME group.
```

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.

Syntax

Use this syntax:

```
empty($GROUP_NAME[index][index][index]. #FIELD_NAME)
```

Example

An example of this function follows:

```
empty($GROUPNAME[0][1][1].#FIELDNAME);
//Set the value of the specified instance of the FIELDNAME field to //null
```

end

The Map Editor 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.

Use begin and end to 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 run 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 use only a single statement, you can omit the begin and end.

Note: Do not end conditions with a semicolon (;). This terminating syntax is necessary for statements *only.*

Syntax

Use this syntax:

```
if condition then
begin
    statement1;
    statement2;
end
```

eof

The eof function explicitly checks whether the input file has reached the end of file condition. This function enables you to verify whether there is more data in the file. The eof function was implemented because the readblock function returns a zero (0) when the translator reaches the end of the input file and also when it encounters new line characters, and when you use the readblock function there is no way to differentiate between these two conditions.

The eof function returns an integer, as defined in the following table:

Integer Returned by eof Function	Definition
1	End of input file reached
0	Input file has more data

Syntax

Use this syntax:

```
eof(0);
//0 = input file, any other argument value results in a translator exception
```

Example

An example of this function follows:

```
integer end_of_input;

end_of_input = eof(0);

if end_of_input = 1 then
  MessageBox("end of input reached",0);
else MessageBox("more input data",0);
```

exist

The exist function returns a nonzero (true) value if there is data in a specified field in internal storage. If there is not data in the specified field, this function returns a zero (false) value. This function is typically used as part of a condition.

Syntax

Use this syntax:

```
if exist($GROUP_NAME[index][index][index]. #FIELD_NAME) then
```

Example

An example of this function follows:

```
if exist($GROUPNAME[0][1][1].#FIELDNAME) then
//Return a non-zero value if the condition is true (data is present in
//the specified instance of the FIELDNAME field). A zero value is
//returned if the condition is false (no data is present in the
//specified instance of the FIELDNAME field).
```

get

The get function enables you to access individual components of a datetime variable. Valid datetime components are:

✦ years

✦ months

✦ days

✦ hours

✦ minutes

✦ seconds

Syntax

Use this 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

An example of this function follows:

```
integer a;
integer b;
datetime d;
a = get days (d);
b = get hours (d);
//Accesses the days from the datetime variable "d" and loads into
//variable "a". Accesses the hours from the datetime variable "d" and
//loads into variable "b".
```

if...then...else

The if, then, and else keywords enable the use of conditional logic. The Map Editor 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. The Map Editor interprets the value as either true or false—a zero value as false, and a nonzero value as true.

The Map Editor evaluates the if/then condition and, if it is true, 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 with if/then to define several blocks of statements, one of which is run. The Map Editor tests the first if/then condition. If the condition is false, The Map Editor tests each sequential condition until it finds one that is true. The Map Editor runs the corresponding block of statements for the true condition. If none of the if/then conditions are true, The Map Editor runs the statements following the else keyword.

The begin function encloses 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 and end. For more information about the begin keyword, see *begin* on page 213. For more information about the end keyword, see *end* on page 225.

Note: Do not end conditions with a semicolon (;). This terminating syntax is necessary for statements *only.*

Example

An example of this function follows:

```
if condition then
begin
     statement1;
     statement2;
end
else
begin
     statement3;
     statement4;
end
```

index

The index function enables you to determine which instance of a loop the translator is currently accessing.

For more information about using indexes in maps, see Appendix D, *Using Indexes in the Map Editor and Translator*.

For more detailed information about using indexes with XML in the Map Editor, see Using Indexes in Common Mapper.doc from the Sterling Commerce Web site (www.sterlingcommerce.com and then select **Customer Support>Support on Demand**).

Note: Support on Demand requires a user name and password, so you must register if you have not completed this step already.

Syntax

Use this syntax:

```
index(integer_variable);
where: integer_variable = integer variable that indicates the
hierarchical level for which you want to
determine the loop count
```
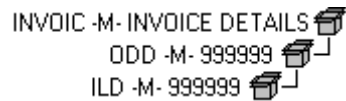
Example

An example of this function follows:

```
index(1);
//This extended rule is located on the ILD group in the following figure.
//This determines the current loop count for the first level (ODD) in
//the hierarchical structure.
```


left

The left function extracts a specified number of characters from the left side of a string variable or field and returns the result as a string.

Syntax

Use this syntax:

```
string_variable = left(string_variable,num_char)
where: num_char = integer variable
```

Example

An example of this function follows:

```
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 numeric function that counts and returns the number of characters in a string. The numeric functions enable you to convert one data type to another.

Syntax

Use this syntax:

```
number_char = len(string);
where: num_char = integer variable
```

Example

An example of this function follows:

```
integer a;
a = len("hello");
// "a" contains the value 5
```

messagebox

The messagebox function enables you to display a text message on the console of the application server. You can view the console output in the application server log file (that is, jboss.log, wl.log, and so forth). This function can serve as a troubleshooting tool. The messagebox function can be used on any field in your map.

Syntax

Use this syntax:

```
messagebox("message",0)
where:  message = message string

messagebox(#fieldname,0);
where: fieldname = name of the field in the map, displays a message with the
string value
```

Note: You must include the zero in the syntax (as previously noted) for this function to work properly.

Example

An example of this function follows:

```
//syntax for a field named TribalFrog

MessageBox("This is field TribalFrog",0);
MessageBox (#TribalFrog,0);

//The output text "This is the field TribalFrog" and the contents of the
//#Tribal Frog field is displayed on the console
//of the application server. You can view the console output in the
//application server log file.
```

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.

Syntax

Use this syntax:

```
string_variable = mid(string_variable,start_pos,num_char)
where: start_pos = integer variable
            num_char = integer variable
```

Example

An example of this function follows:

```
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 character 6.
// "temp_variable" will contain "Shipping"
```

ntoa

The ntoa function is a numeric function that converts real numbers into strings. The numeric functions enable you to convert one data type to another.

Syntax

Use this syntax:

```
ntoa(real,string);
where:  real    = real number variable
        string  = string variable
```

Example

An example of this function follows:

```
real b;
string[20] s;
b = 5.5;
ntoa(b, s);
// "s" contains "5.5"
```

numerrors

The numerrors function returns an integer which contains the current count of errors in the translation report (the count at the time the rule is executed). This function does not require any parameters.

Syntax

Use this syntax:

```
integer error_count;
error_count = numerrors();
```

occurrencetotal

The occurrencetotal function allows an extended rule to check the total occurrences of a particular record thus far in the processing (this is a running total of the number of times the record has occurred at the specific point in time that the occurrencetotal rule is called). This function returns an integer value containing the number of occurrences of the specified record that have been processed.

Note: To reset OccurrenceTotal to zero so you can determine, for example, if a particular block occurred in the current iteration of a group, use the **resetoccurrencetotal** function. See *resetoccurrencetotal* on page 236.

Syntax

```
occurrencetotal(%recordname);
```

Note: The syntax %recordname indicates a record that is referenced by name.

Example

An example of this function follows:

```
If occurrencetotal(%myRec) > 2 then
  //This specifies that if the myRec record has occurred (been processed) more
  //than
  //twice, then the logic that follows this function will be performed by the
  //translator.
```

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.

Readblock, writeblock, and unreadblock are supported only for positional and EDI files.

Note: The readblock function returns a zero (0) 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.

Syntax

Use this syntax:

```
readblock(string_variable);
```

Example

An example of this function follows:

```
while readblock(temp_buffer) do
begin
   if left(tem_buffer,3) = "IEA" then
      begin
            unreadblock();
             break;
        end
   writeblock(temp_buffer);
end
//Read record from input file and place in temp_buffer. Look for
//"IEA" record tag. If found, reset file pointer to where it was
//before the "IEA" record was read. Write contents of temp_buffer to
//output file.
```

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.

For more information about using the corollary Update standard rule to complete the document extraction, see *Setting an Update Standard Rule as Part of Document Extraction* on page 184.

Example

An example of this function follows:

```
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);
//set the value of eofInput to 0 because the end of file has been reached
end
```

resetoccurrencetotal

The resetoccurrencetotal function allows you to reset OccurrenceTotal to zero so you can determine if a particular block occurred in the current iteration of a group, as opposed to learning how many times the translator has encountered a specified block in the course of processing the data on either the input or output side of a map.

Syntax

```
resetoccurrencetotal(%recordname);
```

Note: The syntax %recordname indicates a record that is referenced by name.

Gentran Integration Suite Map Editor Guide

Example

An example of this function follows:

```
resetoccurrencetotal(%myRec);
//This specifies that the Occurrencetotal for the myRec record will be set to
//zero.
```

right

The right function extracts a specified number of characters from the right side of a string variable or field.

Syntax

Use this syntax:

```
string_variable = right(string_variable,num_char)
where: num_char = integer variable
```

Example

An example of this function follows:

```
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 enables information to be retrieved from Gentran Integration Suite. Only the tables and fields available in the Select standard rule are available for the select extended rule.

In the command syntax, "expression" and "receiverlist" can be a string field, string variable, or string literal. 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. For a listing of these table and field names, see *Select and Update Available Options* on page 250.

Syntax

Use this syntax:

```
select fieldname into receiverlist from tablename where key =
expression [and key = expression];
```

Example

An example of this function follows:

```
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:

✦ years

✦ months

✦ days

✦ hours

✦ minutes

✦ seconds

Syntax

Use this syntax:

```
set datetime_component (datetime_variable,integer_variable);
where: datetime_component = individual component of the datetime variable
           datetime_variable      = datetime variable of which you want to
                                                access a component part
              integer_variable         = integer variable
```

Example

An example of this function follows:

```
integer a;
integer b;
datetime d;
set days (d,a);
set hours (d,a);
//Defines the days of the datetime variable "d" from variable "a".
//Defines the hours of the datetime variable "d" from variable "b".
```

sort

The sort function is used to sort the instances of a group in the translator's internal storage by the values of one or more fields in the group.

Caution: This is an advanced function which requires understanding of how field data is accessed in translator storage and how indices are used to reference specific instances of groups.

In the command syntax, "ordering" is either the keyword ASC or DESC to indicate ascending or descending order, respectively. If no ordering is given for a particular field, ascending order is used. Any number of fields may be specified.

Note: If the sorted data is too large, you may receive an out of memory error.

Syntax

Use this syntax:

```
sort (GROUPNAME[iteration], FIELDNAME ordering, FIELDNAME ordering, …)
```

Example

An example of this function follows:

```
sort($POHeader, #PONumber);
sort($PODetail, #POHeaderNumber);
```

strdate

The strdate function converts a datetime type into a string using a format that you specify. This function enables you to include static characters such as a slash (/), which gives you access to full date support.

Syntax

Use this syntax:

```
strdate(datetime,"format",string);
where:
datetime – datetime variable (month specified as 1-12)
format – date format
string – string variable
```

Example

An example of this function follows:

```
datetime d;
string[8] s;

strdate(d,"%y/%m/%d",s);

//Converts a datetime variable into an eight character string in the
//format "year/month/day".
```

Format Specifiers

The following table describes the format specifiers:

Format Specifier	Description
%a	Abbreviated weekday name.
%A	Full weekday name.
%b	Abbreviated month name.
%B	Full month name.
%d	Day of the month as a decimal number (01 - 31).
%H	Hour in 24-hour format (00 - 23).
%I	Hour in 12-hour format (01 - 12).
%j	Day of the year as a decimal number (001 - 366).
%m	Month as a decimal number (01 - 12).
%M	Minute as a decimal number (00 - 59).
%S	Second as a decimal number (00 - 59).
%U	Week of the year as a decimal number, with Sunday as the first day of the week (00 - 51).
%w	Weekday as a decimal number (0 - 6, with Sunday as "0").
%W	Week of the year as a decimal number, with Monday as the first day of the week (00 - 51).
%y	Year without the century as a decimal number (00 - 99).
%Y	Year with the century as a decimal number.
%%	Percent sign.

strstr

The strstr function finds a substring inside a string. This function returns the position of the first instance of the specified substring within the specified string. If strstr does not find the specified substring in the string, it returns a value of -1. This function is zero-based.

Syntax

Use this syntax:

```
integer = strstr("string","substring");
where: integer   = integer variable
          string    = string
          substring = part of the string
```

Example 1

An example of this function follows:

```
integer d;

d = strstr("mississippi","is");

//Finds the first instance of the substring "is" inside the string
//"mississippi" and returns the position of that first substring.
// d = 1 because this function is zero-based
```

Example 2

An example of this function that enables 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," and otherwise do "Y") follows:

```
integer position;
string [1] PONumChar2;
PONumChar2=mid(#PONumber, 2, 1);
position=strstr("0123456789", PONumChar2);

//This function finds a substring within the string, so if the second
//position of purchase order number 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.
```

sum

The sum function is used (along with the sumtotal function) to maintain and validate the sum of a specified numeric field. This function is called with no parameters in the extended rule of the field for which you need to keep a running sum. Optionally, you can add a string parameter—the first character in the string indicates whether the value of the field is treated as a negative number before it is added to the running sum of the field. If the first character

of the string begins with **N** or **n**, the value from the field is treated as a negative and added to the *sumtotal*.

Syntax 1

```
sum();
```

Example 1

An example of this function follows:

```
sum();
//This maintains a running sum for the field on which the extended rule is
//called.
```

Syntax 2

```
sum("nString");
```

Example 2

An example of this function follows:

```
sum("n");
//This adds the value of the field as a negative number to the sumtotal to
//maintains a running sum for the field on which the extended rule is called.
```

sumtotal

The sumtotal function is used (along with the sum function) to maintain and validate the sum of a specified numeric field. This function returns a real numeric value and is used to obtain the current sum of a specified field wherever the value is needed.

For example, if want to verify the sum of a specific field (debitAmount), add the *sum* function to the extended rule of this field. Then, to access this running sum in a later field (debitTotal), use the sumtotal function on the debitTotal field.

Syntax

```
sumtotal(#fieldname);
```

Example

An example of this function follows:

```
sumtotal(debitTotal);
//This returns the current sum of the field debitAmount (on which the sum
//function was used, and returns the current sum of that field into the
//debitTotal field.
```

trim

The trim function enables you to remove white space from both sides of a string by default.
You can also specify a character for the trim function to remove before and after a string.
If you want to remove a character, specify the character in the appropriate case (upper- or
lowercase).

Syntax

Use this syntax:

```
string = trim(string_to_trim) or
string = trim(string_to_trim, character_to_trim)

where: string = string field that contains the end result
       string_to_trim = string field containing the value to be trimmed
       character_to_trim = character to trim from string (no empty spaces)
```

Examples

Use the following syntax to remove white space from both sides of the string:

```
string[25] s;
s = "   my String   ";
$Groupname[1].#Value = trim(s);

//$Groupname[1].#Value now contains the value "my String"
```

Note: This syntax is *not* valid for removing white space:

```
string [25] s;
s = "   my String   ";
$Groupname[1].#Value = trim(s," ");
```

Use the following syntax to remove a leading or trailing character:

```
string [25] s1;
s1 = "zzzStringzzz";
$Groupname[1].#Value = trim(s1,"z");

//$Groupname[1].#Value now contains the value "String"
```

Alternatively, you can also use the following syntax to remove a leading or trailing character:

```
string [25] s1;
string [25] s2;
s1 = "z";
s1 = "zzzStringzzz";
$Groupname[1].#Value = trim(s1,s2);

//$Groupname[1].#Value now contains the value "String"
```

trimleft

The trimleft function enables you to remove white space from the left side of a string by default. You can also specify a character for the trimleft function to remove from the left side of a string. If you want to remove a character, specify the character in the appropriate case (upper- or lowercase).

Syntax

Use this syntax:

```
string = trimleft(string_to_trim) or
string = trimleft(string_to_trim, character_to_trim)

where: string = string field that contains the end result
       string_to_trim = string field containing the value to be trimmed
       character_to_trim = character to trim from string (no empty spaces)
```

Examples

Use the following syntax to remove white space from the left side of the string:

```
string [25] s;
s = "   my String   ";
$Groupname[1].#Value = trimleft(s);

//$Groupname[1].#Value now contains the value "my String   "
```

Use the following syntax to remove a leading character:

```
string [25] s1;
s1 = "zzzString";
$Groupname[1].#Value = trimleft(s1,"z");

//$Groupname[1].#Value now contains the value "String"
```

For similar examples, see *trim* on page 243.

trimright

The trimright function enables you to remove white space from the right side of a string by default. You can also specify a character for the trimright function to remove from the right side of a string. If you want to remove a character, specify the character in the appropriate case (upper- or lowercase).

Syntax

Use this syntax:

```
string = trimright(string_to_trim) or
string = trimright(string_to_trim, character_to_trim)

where: string = string field that contains the end result
       string_to_trim = string field containing the value to be trimmed
       character_to_trim = character to trim from string (no empty spaces)
```

Examples

Use the following syntax to remove white space from the right of the string:

```
string [25] s;
s = "   my String    ";
$Groupname[1].#Value = trimright(s);

//$Groupname[1].#Value now contains the value "   my String"
```

Use the following syntax to remove a trailing character:

```
string [25] s1;
s1 = "Stringzzz";
$Groupname[1].#Value = trimright(s1,"z");

//$Groupname[1].#Value now contains the value "String"
```

For similar examples, see *trim* on page 243.

unreadblock

The unreadblock function provides a method of moving the input file-pointer back one block (a block of data is equivalent to one EDI segment or one positional record). This function unreads the block of data that was just processed by the readblock function.

Note: The unreadblock function works only once and only for the most recent readblock. If you use unreadblock more than once, you will not be able to point to any earlier readblocks.

Unreadblock must only be used in conjunction with the readblock function.

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.

Unreadblock enables 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.

Readblock, writeblock, and unreadblock are supported only for positional and EDI files.

Syntax

Use this syntax:

```
unreadblock();
```

Example

An example of this function follows:

```
while readblock(temp_buffer) do
begin
   if left(tem_buffer,3) = "IEA" then
       begin
             unreadblock();
              break;
          end
   writeblock(temp_buffer);
end
//Read record from input file and place in temp_buffer. Look for
//"IEA" record tag. If found, reset file pointer to where it was
//before the "IEA" record was read. Write contents of temp_buffer to
//output file.
```

update

The update function enables information in Gentran Integration Suite to be updated. This function is similar to the Update standard rule, except that it provides more flexibility. Only

---

Gentran Integration Suite Map Editor Guide

the tables and fields available in the Update standard rule are available for the update extended rule.

The update function also enables you to update process data with a string, instead of using the messagebox function. For information about the messagebox function, see *messagebox* on page 231.

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. For a listing of these table and field names, see *Select and Update Available Options* on page 250.

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.

Syntax

Use this syntax:

```
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.

Example

An example of this function follows:

```
update processdata set xpathresult="hello world" where xpath="example";
```

Syntax for Updating Process Data with a String

Use this syntax:

```
update ProcessData set XPathResult = <some string>
where Xpath = <location in process data>;
```

Example Updating Process Data with a String

An example of this function follows:

```
update ProcessData set XPathResult = #Sender
where XPath = "SenderID";
```

while...do

The while...do function runs a statement repeatedly until the specified termination condition evaluates to zero. Gentran Integration Suite tests the terminating condition before each iteration of the loop, so a while loop runs zero or more times depending on the value of the termination expression.

The begin function encloses 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 and end. See *begin* on page 213 for more information about the begin keyword. See *end* on page 225 for more information about the end keyword.

Note: Do not end conditions with a semicolon (;). This terminating syntax is necessary for statements *only*.

Example

An example of this function follows:

```
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.
```

writeblock

The writeblock function writes the data contained in the argument of a string variable to the output file. The readblock and writeblock functions are used together 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.

Readblock, writeblock, and unreadblock are supported only for positional and EDI files.

Syntax

Use this syntax:

```
writeblock(string_variable);
```

Example

An example of this function follows:

```
while readblock(temp_buffer) do
begin
   if left(tem_buffer,3) = "IEA" then
       begin
            unreadblock();
             break;
         end
   writeblock(tem_buffer);
end
//Read record from input file and place in temp_buffer. Look for
//"IEA" record tag. If found, reset file pointer to where it was
//before the "IEA" record was read. Write contents of temp_buffer to
//output file.
```

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.

For more information about using the corollary Update standard rule to complete the document extraction, see *Setting an Update Standard Rule as Part of Document Extraction* on page 184.

Example

Another example of this function follows:

```
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
```

Select and Update Available Options

This section contains the 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.

For more information, see:

Trading Partner Code List

These are the field names that are available when using the select extended rule on Trading Partner Code Lists:

Note: Refer to this table as CODELIST.

✦ DESCRIPTION
✦ NAME (this is the "list name" of the codelist)
✦ RECEIVERCODE
✦ SENDERCODE
✦ TEXT1
✦ TEXT2
✦ TEXT3
✦ TEXT4

Process Data

These are the field names that are available when using the select or update extended rules on Process Data:

Note: Refer to this table as PROCESSDATA.

✦ XPATH
✦ XPATHRESULT

Correlation

These are the field names that are available when using the update extended rule on Correlations:

Note: Refer to this table as CORRELATIONDATA.

✦ NAME
✦ VALUE

Document Extraction

These are the field names that are available when using the update extended rule with the Document Extraction service:

Note: Refer to this table as DOCUMENTEXTRACTION.

✦ EDISTANDARD
✦ ACCEPTERLOOKUPALIAS
✦ RECEIVERID
✦ SENDERID

Envelope

These are the field names that are available when using the select extended rule on Document Envelopes with the Generic Envelope service:

Note: Refer to this table as ENVELOPE.

✦ PARM
✦ VALUE

Transaction Register

These are the field names that are available when using the select or update extended rules with the Transaction Register:

Note: Refer to this table as TRANSACTIONREGISTER. 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.

✦ FIELD1

- ✦ FIELD2
- ✦ FIELD3
- ✦ FIELD4
- ✦ FIELD5
- ✦ FIELD6

Extended Rules Library

- ✦ *Overview* on page 253
- ✦ *Calling a Rule from an Extended Rule Library in a Map* on page 254
- ✦ *Managing Extended Rule Libraries* on page 255
- ✦ *Importing and Exporting Extended Rule Libraries* on page 260

Overview

This section describes how to use the extended rule library and the properties of the dialog boxes that comprise its functionality. A rules library (used with SWIFTNet and any other data format) contains a list of rules in a separate file outside of the Map Editor source. Map Editor stores the name of the library in its source file, so when you open a map the library is also loaded. Only the library extended rules referenced by a map are compiled into the compiled (.TXO) translation object. This enables you to create a library of extended rules and then add it to any other map, so you do not have to recreate those extended rules after the first time. You can use this functionality with any data format.

> **Note:**   The SWIFT extended rules library (SWIFT_2006.erl), which is automatically installed with Gentran Integration Suite (and checked in), contains all the extended rules necessary to carry out the business logic for SWIFT messages.

The extended rule libraries are used when maps are compiled, not at runtime.

This functionality minimizes the impact to users when, for example, SWIFT updates their messages—without the rule library you would need to update the extended rules for each updated map (correlating to the updated messages), but using the extended rule library you just update the library and then use the library with all the applicable maps.

When you view the checked in libraries through the Extended Rule Library check in interface, you are also able to obtain a list of all the maps that use each library.

The extended rules library can contain many rules. An extended rule consists of a declarations section followed by a statements section. The *declarations* section is required only if you use additional variables. The declarations section is where you declare the

names and types of any variables you use either in the extended rule. The *statements* section is where you define the actions that you want the extended rule to run.

You must declare any variables that are not already defined as part of the input or output specification of the map before you use those variables in an extended rule. For the extended rule libraries, any variables can be passed as parameters. Global variables can be referenced within a library function if the function has "Text Substitution" selected. However, we recommend that you do not use "Text Substitution" then you do not have a reusable library of functions.

Rule libraries are versioned resources. When you create a new rule library you need to check it in to Gentran Integration Suite just like you need to check in maps. This also enables you to check out, version, and delete extended rule libraries. Furthermore, when you view the checked in libraries through the Extended Rule Library check in interface, you can also see all the maps that use each library. This is very important because it enables you to easily view a list of the maps that will need to be recompiled if you change an extended rule in a library (you would recompile all the maps that use that particular library). See *Managing Extended Rule Libraries* on page 255.

Additionally, you can import and export extended rule libraries into Gentran Integration Suite using the Resource Manager. See *Importing and Exporting Extended Rule Libraries* on page 260.

You can call an extended rule from a library in any extended rule in a map.

The extended rule library functionality consists of the following dialog boxes:

✦ Rule Library Manager
✦ Rule Library
✦ Library Rule
✦ Add Parameter

Calling a Rule from an Extended Rule Library in a Map

You can call a rule from any extended rule library (that is currently checked in to Gentran Integration Suite) in any extended rule in your map. The syntax you use to call a rule from a library is:

```
call library_name.rule_name parameter
```

In this syntax, **library_name** is the name of the extended rule library. For example, if the library is SWIFT_2006.erl, the library_name is **SWIFT**. And the **rule_name** is the name of the rule that you defined in the Library Rule dialog box.

Note: You can have multiple rule libraries with the same name and different version numbers, but you can only use one rule library of the same name in a map (the last version of that rule library that was checked in to the system).

Managing Extended Rule Libraries

Overview

When you create a new rule library you need to check it in to Gentran Integration Suite just like you need to check in maps. You can also check out, version, and delete extended rule libraries.

When you view the checked in libraries through the Extended Rule Library check in interface, you are able to obtain a list of all the maps that use each library. This is very important because it enables you to easily view a list of the maps that will need to be recompiled if you change an extended rule in a library (you would recompile all the maps that use that particular library).

Checking In Extended Rule Libraries

To use extended rule libraries (.erl files) in Gentran Integration Suite, you must first check them in from your client computer to Gentran Integration Suite.

If you want to edit a extended rule library, you can check out an extended rule library and Gentran Integration Suite displays the .erl library version, which you can edit in the Map Editor (in the Rule Library Manager).

To check a extended rule library in to Gentran Integration Suite:

1. From the Gentran Integration Suite **Deployment** menu, select **Extended Rule Libraries**.

2. In the Check-in section, click **Go!**

3. To check in an extended rule library (.erl), either type the path to the extended rule library in the **External Rule Library filename** box or click **Browse**, locate the extended rule library on your local disk, and click **Open**. The extended rule library name must not have spaces or apostrophes in it.

4. Click **Next**.

5. In the **Check-in Comments** field, type the appropriate comments and click **Next**. This field is required. Use the Check-in Comments field to note the purpose of the extended rule library or explain the changes made to it.

6. Review the settings for the extended rule library you are checking in and click **Finish** to apply your changes.

Searching for Extended Rule Libraries

To check in a extended rule library version or check out, enable, or disable a extended rule library, you must first specify the appropriate extended rule library. You can locate a specific extended rule library in three ways:

✦ Search for a extended rule library by name.

✦ Select a extended rule library from an alphabetical list.

✦ Select a extended rule library from a list according to extended rule library type.

Searching for a extended rule library by name is more precise and provides fewer results. Searching from an alphabetical list will result in a list of all extended rule libraries or all extended rule libraries beginning with a specified letter or digit.

After you have found the appropriate extended rule library, you can manage it through the Source Manager and the Version Manager.

Searching for a Extended Rule Library by Name

To search for a extended rule library by name:

1. From the Gentran Integration Suite **Deployment** menu, select **Extended Rule Libraries**.

    Note: Field names that display in orange font are required fields. You must complete these fields before proceeding.

2. In the Search section, type the name of the extended rule library. Case does not matter and you can type part of a name.

    Gentran Integration Suite returns a list of matches unless no extended rule libraries meet the criteria you specified.

Searching for a Extended Rule Library from a List

To select a extended rule library from a list:

1. From the Gentran Integration Suite **Deployment** menu, select **Extended Rule Libraries**.

2. In the List section, select **All** or a specific letter or digit (0 - 9) and click **Go!**

    Gentran Integration Suite returns a list of matches unless no extended rule libraries meet your criteria.

About Search Results

When you search for a extended rule library, Gentran Integration Suite returns a results page. The results are displayed in a three-column table. Each row contains icons for the Source Manager and the Version Manager, the extended rule library name, and the extended rule library type. You can sort the list alphabetically by extended rule library name or extended rule library type.

Source Manager

The Source Manager enables you to check out a extended rule library and check in a new version of a extended rule library. It also displays the following information about a extended rule library:

✦ Extended Rule Library name

✦ Extended Rule Library type

✦ Date the extended rule library was checked in

✦ User name of the person who checked in the extended rule library

✦ Comments about changes that have been made to the extended rule library

Version Manager

The Version Manager enables you to enable or disable a version of a extended rule library. If there are two or more versions of the extended rule library, you can select a default extended rule library.

The Version Manager also displays the following information about a extended rule library:

✦ Extended Rule Library name

✦ Extended Rule Library type

✦ Date the extended rule library was checked in

✦ User name of the person who checked in the extended rule library

✦ Comments about changes that have been made to the extended rule library

✦ Whether the extended rule library is enabled

Checking In Versions of Extended Rule Libraries

If you update an extended rule library (or any extended rule in that library) in the Extended Rule Library Editor, you must check in the new version to make it available to Gentran Integration Suite and to release the lock on the extended rule library.

Additionally, you must recompile any maps that use that library. You can view all the maps that use each library through the Delete Libraries interface. This is very important because it enables you to easily view a list of the maps that will need to be recompiled if you change an extended rule in a library (you would recompile all the maps that use that particular library). You can also view the list of maps that use a specific rule library by clicking on the rule name within the search or list view.

Before you can check in a version of a extended rule library, you must first check out the extended rule library from Gentran Integration Suite and you must select the appropriate check box (description follows) to maintain the lock on the extended rule library in Lock Manager.

Note: Extended Rule Libraries may also be unlocked using the Lock Manager.

To check in a new version of a extended rule library to Gentran Integration Suite:

1. From the Gentran Integration Suite **Deployment** menu, select **Extended Rule Libraries**.

2. Find the extended rule library for which you want to check in a new version.

3. Next to the extended rule library for which you want to check in a new version, click **source manager**.

4. Next to **Check in a new version of this external rule library**, click **Go!**

5. To check in an extended rule library (.erl), either type the path to the extended rule library or click **Browse**, locate the extended rule library on your local disk, and click **Open**.

6. Type comments in the Check-in comments field and click **Next**. Use the Check-in comments field to note the purpose of the extended rule library or explain the changes made to it. This field is required.

7. Verify that the **Selected Version** is the version you want to be the default version and click **Next**.

8. Verify that the **Release the lock on the file** check box is selected to release the lock on the extended rule library.

9. Review the settings for the extended rule library you are checking in and click **Finish** to apply your changes. Gentran Integration Suite displays the message, *Update completed successfully.*

Checking Out Extended Rule Libraries

To modify a extended rule library in the Extended Rule Library Editor, you must check out a copy of the extended rule library from Gentran Integration Suite. When you check out a extended rule library, Gentran Integration Suite locks the extended rule library so that no one else can modify the extended rule library while you are editing it.

Note: You can also check out a read-only copy of a extended rule library, which does not lock the extended rule library in Gentran Integration Suite.

To check out a copy of a extended rule library from Gentran Integration Suite:

1. From the Gentran Integration Suite **Deployment** menu, select **Extended Rule Libraries**.

2. Find the extended rule library you want to check out.

3. Next to the extended rule library you want to check out, click **source manager**.

4. Next to the version you want to check out, click **check-out**.

5. In the message box that opens, click **OK** to lock the extended rule library or click **Cancel** to check out a read-only copy of the extended rule library (this does not lock the extended rule library).

6. To access the extended rule library immediately, select **Open** and click **OK**. Gentran Integration Suite opens the file in the Extended Rule Library Editor. Save the extended rule library in the Extended Rule Library Editor. This completes check out.

    Note: The first time you select Open, Gentran Integration Suite prompts you to select an application. Select **Extended Rule Library** or click **Other** and locate Extended Rule Libraryper.exe. Click **OK**.

7. To use the extended rule library later, select **Save** then click **OK**. Gentran Integration Suite prompts you to select a destination location. Browse to the location and click **OK** to save the file and complete check out.

Specifying Default Extended Rule Libraries

Since extended rule libraries are only used at compile-time to create translation (.txo) objects, setting a default version of a rule library has no effect on existing translation objects that have been checked into Gentran Integration Suite. The default version of a rule library is not used by Gentran Integration Suite when performing translations. Instead, specifying a default serves as a way for you to denote which version is the "correct" version of the library (the version of the extended rule library that should be used when compiling maps that use that particular library).

To specify a default extended rule library:

1. From the Gentran Integration Suite **Deployment** menu, select **Extended Rule Libraries**.

2. Find the extended rule library you want and click **version manager**.

3. Select the version you want to make the default and click **Save**.

    Gentran Integration Suite displays the message, *Extended Rule Library status has been successfully updated*.

Viewing a List of Maps that Use an Extended Rule Library

When you view the checked in libraries through the Extended Rule Library check in interface, you are able to obtain a list of all the maps that use each library. This is very important because it enables you to easily view a list of the maps that will need to be recompiled if you change an extended rule in a library (you would recompile all the maps that use that particular library). You can also view the list of maps that use a specific rule library by clicking on the rule name within the search or list view.

To view the list of maps that use an extended rule library:

1. From the Gentran Integration Suite **Deployment** menu, select **Extended Rule Libraries**.

2. Find the extended rule library you want to check out.

3. Next to the extended rule library you want to check out, click **source manager**.

4. Next to the library for which want to view the list of maps that use it, select the **Delete** check box.

   Note: You will not actually complete the deletion procedure.

5. Next to **Delete Selected Versions,** click **Go!** and you are presented with a message box asking if you are sure you want to delete the selected version.

6. Click **OK** and you are presented with the Resource Summary page to confirm the deletion. In the **Maps Referencing This Rule Library** section, note the list of maps that reference the library.

7. Click **Cancel** to cancel the deletion request.

Deleting an Extended Rule Library

To delete an extended rule library:

1. From the Gentran Integration Suite **Deployment** menu, select **Extended Rule Libraries**.

2. Find the extended rule library you want to check out.

3. Next to the extended rule library you want to check out, click **source manager**.

4. Next to the library for which want to view the list of maps that use it, select the **Delete** check box.

5. Next to **Delete Selected Versions,** click **Go!** and you are presented with a message box asking if you are sure you want to delete the selected version.

6. Click **OK** and you are presented with the Resource Summary page to confirm the deletion.

7. Click Next.

8. Click Delete to process the deletion request.

   Caution: Deleting this resource may cause processes to fail—your delete action cannot be reversed.

Importing and Exporting Extended Rule Libraries

Overview

The Gentran Integration Suite Import/Export feature now enables you to import and export extended rule libraries along with the other supported resource types. This means that you can configure an extended rule library on one system and then move or copy it to a different system, thereby avoiding having to recreate the extended rule library on each system. Even if you have libraries that are going to be slightly different from one system to another, you can export the libraries from one system and import them to a different system, and then make the necessary changes to the libraries on the second system.

Exporting Extended Rule Libraries

Before export, resources must be converted into the proper format for storage and transference. The Export option converts a group of resources that you specify into one of the following formats:

✦ XML – Enables you to transfer data or resources between two existing systems.

✦ Installable bundle – Optionally enables you to load during an Gentran Integration Suite install on a new system.

In the export process, after you have defined the file format to use for the export, you select the version of the resources to export:

✦ The Standard export copies non-versioned extended rule libraries and the default version of versioned resources.

✦ The Advanced export copies non-versioned extended rule libraries and enables you to choose to export just default versions or all versions of versioned extended rule libraries.

You can export extended rule libraries that are associated with a particular resource tag or resources not associated with a resource tag. For instructions, see the appropriate topic:

✦ *Exporting Extended Rule Libraries Using a Resource Tag* on page 261

✦ *Exporting Extended Rule Libraries Without a Resource Tag* on page 262

Exporting Extended Rule Libraries Using a Resource Tag

To export extended rule libraries associated with a resource tag:

1. From the **Deployment** menu, select **Resource Manager** > **Import/Export**.

2. On the Import/Export Resources page, next to **Export Resources**, click **Go!**

3. On the Output Format Type page, indicate the type of format to export to, **XML Document** or **Install Bundle**, and then click **Next**.

4. On the Resource Group page, select **Yes** to indicate that you want to export resources according to a tag name and click **Next**.

5. From the list below your selection, select the previously created resource tag, and then click **Next**.

6. Indicate whether to export private certificates that are associated with the resource tag.

7. Click **Finish** to export the resource tag, including any private certificates, and create the export file.

8. When the message, *The system update completed successfully* displays, click **View Export Report** to see the export report.

9. Click **Download Export data (.xml or .jar)** to download the export file to a hard drive or disk.

Note: The extension shown in the parentheses depends on the type of file you chose to export to, either an XML document (.xml) or an installable bundle (.jar).

10. In the **Save As** dialog box, select the location to which to save the file, and click **Save**.

11. In the **File Download** dialog box, click **Save**.

Exporting Extended Rule Libraries Without a Resource Tag

To export resources without a resource tag:

1. From the **Deployment** menu, select **Resource Manager** > **Import/Export**.

2. On the Import/Export Resources page, next to **Export Resources**, click **Go!**

3. On the Output Format Type page, indicate the type of format to export to, **XML Document** or **Install Bundle**, and then click **Next**.

4. On the Resource Group page, accept the default **No** to indicate that you do not want to export resources according to a tag name and click **Next**.

5. On the Export Type page, indicate the type of export to perform for resources versions and click **Next**. Options are:

    ◆   Standard – Export the default version of the resource.

    ◆   Advanced – Export the default or all versions of the resource.

6. On the Select Resources page, indicate Extended Rule Libraries as the resource you want to export, and click **Next**.

7. On the page that reflects the resource to export, select the extended rule library or libraries that you want to export from the **Available** box, click the arrow to move the resources to the **To be exported** box, and repeat for all libraries that you want to export.

8. Complete one of the following steps:

    ◆   If you are exporting only the default version of the library, click **Next**.

    ◆   If you chose to use the **Advanced** option on the Export Type page and if you are exporting all versions of the library, select **Export All Versions** (located below the **Available** box) and click **Next**.

9. If you selected more than one resource type in step 6., the next Resource Type page opens. Repeat steps 7. and 8. for each page the reflects the resources you select.

10. Click **Finish** to export the library and create the export file.

11. When the message, *The system update completed successfully* displays, click **View Export Report** to see the export report.

12. Click **Download Export data (.xml or .jar)** to download the export file to a hard drive or disk.

Note: The extension shown in the parentheses depends on the type of file you chose to export to, either an XML document (.xml) or an installable bundle (.jar).

---

13. In the **File Download** dialog box, click **Save**.

14. In the **Save As** dialog box, select the location to which to save the file, and click **Save**.

Importing Extended Rule Libraries

When you import extended rule libraries, the Import option converts an XML file or installable bundle to Gentran Integration Suite resources format.

Depending on the type of export you used (standard or advanced, default or all versions), the Import option performs the following functions:

✦ Creates new non-versioned extended rule libraries

✦ Creates and checks in new versioned extended rule libraries (assigns time/date of the import)

✦ Updates or preserves existing non-versioned extended rule libraries

✦ Preserves or appends existing checked-in extended rule libraries

---

Caution: To prevent the loss or corruption of existing records, preserve and download the backup file of all imports that you conduct.

The standard import:

✦ Replaces non-versioned records

✦ Appends to existing versioned resources

✦ Sets the default according to imported records

Some important general information about importing resources:

✦ You must manually update imported resources that contain hard-coded, computer-specific information. For example, system paths in scripts must be manually changed to use the new path after importing resources to their new location.

✦ During import, Gentran Integration Suite creates a backup file containing records as they existed prior to import. After you import the resources, you can download and preserve the backup file, which is named backup.xml by default.You can change the name of the backup file so that you do not overwrite an existing backup file.

For both non-versioned and versioned resources, the import process creates a new record where none exists. If you are importing an extended rule library to an environment that already contains the same library, you have the option to update the existing library (selecting Yes for the **Some objects being imported may exist in the system. Do you wish to update them?** parameter) or to preserve the existing library (selecting No for the

Some objects being imported may exist in the system. Do you wish to update them?
parameter).

Note: Each option handles versioned and non-versioned resources differently.

For the update option, selecting Yes for the **Some objects being imported may exist in the system. Do you wish to update them?** parameter is the default option. Depending on whether you are working with versioned or non-versioned resources, different actions occur.

✦ If you are importing non-versioned extended rule libraries, the update option replaces the existing records.

✦ If you are importing versioned libraries, the update option appends to the list when versions already exist, incrementing by one, according to the starting point of the import system.

For the preserve existing option (selecting No for the **Some objects being imported may exist in the system. Do you wish to update them?** parameter) depending on whether you are working with versioned or non-versioned resources, different actions occur.

✦ Non-versioned resources – If a record (file containing the extended rule library or libraries) already exists, nothing is imported. The import process does not change existing records.

✦ For versioned resources – The preserve existing option does not change any versions currently in the system, and adds only new versions.

Importing an Extended Rule Library

To import a file containing an extended rule library or libraries:

1. From the **Deployment** menu, select **Resource Manager** > **Import/Export**.

2. On the Import/Export page, next to **Import Resources**, click **Go!**

3. Type the name of the file containing the extended rule library to import, or use **Browse** to locate and select it by clicking **Open**, and click **Next**. If the import file contains errors, you can either select the file and click **Next** again to continue the import, or click **Cancel** to stop the import process.

4. If the exported file used a resource tag, type a name and description for the new resource tag and click **Next**.

5. Indicate whether you want to update objects that may already exist in Gentran Integration Suite with objects from the import (the default is Yes), and click **Next**.

6. On the Extended Rule Libraries screen, from the Available list, select each library that you want to import, move the resources to the **To be imported list**, and click **Save** when you have selected all the necessary libraries. If you select more than one resource type to import, you must repeat this step for each resource type.

Note: Some resources objects may require you to type the passphrase (created during export) before completing the import.

Gentran Integration Suite Map Editor Guide

7. To update Gentran Integration Suite with the imported file or files, click **Finish**.

Note: This process may take several minutes, depending on the number and size of the libraries being imported.

---

8. When the message, *The system update completed successfully* displays, click **View Import Report** to see the import report.

9. If errors occurred during import, next to **Data which failed to Import**, click **Download** to view the XML file containing the names of the objects that did not import.

Note: This option displays only when the import completed with errors.

---

10. To create a backup of the data as it existed prior to import, next to **Data in SI before Import**, click **Download**.

11. In the **Save As** dialog box, change the default filename, backup.xml, to a unique, meaningful name for this backup, and then click **Save**.

Caution: Completing this step is very important. If you do not download the backup file to a hard drive or disk, you will lose the backup for the data as it existed prior to import.

Using User Exits

A *user exit* is an extended rule that enables a map to temporarily exit translation and perform functions that Gentran Integration Suite does not perform during typical translation.

This section covers the following topics:

✦ About User Exits

✦ Creating a User Exit

✦ Examples of User Exit Code

About User Exits

User exits perform functions that Gentran Integration Suite does not perform during typical translation. For example, user exits can:

✦ Access a database table to perform cross-references or lookups.

✦ Perform complex pricing calculations that enable for multiple customers, where the seller is located, or where the product is sold.

In Gentran Integration Suite, user exits are extended rules that enable Java™ objects to be created and accessed. With user exits you can create custom Java classes or use third-party Java classes. User exits can create objects, access properties, or start methods.

The Map Editor does not inspect Java classes to verify the correctness of method and property names during map compilation.

Caution: Extended rules cannot compare property or method results because property and method types are unknown prior to compilation. Also, extended rules cannot catch Java exceptions and Java methods called by extended rules cannot throw exceptions.

To handle an exception, you must create a user exit containing a wrapper class that calls the real class and handles the exception. Also, the Java class needs a constructor with a string argument. For more information on using constructor with a string argument, see *Creating an Object* on page 270.

For more information, see:

✦   *Long Keyword* on page 268
✦   *Using Data Types* on page 268
✦   *Location Within a Map* on page 269

Long Keyword

The keyword "long" is supported after a parameter is passed to a user exit method. This keyword is used for integers *only* and indicates a 64-bit integer.

This function is backward-compatible; the translator assumes that user exits used with versions of the Map Editor prior to the implementation of this functionality refer to 32-bit integers.

Using Data Types

The following table lists the extended rule data types and the corresponding Java data types for user exits:

Extended Rule Data Type	Java Data Type
Integer	Int
Real	Double
Datetime	java.util.Date
String	java.lang.String
Object	java.lang.Object

The Translation service automatically converts the extended rule data type to the Java data type and vice versa. As a result, even single-character strings are converted to a String object in java because the "char" datatype is not supported. If the conversion is not

completed, the extended rule is immediately terminated and the Translation service writes an error to the translator report.

Objects

Java objects use the data type *object*. Objects provide properties (data values that can be retrieved or updated) and methods (functions that can be invoked to return a result).

The object must be declared and then created using the *new* command, passing the full class name of the object and any constructor parameters.

Invoke an object's methods using the syntax:

```
object.method(parameters)
```

Invoke an object's properties using the syntax:

```
object.property
```

Location Within a Map

The location of the user exit in the map depends on what the user exit must do and what you want the scope of the rule to be. You can declare, create, and use an object in an extended rule for a map component at any hierarchical level, including field-level extended rules, if appropriate. Typically, you might:

✦ Declare and create an object in the pre-session rule.

✦ Use the object in other rules throughout the map.

For more information about creating extended rules for map components, see Chapter 9, *Using Extended Rules*.

Creating a User Exit

User exits are an advanced feature of Gentran Integration Suite. To use them, you must know or be familiar with:

✦ Java programming

✦ Translator Programming Language (TPL) constructs for creating and manipulating Java objects

✦ Full name of the Java classes to be used

✦ How and when extended rules are invoked and their scope

Before you run a user exit, you must install the Java class or classes so that Gentran Integration Suite can locate them. First, you must package the Java classes into one or more .jar files. Then, you can install the .jar files in Gentran Integration Suite using the install3rdParty.sh script.

To create a user exit:

1. Start the Map Editor.

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

3. Determine where the user exit will be located in the map (for example, session rules and other extended rules).

> **Note:** When you apply a user exit to a map component, subordinate map components may also be able to run the same user exit.

4. Construct the user exit. See the following examples.

For information about extended rules, see Chapter 9, *Using Extended Rules*.

Examples of User Exit Code

The following examples illustrate the steps of constructing a user exit.

Defining an Object Variable

This sample code defines a variable that represents a Java object:

```
object ob;
```

Creating an Object

This sample code creates an instance of a Java object:

```
object ob;
ob = new ("somepackage.SomeClass", "example");
//Creates an instance of the somePackage.SomeClass class, passing
//"example" to the constructor.
```

This is a Java example of the constructor that the code creating an instance of a Java object (above) will call using reflection:

```java
package somepackage;


public class SomeClass {
 public String someProperty; //this is the string property used in the user
exit example

  /**
   * This is the constructor, that the user exit example
   * ob = new ("somepackage.SomeClass", "example");
   * will call using reflection. The line is calling a SomeClass
   * constructor in the somepackage and giving it a string argument
   * therefore there the below constructor must be declared to
   * match that signature
   *
   * @param ex
   */
 public SomeClass(String ex)
 {
  //this is the constructor needed
  //must catch all exceptions
 }

  /**
   * This is the method that matches the user exit example
   * result = ob.someMethod("parm1", 2);
   * will call using reflection. The line is calling someMethod
   * passing a String parameter and an integer parameter, and
   * expects an integer returned.
   *
   * @param parm1
   * @param numparm
   * @return
   */
 public int someMethod(String parm1, int numparm)
 {
  //this is the method needed for the user exit example
  //must catch all exceptions
  return 0;
 }

}
```

Getting a Property Value

This sample code obtains a property value from a Java object:

```
object ob;
string[50] prop;
ob = new ("somepackage.SomeClass", "example");
value = ob.someProperty;
//Get the value from the someProperty property of ob and store it
//in prop (assumes someProperty is a String)
```

Setting a Property Value

This sample code sets a property value in a Java object:

```
object ob;
ob = new ("somepackage.SomeClass", "example");
ob.someProperty = "hello";
//Sets the value of the someProperty property of ob to the string
//"hello"
```

Calling a Method

To call a method, use the syntax *objectname.methodname(parameters)*.

This sample code calls a method on a Java object and stores the result in a variable:

```
object ob;
integer result;
ob = new ("somepackage.SomeClass", "example");
result = ob.someMethod("parm1", 2);
//Calls the someMethod method of ob, passing the string "parm1" and the
//integer 2 as the method parameters. Stores the result of the method in
//result (assumes the method returns an int).
```

Chapter 12

Managing Maps

This section provides general information about managing maps in Gentran Integration Suite.

Note: Fields with colored field names are required. If you skip a required field, a message prompts you to supply the missing information.

This section covers the following topics:

- ✦ Checking In Maps
- ✦ Searching for Maps
- ✦ About Search Results
- ✦ Checking In Versions of Maps
- ✦ Checking Out Maps
- ✦ Enabling or Disabling Translation Objects and XML Encoder Objects
- ✦ Specifying Default Maps
- ✦ Importing and Exporting Maps

Checking In Maps

To use maps in Gentran Integration Suite, you must first check them in from your client computer to Gentran Integration Suite. You can check in a map (.mxl or .map) alone. If you check in a translation object (.txo) or a compiled XML encoder object (.ltx), you must also check in the corresponding source map.

Source maps are available in Gentran Integration Suite for research purposes. Also if you want to edit a map, you can check out any type of map and Gentran Integration Suite displays the .mxl or .map version, which you can edit in the Map Editor. Translation objects (.txo) and XML encoder objects (.ltx) can be used by the Gentran Integration Suite Translation service after the objects are checked in and activated.

To check a map in to Gentran Integration Suite:

1. From the Gentran Integration Suite **Deployment** menu, select **Maps**.

2. In the Check-in section, click **Go!**

3. To check in a source map (.mxl or .map), either type the path to the map in the **Map filename** box or click **Browse**, locate the map on your local disk, and click **Open**. The map name must not have spaces or apostrophes in it.

4. To check in a translation object (.txo) or XML encoder object (.ltx), either type the path to the map in the **Compiled Map filename** box or click **Browse**, locate the map on your local disk, and click **Open**. The map name must not have spaces or apostrophes in it.

5. In the **Check-in Comments** field, type the appropriate comments and click **Next**. This field is required.

    Tip: Use the Check-in Comments field to note the purpose of the map or explain the changes made to it.

    ---

6. If you are checking in a translation object (.txo) or XML encoder object (.ltx) and you do not want the map to be enabled, clear the **Enable for Business Processes** check box.

    For more information about enabling and disabling maps, see *Enabling or Disabling Translation Objects and XML Encoder Objects* on page 278.

7. Review the settings for the map you are checking in and click **Finish** to apply your changes.

Searching for Maps

To check in a map version or check out, enable, or disable a map, you must first specify the appropriate map. You can locate a specific map in three ways:

✦ Search for a map by name.

✦ Select a map from an alphabetical list.

✦ Select a map from a list according to map type.

Searching for a map by name is more precise and provides fewer results. Searching from an alphabetical list will result in a list of all maps or all maps beginning with a specified letter or digit. Selecting a map from a list according to map type provides a list limited to one type of map (Uncompiled Draft, .mxl or .map; XML Encoder, .ltx; Translation, .txo).

After you have found the appropriate map, you can manage it through the Source Manager and the Version Manager.

Searching for a Map by Name

To search for a map by name:

1. From the Gentran Integration Suite **Deployment** menu, select **Maps**.

> **Note:** Field names that display in orange font are required fields. You must complete these fields before proceeding.

2. In the Search section, type the name of the map. Case does not matter and you can type part of a name.

Gentran Integration Suite returns a list of matches unless no maps meet the criteria you specified.

Searching for a Map from a List

To select a map from a list:

1. From the Gentran Integration Suite **Deployment** menu, select **Maps**.

2. In the List section, select one of the following:

   ◆ Alphabetically – Select **All** or a specific letter or digit (0 - 9) and click **Go!**

   ◆ Map Type – Select **Translation Object**, **XML Encoder Object, Source Maps XML,** or **Source Maps only** and click **Go!**

Gentran Integration Suite returns a list of matches unless no maps meet your criteria.

About Search Results

When you search for a map, Gentran Integration Suite returns a results page. The results are displayed in a three-column table. Each row contains icons for the Source Manager and the Version Manager, the map name, and the map type. You can sort the list alphabetically by map name or map type.

Source Manager

The Source Manager enables you to check out a map and check in a new version of a map. It also displays the following information about a map:

✦   Map name

✦   Map type

✦   Date the map was checked in

✦   User name of the person who checked in the map

✦   Comments about changes that have been made to the map

Version Manager

The Version Manager enables you to enable or disable a version of a map. If there are two or more versions of the map, you can select a default map.

The Version Manager also displays the following information about a map:

✦   Map name

✦   Map type

✦   Date the map was checked in

✦   User name of the person who checked in the map

✦   Comments about changes that have been made to the map

✦   Whether the map is enabled

Checking In Versions of Maps

If you update a map in the Map Editor, you must check in the new version to make it available to Gentran Integration Suite and to release the lock on the map.

Before you can check in a version of a map, you must first check out the map from Gentran Integration Suite and you must select the appropriate check box (description follows) to maintain the lock on the map in Lock Manager.

Note: Maps may also be unlocked using the Lock Manager.

To check in a new version of a map to Gentran Integration Suite:

1. From the Gentran Integration Suite **Deployment** menu, select **Maps**.

2. Find the map for which you want to check in a new version.

3. Next to the map for which you want to check in a new version, click **source manager**.

4. Next to **Check in a new version of this map**, click **Go!**

5. To check in a source map (.mxl or .map), either type the path to the map or click **Browse**, locate the map on your local disk, and click **Open**.

6. To check in a translation object (.txo) or XML encoder object (.ltx), either type the path to the map or click **Browse**, locate the map on your local disk, and click **Open**.

7. Type comments in the Check-in comments field and click **Next**. Use the Check-in comments field to note the purpose of the map or explain the changes made to it. This field is required.

8. Verify that the **Selected Version** is the version you want to be the default version and click **Next**.

9. If you want the map to be available for business processes to use, verify that the **Enable for Business Processes** check box is selected. If you do not want the map to be available, clear the check box.

10. Verify that the **Release the lock on the file** check box is selected to release the lock on the map.

11. Review the settings for the map you are checking in and click **Finish** to apply your changes. Gentran Integration Suite displays the message, *Update completed successfully.*

Checking Out Maps

To modify a map in the Map Editor, you must check out a copy of the map from Gentran Integration Suite. When you check out a map, Gentran Integration Suite locks the map so that no one else can modify the map while you are editing it.

Note: You can also check out a read-only copy of a map, which does not lock the map in Gentran Integration Suite.

You can check out only a source map (.mxl or .map). So, for example, you select a translation object to check out, you actually check out the corresponding source map for that translation object.

To check out a copy of a map from Gentran Integration Suite:

1. From the Gentran Integration Suite **Deployment** menu, select **Maps**.

2. Find the map you want to check out.

3. Next to the map you want to check out, click **source manager**.

4. Next to the version you want to check out, click **check-out**.

5. In the message box that opens, click **OK** to lock the map or click **Cancel** to check out a read-only copy of the map (this does not lock the map).

6. To access the map immediately, select **Open** and click **OK**. Gentran Integration Suite opens the file in the Map Editor. Save the map in the Map Editor. This completes check out.

Note:	The first time you select Open, Gentran Integration Suite prompts you to select an application. Select **Mapper** or click **Other** and locate Mapper.exe. Click **OK**.

7. To use the map later, select **Save** then click **OK**. Gentran Integration Suite prompts you to select a destination location. Browse to the location and click **OK** to save the file and complete check out.

Enabling or Disabling Translation Objects and XML Encoder Objects

Enabling a translation object (.txo) or XML encoder object (.ltx) makes it available to the Gentran Integration Suite Translation service. Source maps (.mxl or .map) cannot be enabled or disabled.

You can enable or disable a map in two ways:

✦   At the time you check in a map

✦   Through the Version Manager after the map has been checked in

To enable or disable a map that has been checked in (to check a map in, see *Checking In Maps* on page 274):

1. From the Gentran Integration Suite **Deployment** menu, select **Maps**.

2. Find the map you want to enable or disable and click **version manager**.

3. To enable a map, select the **Enable** check box.

    A check mark indicates the map is enabled.

4. To disable a map, clear the **Enable** check box.

   An empty check box indicates the map is unavailable.

5. Click **Save**.

   Gentran Integration Suite displays the message, *Map status has been successfully updated*.

Specifying Default Maps

The default map is the version of the map that is available at the start of a business process. You can specify a default map only for translation objects and XML encoder objects and only if you have two or more versions of the map. One version must be selected as the default.

To specify a default map:

1. From the Gentran Integration Suite **Deployment** menu, select **Maps**.

2. Find the map you want and click **version manager**.

3. Select the version you want to make the default and click **Save**.

   Gentran Integration Suite displays the message, *Map status has been successfully updated*.

Importing and Exporting Maps

The Gentran Integration Suite Import/Export feature enables you to save time and increase the accuracy of duplicating resources on different systems. Rather than just copying map (.mxl or .map) files from one computer to another, this feature enables you to move resources and data between Gentran Integration Suite environments of the same version. The Import/Export feature enables you to:

✦ Move from a test Gentran Integration Suite environment to a production Gentran Integration Suite environment.

✦ Move resources from one Gentran Integration Suite system to another.

The ability to import and export maps means that you can configure resources on one system and then move or copy them to a different system, thereby avoiding having to recreate the resources on each system. Even if you have resources that are going to be slightly different from one system to another, you can export the resources from one system and import them to a different system, and then make the necessary changes to the resource on the second system.

The Import/Export feature supports several different resource types, including maps.

Importing Large XML Files

You import XML files, including maps, into Gentran Integration Suite using the Resource Manager.

The import properties parameter is set to 10MB, but you can increase the parameter if you need to import a larger file.

To change the settings for XML Import file size, modify the size of the **maxImportUpload** parameter so in the following files:

✦ tp_import_export.properties

✦ tp_import_export.properties.in

Appendix A

Error Messages

Gentran Integration Suite error messages and informational messages are noted at the following times:

✦ In the Compile Errors dialog box when you compile the map

✦ In the Error section of the extended rules dialog box when you compile an extended rule containing errors before compiling the map

✦ When you commit an erroneous action in Gentran Integration Suite

The informational messages are dependent on the context of the program, and are intended to be self-explanatory. The error messages are listed in this section, along with the actions you can take to correct the problem.

This section covers the following topics:

✦ Compile Error Messages

✦ Map Editor Error Messages

Note: You can create a Translation Status report, which contains information about the translation of the document and any compliance errors.The errors are listed by those that occur on the input side of the map and those that occur on the output side of the map.

Compile Error Messages

If you compile a map with errors, the compile error messages are displayed in the Compile Errors dialog box. Or, if you compile an extended rule containing errors before compiling the map, error messages are displayed in the Error section of the extended rules dialog box.

After you correct the error, click **Compile** again to verify that the rule is error-free.

The compile error messages are listed by the four- or five-digit message number and the error message text. The error explanations cover the possible causes of the error and the actions that you can take (if appropriate) to correct the error.

Msg ID	Message Text	Explanation/Resolution
1000	expected '.'	The rule does not have the required period (.) between a group name and a field name. Insert a period (.) between the group name and field name.
1001	no statement to compile	The rule 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 comma (,) between parameters. Insert a comma (,) between parameters.
1006	no statements to compile	The body of an IF/ELSE or WHILE condition was empty. Complete the body of the unfinished condition.
1008	expected '#'	The rule does not have the required number sign (#) before a field name. Insert a number sign (#) 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 indexes to access group ...	The rule does not give the full addressing for a group. Complete the addressing for the group.

Msg ID	Message Text	Explanation/Resolution
2003	too many indexes 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 appropriate only 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.
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.

Msg ID	Message Text	Explanation/Resolution
2103	only one wildcard index is permitted	The rule uses more than one wildcard index. Use only one wildcard index 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 destination 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.
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.

Msg ID	Message Text	Explanation/Resolution
4002	expected '+' or '-'	You specified something other than the expected plus sign (+) or minus sign (-). Specify a plus sign (+) or minus sign (-).
4003	expected '*' or '/'	You specified something other than the expected asterisk (*) or slash (/). Specify an asterisk (*) or a slash (/).
4004	expected ')'	You specified something other than the expected right parenthesis ()). Specify a right parenthesis ()).
4005	expected a factor, found ...	The numeric expression is invalid.
4006	expected '('	You specified something other than the expected left parenthesis ((). Specify a left parenthesis (().
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.
4100	expected a date, found ...	A date is required but not supplied. Specify a date.

Msg ID	Message Text	Explanation/Resolution
4101	expected a date modifier, found ...	A date modifier was required but not supplied. Specify a date modifier.
4102	... is not a date	The date expression is invalid.
5000	THEN expected	The compiler expected a THEN condition.
5001	DO expected	The compiler expected a DO condition.
5002	END expected	The compiler expected an END condition.
5003	IF expected	The compiler expected a IF condition.
5004	FROM expected	The compiler expected a FROM condition.
5005	INTO expected	The compiler expected an INTO condition.
5006	END without BEGIN	An END statement was found without a corresponding BEGIN statement. Insert a BEGIN statement in the correct location.
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.
20001	Record..., Field... : date field missing date format	The specified field does not have a date format. Edit the field and select 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.
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.

Msg ID	Message Text	Explanation/Resolution
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.
20030	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 more than one binary length element in one segment.
		Remove the additional binary data and 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 expects that amount of data.
		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 a binary data element to the segment.
20703	group [group] has no active child objects	You tried to compile the map, 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 no Use Code standard rule defines the permitted values.
		Define a Use Code standard rule for the specified attribute.

Msg ID	Message Text	Explanation/Resolution
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. Verify 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 values specified in the code list are valid for an XML attribute.
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.
20714	... : default does not match the attribute type	The default value is the wrong type for the attribute. Correct either the type of the attribute or the default value.
20715	... : Contains illegal character ('<')	The default value contains the invalid character (<). Correct the default value.
20746	Invalid group ordering sequence for group [group name]. Missing start or end Ordering Type.	The start or end ordering type was not set. Configure the start or end ordering type. For more information about ordering types, see *Ordering Tab* on page 326.
20747	Invalid segment/record ordering sequence for segment/record [segment or record name]. Missing start or end Ordering Type.	The start or end ordering type was not set. Configure the start or end ordering type. For more information about ordering types, see *Ordering Tab* on page 326.

Gentran Integration Suite Map Editor Guide

Map Editor Error Messages

The Map Editor error messages are displayed when you commit an erroneous action in the Map Editor. When an error message is displayed, acknowledge the message by clicking **OK** and then take the appropriate action.

In the following table, the error messages are listed alphabetically by the first letter of the error message text. The error explanations provide the actions that you can take to correct the error (if appropriate) and a description that includes possible causes of the error.

Message Text	Explanation/Resolution
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 used by an existing code list. Either change the element ID or delete the existing code list.
The column name [field name] contains an invalid character. The only allowed characters are 'a'-'z', 'A'-'Z', '0'-'9', '_' and ':'. [modified field name] will be used for generated field name.\	You tried to generate fields from a SQL query or table and the chosen column name contains a character that is not permitted in a map field name. You do not need to take any action other than to note that the field name was modified to eliminate the offending character.
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.
Element [XML element name] was defined as abstract, you need to replace abstract content with concrete content	An XML element is defined as abstract and you tried to compile the map. You must either turn off the abstract flag on that element or replace the abstract element with a concrete element. See *Managing XML Elements* on page 144 for more information.
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).
The file that was specified was not found: [file name].	The file you specified to import was not found. Enter the correct file name.
The folder that was specified was not found: [folder name].	The folder you specified to import was not found. Enter the correct folder name.

Message Text	Explanation/Resolution
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.
The Import Wizard failed to import the selected items.	The file or files you selected for import could not be imported.
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.
	Exit Gentran Integration Suite and restart the operating system.
A name must be entered.	You did not specify a name for an object.
Please enter a valid file name.	You entered an invalid filename to import.
	Enter the correct filename.
Please enter a valid folder name.	You entered an invalid folder to import.
	Enter the correct folder name.
A problem occurred while attempting to close loop .... This is probably because of incorrect standards.	An error occurred when reading from the standards.
The product version was not selected.	You did not indicate which product version was used to create the file you specified for import.
	Specify the product version used to create the file.
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.
	Exit Gentran Integration Suite and restart the operating system.
A serious error was encountered whilst accessing the clipboard and the action was abandoned.	Gentran Integration Suite ended a cut, copy, or paste operation because a serious error occurred.
	Perform the cut, copy, or paste operation again.
A system error was encountered while compiling the map.	While compiling a map, a system-related problem, such as lack of disk space, prevented the compile from completing.
	Close unnecessary applications to free disk space, reboot your computer (if necessary), and then recompile the map.
An accumulator must be chosen for this entry.	You did not specify the accumulator to be used in a Use Accumulator standard rule.

Gentran Integration Suite Map Editor Guide

Message Text	Explanation/Resolution
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 Accumulator standard rule.
No alternate accumulator has been selected.	In a Use Accumulator standard rule, you attempted to create an accumulator entry that used an alternate accumulator, but did not specify which alternate accumulator must be used.
No character ranges have been specified for this token.	You did not specify the characters permitted 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 Accumulator standard rule.
No more field mappings can be created.	You attempted to create more than eight field mappings for a Select standard rule.
Standard Rule Select Data Missing	A Select Transaction Register rule is not preceded by one or more Update Transaction Register standard rules. Modify the map so that a Select Transaction Register is preceded by one or more Update Transaction Register standard rules.
Standard Rule Duplicate Transaction Register Error	The data from a Transaction Register standard rule is duplicate data. Investigate why duplicate transaction register data was encountered.
The constant ID must be unique.	You entered a constant identifier in the ID field that is already used in an existing constant. Type a unique constant identifier in the ID field.
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 data format or paste positional data into an XML data format.
The file format cannot be deleted.	You tried to delete a data 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 map could not be compiled.	The map is not ready to be compiled.

Message Text	Explanation/Resolution
The token code must be unique.	You attempted to create a syntax token with the same token identifier as an existing syntax token. Type a unique token identifier in the Token field.
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.	Because of an unknown error, Gentran Integration Suite was unable to begin creating the link.
This file is not a valid 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 subelement delimiter.
You have entered an invalid tag delimiter character or character code.	You entered an invalid tag delimiter.
You must type a description of the translation object.	You attempted to save the translation object without entering the description on the Map Details dialog box. Type the translation object description in the Description field on the Map Details dialog box. This description must be unique because Gentran Integration Suite uses it to identify the map.
You must type a valid character range.	You entered an invalid character range for a syntax token.

Gentran Integration Suite Map Editor Guide

Message Text	Explanation/Resolution
You must type a valid syntax token.	You did not type a valid token identifier in the Token field when creating or editing a syntax token.
You must type 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 type an ID.	You did not specify a constant identifier for the constant in the ID field when creating or editing a syntax token.
You must type the name of the author of this translation object	You attempted to save the translation object without entering the name of the author on the Map 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.

Appendix B

Map Editor Properties

This section describes all of the properties for the map components of the data formats that Gentran Integration Suite supports, as well as the properties of other supporting dialog boxes. Each map component has a property dialog box, which consists of various tabs, which contain the properties. In this section, the properties are described in tables, arranged alphabetically by tab name.

Data format properties information is also available in the Help. When you view a data format properties dialog box, press F1 for Help specific to the tab you are viewing.

This section covers the following topics:

- ✦ Add Parameter Dialog Box
- ✦ Character Set Tab
- ✦ Character Sets Tab
- ✦ Choice Tab (SWIFT Record)
- ✦ Code List
- ✦ Colours
- ✦ Column Tab (SQL)
- ✦ Conditions Tab
- ✦ Confirmations Tab
- ✦ Cursor Operation Tab (SQL)
- ✦ Data Sources Tab (SQL)
- ✦ Decimal Point Tab (Positional and XML)
- ✦ Delimiters Tab (EDI)
- ✦ Delimiters Tab (SWIFT Record, Composite, Field)
- ✦ Delimiters Tab (VLD)
- ✦ Edit Accumulator Entry
- ✦ Edit Character Range
- ✦ Edit Code List
- ✦ Edit Code List Entry
- ✦ Edit Constant
- ✦ Edit Syntax Tokens
- ✦ Element Delimiter Output Tab (EDI)
- ✦ Encoding Tab (File Properties and SWIFT Properties)
- ✦ Entities Tab (XML)
- ✦ Entity Properties (XML)
- ✦ Extended Rule Tab
- ✦ Files Tab
- ✦ Font
- ✦ Key Field Tab, Input Side of Map
- ✦ Key Field Tab, Input Side of Map (SQL)
- ✦ Key Field Tab, Output Side of Map
- ✦ Library Rule Dialog Box
- ✦ Links Tab
- ✦ Loop Extended Rules Tab
- ✦ Looping Tab (CII)
- ✦ Looping Tab (EDI, Positional, Variable Length Delimited, SWIFT Record)
- ✦ Looping Tab (Groups)
- ✦ Looping Tab (SQL)

- ✦ Map Constants
- ✦ Map Details
- ✦ Map Test
- ✦ Message Type Tab (SWIFT)
- ✦ Mode Tab (CII)
- ✦ Name Tab
- ✦ Namespace Tab
- ✦ Numeric Validation Tab (SWIFT)
- ✦ Ordering Tab
- ✦ Output Tab (XML)
- ✦ Position Tab (Positional)
- ✦ Positional Defaults Tab
- ✦ Positional Field Editor
- ✦ Query Tab (SQL)
- ✦ Record Tab (Positional)
- ✦ Repeat Tab (EDI and SWIFT)
- ✦ Repeating Tab (XML)
- ✦ Repeat Suppression (Positional, EDI, Variable-Length Delimited)
- ✦ Rule Library Dialog Box
- ✦ Rule Library Manager Dialog Box
- ✦ Special Tab (EDI)
- ✦ SQL Operation Tab (SQL)
- ✦ SQL Tab (SQL)
- ✦ Standard Formats Tab
- ✦ Standard Rule Tab
- ✦ SWIFT Validation Tab (SWIFT Record)
- ✦ Syntax Record Tab
- ✦ Syntax Tokens
- ✦ Tag Tab (CII)
- ✦ Tag Tab (EDI)
- ✦ Tag Tab (Positional)
- ✦ Tag Tab (SWIFT Record)
- ✦ Tag Tab (VLD)
- ✦ Tag Tab (XML)
- ✦ Tree Tab
- ✦ Type Tab (SWIFTNet)
- ✦ Type Tab, Attributes (XML)
- ✦ Type Tab, Content Particles (XML)
- ✦ Validation Tab (EDI Composites and SWIFT Composites)
- ✦ Validation Tab
- ✦ Version Tab

Add Parameter Dialog Box

The Add Parameter dialog box enables you to add parameters (define variable) to the library. This is equivalent to the declarations section.

The following table describes the properties of the Add Parameter dialog box:

Property	Description
Parameter Name	Type the unique name of the parameter. **Note:** You cannot use spaces or special characters.
Parameter Type	Select the parameter type. Valid values are: ◆ Integer ◆ Real ◆ String ◆ Date
String Length	Specify the length of the string. Only available if you select String for **Parameter Type**.
Passing Mechanism	???Specify the passing mechanism for this parameter. Valid values are: ◆ IN ◆ IN/OUT
OK	Saves the function and closes the dialog box.
Cancel	Cancels the function without saving changes.

Character Set Tab

The following table describes the property on the CII TFD Properties dialog box Character Set tab:

Property	Description
Character Set	Select 8 Bit if the TFD is single-byte, and 16 Bit if the TFD is double-byte.

Character Sets Tab

The following table describes the property on the CII File Properties dialog box Character Set tab:

Property	Description
Select 8 Bit Character Set	Valid values are Default, JIS-X0201, and Other. Default instructs the translator to use the default character setting of the computer on which the translator is running. Use Other if you must specify any other type of character encoding.
Select 16 Bit Character Set	Valid values are Default, JIS-X0208, and Other. Default instructs the translator to use the default character setting of the computer on which the translator is running. Use Other if you must specify any other type of character encoding.

Choice Tab (SWIFT Record)

The following table describes the properties for the SWIFT Choice tab on SWIFT Record Properties dialog box:

Property	Description
Choice Type	Specifies that a SWIFT message defines a list of optional tags (for example, for MT102 at position 13, the field tags 50A,50B,50C are part of a choice and only one of them should appear). Valid values are: ◆ Normal ◆ Start of a choice ◆ Inside a choice ◆ End of a choice

Code List

The following table describes the properties on the Code List dialog box:

Property	Description
Table list	Lists the table identifiers.

Property	Description
New	Accesses the Edit Code List dialog box, which enables you to create a new code list.
Change	Accesses the Edit Code List dialog box, which enables you to edit the selected code list.
Delete	Deletes the selected code list.
Import	Accesses the Open dialog box, which enables you to import a code list.
Export	Accesses the Save As dialog box, which enables you to export the selected code list.
Copy	Copies the selected code list.
Paste	Pastes a previously copied code list in a map.

Colours

The following table describes the properties for the Colours dialog box dialog box:

Property	Description
Item	The type of map entry that you for which you want to select colours. Valid values are: ◆ Group ◆ Record/Segment ◆ Composite ◆ Field/Element
Attributes	Enables you to further separate map entry items according to characteristics like active/inactive or mandatory/conditional.
Sample text	Displays the options you have selected.
Foreground colour	The colour in which the text of the map entry item will be displayed.
Background colour	The colour in which the background (highlighting) of the map entry item will be displayed.

Column Tab (SQL)

The following table describes the properties on the Column tab of the SQL Field Properties dialog box. This tab is available only if the field is contained in an output record.

Property	Description
Name	List of column names in the table which is associated with the parent output record.
Number	List of column numbers in the table which is associated with the parent output record.
Nulls Allowed	Whether Gentran Integration Suite will fill in a blank field with nulls during INSERT and UPDATE SQL statement operations to the column selected from the Name list.
Table Key	Used in a WHERE clause and enables Gentran Integration Suite to format WHERE clauses used in an UPDATE or DELETE (but not INSERT) operations.
Clear	Disassociates this field from any column.

Conditions Tab

The following table describes the properties on the Field (or Element) Properties dialog box Condition tab:

Caution: The Map Editor enables you to edit these conditional relationships, but if you do, you will generate a compliance error. You must use this dialog box for viewing *only*.

Property	Description
Type of relationship	Select the field connection condition from the type of relationship list. Valid values are: ◆ Paired/Multiple – If any of the specified fields are present, all fields must be present. ◆ Required – At least one of the specified fields must be present. ◆ Exclusion – No more than one of the specified fields can be present. ◆ Conditional – If the first Condition field is present, the rest of the fields must also be present. ◆ List Conditional – If the first Condition field is present, at least one of the specified fields must also be present.

Property	Description
Condition field	Select the first field from the Condition field list. This is the field on which the conditional relationship depends if you chose Conditional or List Conditional from the Condition Code list and selected an indexing method from that list.
Available fields	All the fields in the translation object that are valid to be used in a condition at this point.
Fields used in relationship	The fields that you selected (by selecting a field or fields in the Available fields list and clicking the Add button) to be a part of the conditional relationship.
Add	Moves selected fields in the Available fields list to the Fields used in relationship list. The fields are included as a part of the conditional relationship.
Remove	Moves selected fields in the Fields used in relationship list back to the Available Fields list. The fields are removed from the conditional relationship.

Confirmations Tab

The following table describes the properties for the Confirmations tab on the Preferences dialog box, which enables you to specify when you want confirmation messages displayed:

Property	Description
Confirm everything	If selected, specifies that you want the system to backup the map prior to saving. The system will save a copy of the map with the same root name as the map, but with the file extension (.BAK). The backup copy will be stored in the same folder as the map.
	Note: The Backup before saving function is only available when you perform a Save function (not a Save As).
Confirm when I	Specifies individual confirmation messages by action performed.
	Note: These parameters are only available if the **Confirm everything** check box is cleared.

Cursor Operation Tab (SQL)

The following table describes the properties of the SQL Cursor Operation Record Properties dialog box Cursor Operation tab:

Property	Description
Query Record Association	The query statement record with which you associate the cursor operation. The statement record must return a result set.
Cursor Operation	The operation that the translator performs on the result set. Valid values are: ◆ No Op - Performs no operation and enables you to test the effect of removing a cursor operation without actually removing the object. ◆ Move First - Move the cursor to the First row of the result set returned by the Associated Query selected. ◆ Move Next - Move the cursor to the Next row of the result set returned by the Associated Query selected. ◆ Move Last - Move the cursor to the Last row of the result set returned by the Associated Query selected. ◆ Move Previous - Move the cursor to the previous row of the result set returned by the Associated Query selected. ◆ Close - Close the cursor opened for the Associated Query selected.

Data Sources Tab (SQL)

The following table describes the properties of the SQL File Properties dialog box Data Sources tab:

Property	Description
Currently Defined Data Sources	Data sources currently defined on this machine.
DSN	Data source name, as identified by ODBC. You must configure a database pool for Gentran Integration Suite with the same name as the DSN. Each DSN must be unique.

Property	Description
UID	User ID, if necessary.
	◆ This UID is used during mapping. When the current map is used at run time, the UID is used.
	◆ Complete if you use a DSN that requires a user ID to make a connection.
PWD	Password, if necessary. This PWD is used during mapping. When the current map is used at run time, the PWD is used.
	Complete if you use a DSN that requires a password to make a connection.
DATABASE	Database name.
User supplied name for this data source	How you want to refer to the data source name. Complete if you want this name to be different from the ODBC DSN. This name will be used to refer to the data source elsewhere in the map.
Connect to data source to build table schema and test SQL statements	Whether the Map Editor connects to this data source and uses the data source to create the database table schema, test the SQL queries, and generate a result set.
Use Transaction	Whether the translator performs all the operations on the data source in a transaction.
	If there is an error during translation, Gentran Integration Suite rolls the database back to a previous state that does not contain errors. The rollback occurs only if *all* of the following conditions apply:
	◆ Use Translation is selected.
	◆ You are using a local pool in Gentran Integration Suite.
	◆ There were errors in the translator report.
	◆ The "no_rollback_on_validation_error" BPML parameter is either not set or is set to FALSE.
	Note: If you are using a system pool have specified the "no_rollback_on_validation_error" BPML parameter with a value of "Yes," then the translator performs a commit. Any SQL errors are reported in the translator report.
SQL Data Sources	Accesses the Select Data Source dialog box, which enables you to create a data source or select one that was previously created.
Clear Selected Data Source	Clears the parameter boxes to enable you to add another data source.
Test Connection	Tests the connection of the selected SQL data source.

Property	Description
Update/ Add	Recreates the database schema for the selected data source if you specified Test Connection. After the Map Editor recreates the schema, it automatically checks all the objects in the file format that depend on that schema. If possible, the Map Editor also modifies the data type and validation information for those objects to match the new schema. Then, the Map Editor performs a consistency check and tries to fix those inconsistencies.
	If you do not select a data source or there are no data sources in the list, Gentran Integration Suite displays Update as Add. The Add option enables you to add a data source you previously created to the Currently Defined Data Sources list.
	See *Checking Database Consistency* on page 127 for more information about database consistency checking.
Remove	Deletes the selected data source from the list, invalidates it, and then clears all fields or records that reference the data source. You are prompted to remove the selected data source (from the list, not from the computer); click **OK** to do so.

Decimal Point Tab (Positional and XML)

The following table describes the properties of the Positional File Properties and XML File Properties dialog box Decimal Point tab:

Property	Description
Define Decimal Point	Whether the decimal point character is user-defined.
Decimal Point Character	The character that the translator will identify as a decimal point. The default is a period (.).

Delimiters Tab (EDI)

The following table describes the properties of the File Properties dialog box Delimiters tab:

Property	Description
Specify defaults	Whether to activate the default boxes on the tab and establish syntax defaults.
	Note: If the delimiters differ from the defaults specified, type either the character or the hexadecimal value in the correct box.

Property	Description
Stream segments	Whether the translator writes segments without a carriage return/line feed at the end. The default is not checked, which indicates that each segment is written with a carriage return/line feed at the end.
Tag Delimiter	Default character that marks the end of each segment tag (identifier).
	This box is active only if the Specify defaults check box is selected.
	Type either the character or the hexadecimal value in the correct box.
Segment Delimiter	Default character that marks the end of each segment.
	This box is active only if the Specify defaults check box is selected.
	Type either the character or the hexadecimal value in the correct box.
Element Delimiter	Default character that marks the end of each element.
	This box is active only if the Specify defaults check box is selected.
	Type either the character or the hexadecimal value in the correct box.
Sub Element Delimiter	Default character that marks the end of each subelement.
	This box is active only if the Specify defaults check box is selected.
	Type either the character or the hexadecimal value in the correct box.
Repeating Element Delimiter	Default character that marks the start of a new repetition of an element or subelement.
	This box is active only if the Specify defaults check box is selected.
	Type either the character or the hexadecimal value in the correct box.
Release Character	Character that, when used before any defined delimiter, restores the character used for the delimiter to its original meaning. For example, if the plus sign (+) is the element delimiter and question mark (?) is the release character, when ?+ is a part of the data, Gentran Integration Suite reads the + as a part of the data instead of marking the end of the element.
	This box is active only if the Specify defaults check box is selected.
	Type either the character or the hexadecimal value in the correct box.
	If you are mapping delimited data and the setup outside the map does not enable you to specify the release character, type the value you expect in this box.
Decimal Separator	Character that indicates the decimal point in a numeric field, such as a period (.) or comma (,).
	This box is active only if the Specify defaults check box is selected.
Default suppression and padding of leading zero on Numeric values	Leading zeros are suppressed for all values greater than 1. Decimal values less than 1 will retain a single leading zero (e.g., 000.25 --> 0.25).
	This is the default selection.
	This box is only active on the output side of the map for all EDI types.
Suppress leading zero on Numeric R* format values (e.g., 0.25---> .25)	Same as default behavior except leading zeros are also suppressed on elements that use an R-format numeric and whose value is less than 1.
	This box is only active on the output side of the map for all EDI types.

Property	Description
Pad with leading zero on Numeric values (e.g., 25---> 000025)	Numeric values, regardless of format, are padded with leading zeros out to the maximum length of the field. This box is only active on the output side of the map for all EDI types.
Always output trailing delimiters for fields with data	This is supported on the Output side of the map only, as a syntax-level option and field-level option (NACS uses it as a syntax-level option).
	If this option is selected, a trailing level option is always output after a field with data. For example, instead of **1234*abcd**, **1234*abcd*** is output.

Delimiters Tab (SWIFT Record, Composite, Field)

The following table describes the properties for the Delimiters tab on the SWIFT Record Properties, Composite Properties, and Field Properties dialog boxes for SWIFT fields:

Property	Description
Start	Default character that marks the beginning of the composite.
End	Default character that marks the end of the composite.

Delimiters Tab (VLD)

The following table describes the properties on the Variable Length Delimited Data Properties Delimited tab:

Property	Description
Use defaults	Selects all of the default delimiter settings.
Include column (field) names from first record	On the input side, specifies that the first record is a header (therefore the information in it will not be processed as data).
	On the output side, instructs the translator to create a header record in the output file.
Field Delimiter	Character that separates fields.
Quote Character	Character that quotes fields.
Quote fields containing delimiter (default)	Output side only. Specifies that only fields that contain the delimiter are quoted.

Property	Description
Quote text fields and fields containing delimiter	Output side only. Specifies that all string fields and all fields that contain the delimiter are quoted.
Quote all fields	Output side only. Specifies that all fields are quoted.

Edit Accumulator Entry

See Chapter 8, *Using Standard Rules* for more information about using the Use Accumulator standard rule.

The following table describes the properties on the Edit Accumulator Entry dialog box. For the valid accumulator operations, see *Accumulator Operations* on page 160.

Property	Description
Primary Accumulator	Primary accumulator. The following criteria apply: ◆ Before any calculations are performed on an accumulator, its content is zero (0). When you use an accumulator, a new accumulator is added to the end of this list. ◆ There is *only one* set of accumulators for each map. This means that accumulator 0, whether it is used in the Primary accumulator or Alternate Accum box, is the same accumulator with the same contents. If you assign calculations to accumulator 0 at the beginning of the map and then use accumulator 0 again later in the map, the content of that accumulator is the result of the earlier calculation. Any additional calculations you assign to that accumulator are performed on the contents resulting from an earlier calculation.
Name	Descriptive alias that enables you to indicate what the accumulators you create are used for.
First	First operation that is performed. The First box is active only after you select a primary accumulator. Before any calculations are performed on an accumulator, its content is zero (0). When you use an accumulator, a new accumulator is added to the end of this list.
Second	Second operation that is performed, 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 operation.
Third	Third operation that is performed, after the second operation is complete. The Third box is active only after you select a Second operation.
Fourth	Fourth operation that is performed, after the third operation is complete. The Fourth box is active only after you select a Third operation.
Alternate Accum	Alternate accumulator operation.

Edit Character Range

The following table describes the properties on the Edit Character Range dialog box:

Property	Description
Start Character	Character that begins the permitted token range in the Start character box. For example, if the character range you want to define is B through D, type B in the Start character box. You can also type hexadecimal character codes. **Note:** The Start character can only be one character, upper- or lower-case alphabetics or numerics 1 - 9.
End Character	Character that terminates the permitted token range in the End character box. For example, if the character range you want to define is B through D, type D in the End character box. You can also type hexadecimal character codes. **Note:** The End character can only be one character, upper- or lower-case alphabetics or numerics 1 - 9.
OK	Closes the Edit Character Ranges dialog box and saves changes.
Cancel	Closes the Edit Character Ranges dialog box without saving changes.

Edit Code List

For more information about using the Use Code standard rule, see Chapter 8, *Using Standard Rules*.

The following table describes the properties on the Edit Code List dialog box:

Property	Description
Table ID	Name of the field for which this code list is used.
Desc	Description of the field for which this code list is used.
Allowed Codes	Code list entries that are permitted for this code list.
New	Accesses the Edit Code List Entry dialog box, which enables you to create a new code list entry.
Change	Accesses the Edit Code List Entry dialog box, which enables you to edit the selected code list entry.
Delete	Accesses the Edit Code List Entry dialog box, which enables you to delete the selected code list entry.

Property	Description
Load	Accesses the Load Code List dialog box, which enables you to select, from a standard code list, code list entries that you want to load or the entire list of code list entries.

Edit Code List Entry

See Chapter 8, *Using Standard Rules* for more information about using the Use Code standard rule.

The following table describes the properties on the Edit Code List Entry dialog box:

Property	Description
Value	Value of the code list entry.
Description	Code list entry value description. The description is used if you specify (in the Store Fields list on the Field Properties dialog box) a field to which you want the code list entry description mapped.

Edit Constant

For more information about using constants, see Chapter 8, *Using Standard Rules*.

The Edit Constant dialog box is accessed from the **Map Constants** dialog box. The following table describes the properties on the Edit Constant dialog box:

Property	Description
ID	Constant identifier. The ID is typically a description of the field in which the constant is used. If you must refer to the constant in an extended rule, you must use the data from this field.
Type	Category of the constant. Valid values are: ◆ Integer **–** Numeric constants that are a positive or negative natural (non-fraction) number or zero (0). ◆ Real **–** Numeric constants that are a positive or negative integer with an explicit decimal point. ◆ String **–** Alphanumeric constants.
Value	Constant expression. This is the value of the constant.

Edit Syntax Tokens

The following table describes the properties of the Edit Syntax Tokens dialog box:

Property	Description
Token	Unique one-character alphanumeric value that the translator will recognize as containing the permitted range of characters you designate. **Note:** The Token can only be one (1) unique character, upper- or lower-case alphabetics or numerics 1 - 9.
Character ranges	Character range or ranges that you defined for this token. You can define more than one character range for each token. For example, you can define the token A as permitting the character range A - Z and the character range a - z. This indicates that token A permits upper-case and lower-case alphabetics only.
OK	Closes the Edit Syntax Tokens dialog box and saves changes.
Cancel	Closes the Edit Syntax Tokens dialog box without saving changes.
New	Accesses the Edit Character Range dialog box to enable you to create a new character range.
Change	Accesses the Edit Character Range dialog box to enable you to edit the selected character range.
Delete	Deletes the selected character range. **Note:** The selected entry will be deleted without warning.

Element Delimiter Output Tab (EDI)

The following table describes the properties of the Element Properties dialog box Element Delimiters Output tab:

Property	Description
Always output an element delimiter for this field if it	This is supported on the Output side of the map only, as a field-level option. If this option is selected, a trailing level option is always output after an element with data. For example, instead of **1234*abcd**, **1234*abcd*** is output.

Encoding Tab (File Properties and SWIFT Properties)

The following table describes the property of the Encoding tab for the File Properties dialog boxes (including SWIFT):

Property	Description
Encoding	Specify the type of character encoding by choosing an encoding format from the list. **Note:** For SWIFT, encoding can be set but it is not necessary.

Entities Tab (XML)

The following table describes the properties of the Entity Properties dialog box Entities tab (XML only):

Property	Description
Name	Name of the entity.
Description	Brief description of the entity.
New	Accesses the Entity Properties dialog box, which enables you to create an entity.
Change	Accesses the Entity Properties dialog box, which enables you to edit the selected entity.
Delete	Deletes the selected entity.
Copy	Copies the selected entity, enabling you to use the entity in another map.
Paste	Pastes a previously-copied entity, enabling you to copy the entity from one map to another.

Entity Properties (XML)

The following table describes the properties of the Entity Properties dialog box (only applies to XML):

Property	Description
Name Tab	

Property	Description
Name	Entity name.
Description	Description of the entity. This box is used to differentiate the entity from similar entities.
Tag Tab	
Tag	Entity identification tag as it appears in the XML document. The translator uses the entity name by default. The tag must match the entity tag in the XML document.
Entity Tab	
Entity Value	Entity data. The translator inserts this data when it encounters the entity.

Extended Rule Tab

The following table describes the properties of the Extended Rule tab:

Property	Description
Extended Rule	Area where you type the extended rule. For more information about extended rules, see Chapter 9, *Using Extended Rules*.
Full Screen	Enlarges the extended rule area to the size of your entire display to give you more room to see what you type (click the right button to return to normal display size).
Compile	Compiles the extended rule. This function enables you to view compile errors for this rule prior to compiling the translation object. This function gives you immediate feedback about the accuracy of your rule. Double-click an error to immediately navigate to the line containing the error. **Note:** The rule is also compiled when you compile the map.
Errors	If you compiled this extended rule, any warnings or errors are displayed in the Errors list. Double-click an error to make the cursor go to the line containing the error.

Files Tab

The following table describes the properties for the Files tab on the Preferences dialog box, which enables you to specify Gentran Integration Suite folder and backup options:

Property	Description
Backup maps to a .BAK file before saving	If selected, specifies that you want the system to backup the map prior to saving. The system will save a copy of the map with the same root name as the map, but with the file extension (.BAK). The backup copy will be stored in the same folder as the map. **Note:** The Backup before saving function is only available when you perform a Save function (not a Save As).
When browsing for maps, start in this folder	Specify a default folder to which you want to be directed when you browse for a map.
When compiling a map, save to this folder	Specify a default folder where the compiled maps will be stored.

Font

The following table describes the properties for the Font dialog box dialog box:

Property	Description
Font	The list of the available fonts on your machine.
Font Style	The desired style of the selected font.
Size	The size of the selected font.
Sample	Displays the options you have selected.
Script	The available language scripts for the selected font.

Key Field Tab, Input Side of Map

The following table describes the properties of the Key Field tab on the Input side of the map; the properties on this tab apply to EDI Segments, Positional Records, Variable Length Delimited Data Records, XML File properties, SWIFT Records, and XML Elements.

For SWIFTNet, key field matching in the translator is used to match by Qualifier for generic fields, and by the data content of 16R/16S ISO 15022 block tags.

Property	Description
Field	Elements in the segment (or, for XML, contains all the attributes that are defined for this element). Select an element from the Field list that will or will not match the constant chosen from the Matching Rules section. This list helps you recognize an ambiguous segment definition. If the specified condition is not met, the segment does not conform to the definition, but processing still continues. **Note:** Key fields are intended to be used with string fields only. If you use a key field on a numeric field, the result is not guaranteed.
Use constant/ Edit	List of defined constants. The Map Editor matches the selected constant against the key field.For XML, indicates that the translator must match the element if the contents of the selected attribute match the literal constant selected from the list. If you must add or change a constant, click **Edit** to access the Map Constants dialog box.
Use codelist/ Edit	List of defined code lists. The Map Editor matches the selected code list against the key field. For XML, indicates that the translator must match the element if the contents of the selected attribute match the selected code list. If you must add or change a code list, click **Edit** to access the Code Lists dialog box.
Match record when key does not match	Use if you have specified a key field and a matching rule. Select this check box to specify when the segment will be matched, if the Field does not contain the value specified in the Matching rules section. If the specified condition is not met, the segment does not conform to the definition, and processing continues. Not used for SWIFT. The Matching Rules section is active only if you selected a map component from the Key Field list.

Key Field Tab, Input Side of Map (SQL)

The following table describes the properties of the SQL Input Record dialog box Key Field tab (on the Input side of the map only):

Property	Description
None Use constant Use codelist Use Field	Type of information that the translator uses to match this record. ◆ Select Use constant or Use codelist to activate the middle section of the dialog box. ◆ Select Use Field to activate the lower section of the dialog box.
Match record when key does not match	Translator must match the record if the Key Field does not have the value specified.

Property	Description
Key Field	Field that the translator checks to verify whether the key field does or does not (depending on which you specify) match the constant, code list, or field value or values. **Note:** Key fields are intended to be used with string fields only. If you use a key field on a numeric field, the result is not guaranteed.
Constants Edit	List of defined constants. The translator matches the selected constant against the key field. To add or change a constant, click Edit to access the Map Constants dialog box.
Code Lists Edit	List of defined code lists. The translator matches the selected code list against the key field. To add or change a code list, click Edit to access the Code Lists dialog box.
Key Field 1	First key field and contains all the active fields from this record. **Note:** Key fields are intended to be used with string fields only. If you use a key field on a numeric field, the result is not guaranteed.
Use Field 1	Specifies that the first key field must match the value in the field from an earlier record and contains all the active fields from the preceding records (not including this record).
Key Field 2	Second key field, if necessary, and contains all the active fields from this record. **Note:** Key fields are intended to be used with string fields only. If you use a key field on a numeric field, the result is not guaranteed.
Use Field 2	Specifies that the second key field must match the value in the field from an earlier record and contains all the active fields from the preceding records (not including this record).
Key Field 3	Third key field, if necessary, and contains all the active fields from this record. **Note:** Key fields are intended to be used with string fields only. If you use a key field on a numeric field, the result is not guaranteed.
Use Field 3	Specifies that the third key field must match the value in the field from an earlier record and contains all the active fields from the preceding records (not including this record).

Key Field Tab, Output Side of Map

The following table describes the properties of the Key Field tab on the Output side of the map; the properties on this tab apply to EDI Segments, Positional Records, SQL Output Record Properties, Variable Length Delimited Data Records, XML File properties, SWIFT Records, and XML Elements.

For SWIFTNet, key field matching in the translator is used to match by Qualifier for generic fields, and by the data content of 16R/16S ISO 15022 block tags.

Property	Description
Field	If you specify a key field, the translator generates an output record only if the specified field has data in it.
	Note: Key fields are intended to be used with string fields only. If you use a key field on a numeric field, the result is not guaranteed.

Library Rule Dialog Box

The Library Rule dialog box enables you to define an extended rule and any parameters you want to pass to it. This is equivalent to the statements section.

The following table describes the properties of the Rule Library dialog box:

Property	Description
Rule Name	Name of the rule.
	Note: You cannot use spaces or special characters.
Rule Purpose	Description of the purpose of the rule (what the rule is used to accomplish).
Return Type	Specify the data type of the value that will be returned. Valid values are: ◆ Integer ◆ Real ◆ String ◆ Date
String Length	Specify the length of the string. Only available if you select String for **Return Type**.
Parameters	List of parameters displaying the parameter name, type, and passing mechanism (reference).
Add Parameter	Accesses the Add Parameter Dialog Box so you can add a new parameter.
Modify Parameter	Accesses the Add Parameter Dialog Box for the selected parameter so you can modify it.
Delete Parameter	Deletes the selected parameter.
Please enter the contents of the extended rule below	Area where you type the extended rule. For more information about extended rules, see *Using Extended Rules*.

Property	Description
Errors	If you compiled this extended rule, any warnings or errors are displayed in the Errors list. Double-click an error to make the cursor go to the line containing the error.
OK	Saves the function and closes the dialog box.
Compile	Compiles the extended rule. This function enables you to view compile errors for this rule prior to compiling the translation object. This function gives you immediate feedback about the accuracy of your rule. Double-click an error to immediately navigate to the line containing the error. **Note:** The rule is also compiled when you compile the map.
Cancel	Cancels the function without saving changes.
Full Screen	Enlarges the extended rule area to the size of your entire display to give you more room to see what you type (click the right button to return to normal display size).

Links Tab

The following table describes the properties for the Links tab on the Preferences dialog box, which enables you to specify how you want the mapping links (visual lines that connect two mapped items) displayed:

Property	Description
Show no links	Do not visually display mapping links.
Show to or from the currently selected element	Display only the mapping links for the currently selected map component (this option enables you to concentrate on the selected field and removes the confusion of viewing many links at once).
Show links to or from all visible elements	Display all the mapping links.

Loop Extended Rules Tab

The following table describes the properties on the Loop Extended Rules tab:

Property	Description
On Begin	Extended rule to be run before the translator processes the map object.

Property	Description
On End	Extended rule to be run after the translator concludes processing the map object. The Map Editor processes On End rules at the end of each loop occurrence, not at the end of all loops.
Full Screen	Maximizes the dialog box.
Compile	Compiles the extended rule. This function enables you to view compile errors for this rule prior to compiling the translation object. This function gives you immediate feedback about the accuracy of your rule. Double-click an error to immediately navigate to the line containing the error. The rule is also compiled when you compile the map.
Extended rule	Defines the extended rule.
Errors	Displays any errors generated when you clicked the Compile button to compile the extended rule. Double-click an error to make the cursor go to the line containing the error.

Looping Tab (CII)

The following table describes the properties on the CII TFD Properties dialog box Looping tab:

Property	Description
Normal Loop Start Loop End Loop Repeat	Type of loop. Values are: ◆ Normal – In the loop, but is not the beginning or ending ◆ Loop Start – Marks the beginning of the loop ◆ Loop End – Marks the end of the loop ◆ Loop Repeat – Marks the end of one loop iteration and the beginning of the next iteration

Gentran Integration Suite Map Editor Guide

Looping Tab (EDI, Positional, Variable Length Delimited, SWIFT Record)

The following table describes the properties on the Looping tab for the EDI Segment, Positional Record, SWIFT Record, Group Properties and Variable Length Delimited Data Record dialog boxes:

Property	Description
Min usage	Minimum number of times the segment must loop (repeat).
	For a conditional loop, the minimum usage is 0 (zero). For a mandatory loop, the minimum usage is 1 or greater.
	Note: If a segment is required for HIPAA, indicate the requirement by setting the minimum usage to **1**. If a segment is designated as situational for HIPAA, indicate this by setting the minimum usage to **0**.
Max usage	Maximum number of times the segment must loop (repeat).
Normal Loop Start Loop End	Type of loop. Valid values are: ◆ Normal – In the loop, but is not the beginning or ending ◆ Loop Start – Marks the beginning of the loop ◆ Loop End – Marks the end of the loop **Note:** This parameter is not valid for SWIFT.
Get usage from related field	This is supported on the input side of the map only. If this option is selected, you can associate the usage of a group or segment with the value stored in another field. **Note:** This options is disabled for SWIFT. The field associated with the usage must meet the following requirements: ◆ It must be in a segment that is contained in the same group as the segment whose usage is being determined. ◆ It must be in a non-repeating segment. ◆ It must be in a segment that precedes the segment whose usage is being determined. ◆ It must be an integer field.

Looping Tab (Groups)

The following table describes the properties of the Looping tab for EDI Group, Positional Group, Variable Length Delimited Data Group, CII Group, SWIFT Group, and SQL Group dialog boxes:

Property	Description
Minimum usage	Minimum number of times the group must be repeated.
	For a conditional group, the minimum usage is 0 (zero).
	Note: If a group is required for HIPAA, indicate the requirement by setting the minimum usage to **1**. If a group is designated as situational for HIPAA, indicate this by setting the minimum usage to **0**.
Maximum usage	Maximum number of times the loop must be repeated.
Promote records to parent	Select this check box to specify that when the group is compiled, the subordinate records and groups are extracted from the loop and located in the parent group. This function is valid for single iteration subgroups only.
	Not used with SWIFT.
	Note: Select this check box if you are mapping the fields in this group to data that is not grouped in the output.
Get usage from related field	This is supported on the input side of the map only. If this option is selected, you can associate the usage of a group or segment with the value stored in another field.
	Note: This options is disabled for SWIFT.
	The field associated with the usage must meet the following requirements:
	◆ It must be in a segment that's contained in the same group as the segment whose usage is being determined.
	◆ It must be in a non-repeating segment.
	◆ It must be in a segment that precedes the segment whose usage is being determined.
	◆ It must be an integer field.

Looping Tab (SQL)

The following table describes the properties on the Looping tab for the SQL Input and Output Record dialog boxes:

Property	Description
Min Usage	Minimum amount of times the loop must repeat.
	If the Min Usage box contains 0 (zero), the record is conditional. If the Min Usage box contains 1 or greater, the record is required.
Max Usage	Maximum amount of times the loop can repeat.
Normal Loop Start Loop End	Type of loop. Select Normal. This record is in the loop but is not the beginning or ending record. Loop Start and Loop End are used for EDI.
Automatically get next row from Statement record	If selected, the translator performs a movement operation as the input record loops. This property is deactivated on SQL Output Records.

Map Constants

The following table describes the properties on the Map Constants dialog box:

Property	Description
ID ... Type ... Value	Lists the data for all constants currently defined in the translator.
Close	Exits the Map Constants dialog box.
New	Accesses the Edit Constant dialog box, which enables you to create a new constant.
Edit	Accesses the Edit Constant dialog box, which enables you to edit the selected constant.
Delete	Removes the selected constant from the translator.

Map Details

The Map Details dialog box enables you to edit the details of the map, including the description and version information.

This dialog box also enables you to instruct the translator to use the pad character and alignment settings of each string field when reading a positional file, to determine how to

trim pad characters from the string data. The following table describes the properties of the Map Details dialog box:

Property	Description
Author	Person or department that created the map. Required.
Description	Unique description of the map. Required. The translator uses this description to identify the map. Some identifying characteristics that you can use are the trading partner that this map is used for, the standard, the version, or the type of transaction this map uses.
	For example, MWT X 3030 850 is the description of a map used with partner MWT, for an ANSI X12 version 003030 Purchase Order (850).
Map Function	Select the Sterling Integrator map type from the list. Required. The other map options are available to enable you to import maps from other Sterling Commerce products in order to convert them to Gentran Integration Suite maps.
	Note: If you are importing a map, you must change the map function to Sterling Integrator.
System	This option is not needed for a Sterling Integrator map type.
Use Configurable Trimming	Instructs the translator to use the pad character and alignment settings of each string field when reading a positional field to determine how to trim pad characters from the string data.
	If the alignment setting indicates that the data is aligned to the left of the field, then pad characters are trimmed from the right of the field. If the alignment setting indicates that the data is aligned to the right of the field, then pad characters are trimmed from the left of the field. If the alignment setting indicates that the field data is aligned in the center of the field, then pad characters are trimmed from both the left and right of the field. Default is center alignment.
	For example, use configurable trimming to preserve either trailing or leading spaces in positional data.
Gentran:Server for Windows NT 2.x Compatible Rule Execution	Use this option only if you are using a map that was created with Gentran:Server for Windows NT 2.x and you need the rules to run exactly as they did in that product.
Major version/ Minor version	Designate different versions of a map. The valid values for each box are in the range 0 - 255. These two boxes are available for your use only. Gentran Integration Suite does not use them.
	For example, if you type 5 in the Major version box and 45 in the Minor version box, the map version number is 5.45.
Compiled on	Date that the map was compiled. The field is blank if the map has not been compiled.
EDI Associations area	EDI information, if you selected Delimited EDI for the input, output, or both sides of the map.
	Note: Change these fields only if you want to change the EDI agency, version, transaction, or functional group of an existing map. You can copy and change an existing map.

Property	Description
OK	Saves changes and exits the dialog box.
Cancel	Exits the dialog box without saving changes.

Map Test

The following table describes the properties on the Map Test dialog box:

Property	Description
Login Server and port	Gentran Integration Suite server name and *dashboard* port in the format [Gentran Integration Suite server name]:[dashboard port]. The port number required for this parameter is *not* the base port for the install but rather the dashboard port. The dashboard URL and port are typically listed in the UNIX shell after Gentran Integration Suite is successfully started. This box cannot be left blank.
Login Server User Name	Gentran Integration Suite user name that is used by the Map Editor to access the Map Test service.
Login Server Password	Gentran Integration Suite password that is used by the Map Editor to access the Map Test service.
Proxy Server and port	Proxy server name and port for the client machine in the format [proxy server name]:[port]. **Note:** This parameter is only necessary if you are executing translation beyond a system firewall.
Translation Object Browse	Name of the compiled translation object. Browse enables you to select the translation object. This parameter is mandatory. **Note:** The path you supply must be valid and the translation object extension must be .txo
Data File Browse	Name of the data file you wish to use when translating data with the test map. Browse enables you to select the data file. **Note:** If you supply a path, it must be valid.

Message Type Tab (SWIFT)

The following table describes the properties for the Message Type tab on SWIFT Properties dialog box:

Property	Description
Message Type	Number of the SWIFT message being implemented by the map (for example, 102).
Message Practice ID	Identifier for the custom designed market practice.
	Note: This identifier must be upper-case alphanumeric with no spaces.

Mode Tab (CII)

The following table describes the properties for the CII File Properties dialog box Mode tab:

Property	Description
Dividing Mode	This option is not used in the Map Editor; it does not affect translation. For outbound processing, specify dividing mode in the document envelope.
Non-Transparent Mode	This option is not used in the Map Editor; it does not affect translation. For outbound processing, specify non-transparent mode in the document envelope.
Always output loop repeats	Whether loop repeat tags must be put in the output file, up to the maximum usage for the group.

Name Tab

The following table describes properties on the Name tab:

Property	Description
Name	Map component name.
	Note: Do not use spaces or hyphens (-) in the name. You can use the underscore (_) to separate words.
Business Name	Indicates what business data the map component contains.
	Note: Do not use spaces or hyphens (-) in the name. You can use the underscore (_) to separate words.

Property	Description
Description	Description of the map component.
Additional notes	When you import an XML schema, the annotation attached to each XML map component is saved in this box. For non-XML map components, contains any user-defined comments.

Namespace Tab

The following table describes the properties on the XML File Properties dialog box Namespace tab:

Property	Description
Enable namespace support	Enable XML namespaces in this side of the map.
Use same namespace as parent element	Use the namespace specified for the immediate parent element.
Use this namespace	Use an existing namespace in the field, which you type in the following field.
Define namespace prefixes New Edit Delete	Output side only. Click **New** and two fields display in this area. The first field is where you type the prefix. The second field is where you type the namespace. To edit a namespace prefix, click the line you want to edit and click **Edit**. To delete a namespace prefix, click the line you want to delete and click **Delete**.

Numeric Validation Tab (SWIFT)

The following table describes the properties of the Field Properties dialog box (for SWIFT) Repeat tab:

Property	Description
Pad with leading zero on Numeric values (e.g., 25---> 000025)	When selected, specifies that numeric values, regardless of format, are padded with leading zeros out to the maximum length of the field. This parameter is only active for numeric SWIFTNet fields.

Ordering Tab

The following table describes the properties on the Ordering tab (used on the Group Properties, Variable Length Delimited Data Record Properties, Positional Record Properties, SWIFT Record Properties, and EDI Segment Properties dialog boxes):

Property	Description
Normal Start random order Inside random order End random order	Used for the *input* side of the map only. The order in which the group, segment, or record can occur. Not used for SWIFT. **Note:** If you are using a non-EDI syntax and wish to use this feature, you must manually set the ordering start/inside/end group and segment points and also set the associated Ordering Tag. **Note:** For SWIFTNet, use the random order feature to process ISO 15022 messages (MT 5xx), for which a defined set fieldtags/qualifiers can appear in any order.
Ordering Tag	Used to perform validation on the ordering types to ensure there is a start and end for every defined ordering sequence. The following caveats apply to this parameter: ◆ The Ordering Tag is populated automatically when reading EDI standards, including the HIPAA X12 transactions. ◆ For EDI syntaxes, the group Ordering Tag is populated with the group name (for example, if the group name is 100_NM1, then the Ordering Tag is populated with 100—the underscore and any alphabetic characters are removed from the group name to generate the Ordering Tag). ◆ For EDI syntaxes, the segment Ordering Tag is populated with the segment name. For example, if the segment name is REF, the Ordering Tag is populated with REF. Or, if the segment name is REF:2, the Ordering Tag is populated with REF (the colon and any subsequent numbers are removed to generate the Ordering Tag). ◆ For SWIFT maps, this feature is automatically set. If you are using a non-EDI syntax and wish to use this feature, you must manually set the ordering start/inside/end group and segment points and also set the associated Ordering Tag. ◆ The Ordering Tag must be the same for a related sequence of groups or segments. ◆ The Ordering Tag is not valid for the XML syntax. ◆ This parameter is only enabled (and is mandatory) when you select Start random order, Inside random order, or End random order.

Output Tab (XML)

The following table describes the properties of the XML File Properties dialog box Output tab (available only on the output XML side of the map):

Property	Description
No prolog or document type declaration	If selected, the translator does not generate any header information for the XML document (neither a prolog nor a document-type declaration).
Prolog specified	If selected, the translator generates a prolog at the start of the XML document.
Prolog and document type declaration specified	If selected, the translator generates both a prolog and a document type declaration at the start of the XML document.
Public ID	The public identifier that the translator uses to create the document type declaration. This box is available only if you select the Prolog and document type declaration option.
System ID	Identifier that the translator uses to create the document-type declaration. This box is available only if you select the Prolog and document type declaration option.
No newlines	Output data is wraparound (streamed).
One element per line, indented to show element hierarchy	Output data is formatted hierarchically and indented.
One element per line, no indentation	Output data is formatted without indentation.
Encoding	Type of character encoding by choosing an encoding format from the list.

Position Tab (Positional)

The following table describes the properties on the Field Properties dialog box Position tab (available only on the Input side of the map):

Property	Description
Start position	Starting position of the specified element in the record.
Number of characters	Total length of the element.

Property	Description
Character used in empty portions of the field	Pad character that the translator uses for this field if the field value is less than the number of valid positions. A pad character holds the places that the field value does not occupy.
	For example, if a field is eight positions long and the field value is AAA (which is only three characters), you will need five placeholders to round out this eight-character field. If you use the ampersand (&) as the pad character, and left-align the field, the field will appear as AAA&&&&&.
Alignment	Whether the field is left-aligned, centered, or right-aligned in the valid number of characters.

Positional Defaults Tab

The following table describes the properties for the Positional Defaults tab on the Preferences dialog box, which contains the global defaults for the positional file format:

Property	Description
Data Type	Type of field formatting allowed. Valid values are:
	◆ String
	◆ Numeric
	◆ Dt/Tm
	Note: See *Formatting Data in Fields* for an explanation of these data formats.
Pad Character	Enter the type of "padding" (a character that holds the places that the field value does not occupy) that the system should use for fields if the field value is less than the number of enabled positions.
	Note: The default setting is a space ("0").
	For example, if a field is eight positions long and the field value is **AAA** (only 3 characters), you need 5 placeholders to round out this 8-character field. If you use the character **&** as the padding character, and left align the field, the field appears as **AAA&&&&&**.
	Note: When using the CII standard, this field should be set to type **K** (double-byte). Since type K fields have double-byte spaces, so you cannot use SP as a pad character. Instead, you have to use the hex value for double-byte spaces which in Japanese is **0x8140**.
Alignment	Specify whether fields are left-aligned, centered, or right-aligned in the enabled number of characters.
Allow/generate a '+' for positive numbers	If selected, enables you to use and generate a "+" on the input and/or output sides of a map.

Positional Field Editor

The following table describes the properties for the Positional Field Editor dialog box, which enables you to quickly and easily edit the most common attributes of each field in the record, and add or delete fields from the record:

Property	Description
Record	Displays the record name, tag, and tag position.
Name	Name of the field.
Mandatory	Indicates that the field is required.
Description	Description of the field.
Data Type	Type of data for this map component. Valid values are: ◆ String - Alphanumeric ◆ Number - Numeric, real, overpunched, or packed (for Positional only) ◆ Date/time - Date or time ◆ Bin Data - binary data (only available if you select "Binary" on the Special tab of the EDI Segment Properties dialog box) ◆ Bin Length - length of binary data (only available if you select "Binary" on the Special tab of the EDI Segment Properties dialog box) **Note:** 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.
Format	How the data in the map component will be formatted. Depending on the data type you selected, you can either: ◆ For the String data-type, select a syntax token to denote that this map component must be formatted as the syntax token dictates. **Note:** Free Format indicates that any characters are acceptable in the field. The translator does not check the characters for compliance. ◆ For Number or Date/Time data-type, select the data format from a list.
Start Pos	Starting position of the field in the record.
Min Length	Minimum number of characters in the field.
Max Length	Maximum number of characters in the field.

Property	Description
Fields	Displays each field in the record, in sequence. The information that is displayed in this list is the field name, whether it is mandatory or conditional (M or C), the field type, the starting position, and the positional length of each field.
New	Adds a new field to the record directly after the selected field. **Note:** You need to complete the Field Details for the new field.
Delete	Deletes the selected field.
Auto Position	Activates the Auto Position function, in which the system automatically positions the fields in the record. The criteria used is that each field is positioned directly after the previous field and is of the length specified in the Maximum box on the Field Properties dialog. Click **Yes** to acknowledge the warning message that fields will be sequenced in order. **Note:** It is only valid to use the Auto Position function if you do *not* define a record Tag, and if you define every field in the record in the sequence that each field occurs.

Query Tab (SQL)

The following table describes the properties of the SQL Field Properties dialog box Query tab (available only if the field is contained in an input record):

Property	Description
Associated Statement Record	A list of statement records.
Query SQL	SQL for the currently selected statement record.
Name	List of column names in the result set of the currently selected statement record. This list is applicable only if you specified to connect to the database and tested the SQL.
Number	List of column numbers in the result set of the currently selected statement record. ◆ This list is applicable only if you specified to connect to the database and tested the SQL. ◆ This list uses a one-based index, which increments by one.
Nulls Allowed	Whether nulls are permitted in the column selected from the Name list. For informational purposes only.
Clear	Disassociates this field from any query.

Record Tab (Positional)

The following table describes the properties of the Positional File Properties dialog box Record tab:

Property	Description
Record Length	Maximum length (number of bytes) in each record in the positional file.
Record Delimiters	Characters that will be used in the positional file to signify the end of each record. The two fields enable you to type two record delimiters that are used consecutively to terminate a record. The 1 (first) record delimiter field is required. The 2 (second) record delimiter field is optional.

Repeat Tab (EDI and SWIFT)

The following table describes the properties of the EDI and SWIFT Composite Properties dialog box and SWIFT Field Properties dialog box Repeat tab:

Property	Description
Minimum usage	Minimum number of times the map component must loop (repeat). For a conditional loop, the minimum usage is zero. For a mandatory loop, the minimum usage is 1 or greater.
Maximum usage	Maximum number of times the map component must loop (repeat).
Treat this field as a repeating field	If you specified a maximum use of 1 or 0, you can select this check box, which makes this field act as a repeating field. Not used for SWIFT.

Repeating Tab (XML)

The following table describes the properties of the Repeating tab for the XML File Properties, Content Particle Properties, and XML Element Properties dialog boxes:

Property	Description
Conditional	Map component is not required.
Mandatory	Map component must appear in the map.

Property	Description
Can not repeat	Element does not repeat (is a single instance). Not displayed if map component is an XML File.
Can repeat	The element can repeat (loop) as many times as necessary. Not displayed if map component is an XML File.
Can repeat, with a maximum usage	Element can repeat (loop) as many times as is designated in the Maximum usage box. Not displayed if map component is an XML File.
Maximum usage	Number of times the element can repeat (loop). Not displayed if map component is an XML File.

Repeat Suppression (Positional, EDI, Variable-Length Delimited)

The following table describes the properties of the Repeat Suppression tab for the Positional File Properties, EDI File Properties, and Variable-Length Delimited Properties dialog boxes:

Property	Description
Repeat suppression	Supported on both input and output sides of the map for ACH only. When selected, if any field in a repeating segment has a value that is the same as the identical field in the prior segment, the field content is replaced with the specified repeat suppression character.
Repeat suppression character	Only valid if Repeat suppression is selected. Type the repeat suppression character (this is the character which replaces the field content (if the field contains duplicate content as a field in the prior segment)).

Rule Library Dialog Box

The Rule Library dialog box enables you to create, modify, or delete individual rules (functions) from the library.

The following table describes the properties of the Rule Library dialog box:

Property	Description
Library Name	The name of the library. This name will be the name of the library file stored outside Map Editor in the **Sterling Commerce/Map Editor/Extended Rule Library** directory with the .erl file extension appended. **Note:** You cannot use spaces or special characters.
Author	The name of the person who created the extended rule library.
Description	Description of the library.
Version	Version of the rules as they related to the standard.
Rules	Lists the rules already defined, including the rule name and purpose.
Add Rule	Accesses the *Library Rule Dialog Box* on page 316 so you can add a new rule.
Modify Rule	Accesses the *Library Rule Dialog Box* on page 316 for the selected rule so you can modify it.
Delete Rule	Deletes the selected rule.
OK	Saves the function and closes the dialog box.
Cancel	Cancels the function without saving changes.

Rule Library Manager Dialog Box

The Rule Library Manager enables you to create, modify, or delete a library file. You can also add an existing library to the map.

The following table describes the properties of the Rule Library Manager dialog box:

Property	Description
Referenced libraries	Displays the list of extended rule libraries that are available for use in this map. The information displayed includes the name of the libraries and the number of rules they contain.
Add Library	Accesses the Open dialog box so you can import a new extended rule library for use in this map. **Note:** Select the appropriate .erl file from the **Sterling Commerce/Map Editor/Extended Rule Library** directory and click **Open** to load the library.
New Library	Accesses the Rule Library Dialog Box so you can add a new library.
Modify Library	Accesses the Rule Library Dialog Box for the selected library so you can modify it.

Property	Description
Delete Library	Deletes the selected library.
Save	Saves the function and closes the dialog box.
Cancel	Cancels the function without saving changes.

Special Tab (EDI)

The following table describes the properties of the EDI Segment Properties dialog box Special tab:

Property	Description
Normal	This segment is not a wildcard or floating segment.
Wildcard	*Caution:* Use the wildcard function with caution because this segment will pass through the translator without being processed. In this instance, the translator treats the segment as one field, though the segment may contain many fields. For each wildcard segment, you must be sure that the first field is large enough to contain the entire segment (the accumulated length of all the fields in the segment). Define the length for the first field in the segment in the Max Length box on the Field Properties dialog box.
Floating	This segment floats (does not have a fixed position in the transaction set).
Binary	Select this check box if you want to flag this segment as containing binary data. When this option is selected, the Key Field and Wildcard functions are unavailable. *Caution:* For the ANSI X12 841 (Binary Document), the segment must be flagged as containing binary data. **Note:** 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.

Gentran Integration Suite Map Editor Guide

Property	Description
Alternate Tag	An additional segment identification code tag. For example, the translator recognizes an EDI segment as: `<TAG>[Delimiter]<DATA>[Delimiter]<DATA>.<DATA>[Segment Terminator]` The alternate tag for each segment enables the translator to recognize that segment. Specify an alternate tag only if you selected Wildcard under Special Properties, to define a segment that will be recognized by the translator. This tag enables you to create loops that contains wildcard segments and is used as a breaking point for the translator (to break out of the loop to start processing again). This is useful during interchange breaking, when you want to process the beginning and end of a file, but not the middle. You might use wildcards if you have two of the same segments in your map (for example, N1), and you must differentiate the two segments. If one N1 segment will contain Bill To information and the other N1 segment will contain Ship To information, you can use wildcards to differentiate between the functions of the two segments.

SQL Operation Tab (SQL)

The following table describes the properties on the SQL Output Record Properties dialog box SQL Operation tab:

Property	Description
Data Source	List of the associated data sources.
Table/View	List of the database tables for the selected data source.
Insert Update Delete	Operation that the translator performs against the table.
Automatically change and retry operation when needed	If you select this check box and also select: ◆ Insert and a row already exists, the translator performs an update. ◆ Update and a row does not exist, then the translator performs an insert.
On failure, automatically switch selected operation and retry Inserts as Updates or Updates as Inserts	If selected, if the operation is not completed the translator will automatically switch the operation type (that is, update to insert or insert to update) and retry the operation. When the Delete option is selected, this check box is inactive because it does not apply to the delete operation.

SQL Tab (SQL)

The following table describes the properties on the SQL Statement Record Properties dialog box SQL tab:

Property	Description
Data Source	List of all the data sources associated with this side of the map.
SQL Statement	Enables you to type SQL. This box also accepts C-style comments. For example, /*This is a comment*/
Full Screen	Maximizes the SQL statement list.
Returns a Result Set	SQL statement returns results.
SQL statement is a Stored Procedure call	SQL statement is a stored invocation procedure instead of a query or command.
Test SQL	Tests the SQL statement.
Tables/Views	List of database tables and views associated with the selected data source if a schema has been generated for the data source.
Column Info	Column information from the selected database table if a schema has been generated for the data source. The data type displayed is the translator data type of the column, not the database data type.

Standard Formats Tab

The following table describes the properties for the Standard Formats tab on the Preferences dialog box:

Property	Description
Six character dates	The six-character format in which date fields will be interpreted. This default is used when documents are initially loaded from the standard. This default can be overridden on the Field Properties dialog. This parameter is a global option that will change the default date format for all maps. However, the format of the existing date fields will not be changed, and the default is only used for new maps.
	See *Using a Date/Time Field* for an explanation of these data formats.
	Note: This value should be set to correspond to the date formats used in the standard or standards you are using. Most standards use **YYMMDD** as the default, where YY is the two-digit year (last 2 digits), MM is the two-digit month, and DD is the two-digit day.

Property	Description
Eight character dates	The eight-character format in which date fields will be interpreted. This default is used when documents are initially loaded from the standard. This default can be overridden on the Field Properties dialog. This parameter is a global option that will change the default date format for all maps. However, the format of the existing date fields will not be changed, and the default is only used for new maps. See *Using a Date/Time Field* for an explanation of these data formats. **Note:** This value should be set to correspond to the date formats used in the standard or standards you are using. Most standards use **YYYYMMDD** as the default, where YYYY is the four-digit year, MM is the two-digit month, and DD is the two-digit day.
Positive Number Format Default	Specifies the value used at field-level configuration dialog boxes. The default is **Use properties file setting**. Valid choices are: ◆ Use properties file setting (specifies that the rule or rules specified in the customer_overrides.properties file is followed—this is the default) ◆ Don't allow/generate '+' prefix (specifies that regardless of whether the customer_overrides.properties file contains a rule or rules to allow the "+" prefix, the plus sign will not be used for the field) ◆ Allow/generate '+' prefix (specifies that even if the customer_overrides.properties file indicates that the "+" prefix is not allowed in maps, the rule or rules will be overridden and the plus sign will be used for the field)

Standard Rule Tab

The following table indicates where more information is available for the standard rule functions:

For More Information About	See
Select	*Using the Select Standard Rule* on page 171
Update	*Using the Update Standard Rule* on page 180
Use System Variable	*Using the System Variable Standard Rule* on page 155
Use Constant	*Using the Use Constant Standard Rule* on page 156
Use Accumulator	*Using the Use Accumulator Standard Rule* on page 159
Loop Count	*Using the Loop Count Standard Rule* on page 159
Use Code	*Using the Use Code Standard Rule* on page 166

The following table describes the property for the Standard Rule tab:

Property	Description
Standard Rule	A standard rule that will affect this field or element during processing. The rules are mutually exclusive. For property descriptions of each standard rule, see Chapter 8, *Using Standard Rules*.

SWIFT Validation Tab (SWIFT Record)

The following table describes the properties for the SWIFT Validation tab on SWIFT Record Properties dialog box:

Property	Description
Field Syntax	Validation syntax for the field tag as defined by the SWIFT standards (for example, field 73 is 35x['CRLF'35x]0-5). This parameter is automatically populated when you create a map using the SWIFT data dictionary.

Syntax Record Tab

The following table describes the File Properties dialog box Syntax Record tab (for EDI only):

Property	Description
Use syntax record	Select the Use syntax record check box if you want the translator to generate or process a syntax record. A syntax record is used in the EDIFACT (UNA) standard to indicate which control characters (for example, delimiters) that the translator uses to process data.
Tag	Syntax record identifier. This box is active only if the Use syntax record check box is selected.
Position	Syntax record position. This box is active only if the Use syntax record check box is selected.
Length	Length of the syntax record. This box is active only if the Use syntax record check box is selected.

Property	Description
Contains data	Select the Contains Data check box if the syntax record contains data that must be processed. Typically, if a syntax record is received from a EDIFACT partner, the syntax record (UNA) is discarded and processing continues with the interchange.
	This box is active only if the Use syntax record check box is selected.
Release Character Pos	Position in the record where the release character occurs. The release character is used before any defined control character to restore the character used for the control character to its original meaning.
	This box is active only if the Use syntax record check box is selected.
Decimal Separator Pos	The position in the record where the decimal separator occurs. The decimal separator indicates the decimal point in a numeric box, such as a period (.) or comma (,).
	This box is active only if the Use syntax record check box is selected.
Tag (Delimiter Position)	The position in the record where the tag delimiter occurs. The tag delimiter is the character that determines the end of each segment tag.
	This box is active only if the Use syntax record check box is selected.
Segment (Delimiter Position)	Position in the record where the segment delimiter occurs. The segment delimiter is the character that determines the end of each segment.
	This box is active only if the Use syntax record check box is selected.
Element (Delimiter Position)	Position in the record where the element delimiter occurs. The element delimiter is the character that determines the end of each element.
	This box is active only if the Use syntax record check box is selected.
Repeating Element (Delimiter Position)	Position in the record where the repeating element occurs. The repeating element delimiter is the character that determines the start of a new repetition of an element or subelement.
	This box is active only if the Use syntax record check box is selected.
Sub Element (Delimiter Position)	Position in the record where the subelement delimiter occurs. The subelement delimiter is the character that determines the end of each subelement. Subelements are also known as component elements.
	This box is active only if the Use syntax record check box is selected.

Syntax Tokens

The following table describes the properties of the Syntax Tokens dialog box:

Property	Description
Token	Value designated as the syntax token, for each existing Syntax Token.
	The Token contains a range of characters that, when applied to a field/element, dictate the way that field/element must be formatted.

Property	Description
Allowed character ranges	Range of characters that are permitted for each existing Syntax Token.
Close	Closes the Syntax Tokens dialog box.
New	Accesses the Edit Syntax Token dialog box to enable you to create a new syntax token.
Change	Accesses the Edit Syntax Token dialog box to enable you to edit the selected token.
Delete	Deletes the selected syntax token. **Note:** The selected entry will be deleted without warning.
DBCS	Accesses the DBCS dialog box. The DBCS button is only available if you are executing a double-byte version of Windows. This feature enables you to create double-byte syntax tokens.

Tag Tab (CII)

The following table describes the properties on this Tag tab for the CII TFD Properties dialog box:

Property	Description
Tag	The element ID, which is defined by a CII standard.
Hex Decimal	Whether you view the tag in hexadecimal or decimal and applies only to what you see in the Map Editor. This setting does not affect translation.

Tag Tab (EDI)

The following table describes the properties on the Tag Tab for the EDI Segment Properties dialog box:

Property	Description
Tag	Segment identification code tag, as defined by the standards. For example, the translator recognizes an EDI segment as: `<TAG>[Delimiter]<DATA>[Delimiter]<DATA>.<DATA>[Segment Terminator]` The segment tag for each segment enables the translator to recognize that segment. Use the segment tag only if you specified Wildcard under Special Properties, to define a segment that the translator will recognize. This tag enables you to create loops that contains wildcard segments (segments that are not processed) and is used as a breaking point for the translator (to break out of the loop to start processing again). This is useful during interchange breaking, when you want to process the beginning and end of a file, but not the middle.

Tag Tab (Positional)

The following table describes the properties for the Tag tab on Positional Record Properties dialog box:

Property	Description
Tag	Record identification code tag. The record tag for each record enables the translator to recognize that record.
Position	Position of the tag (record identification code) in the record (where the tag begins within the record).
Length	Length of the tag (record identification code) in the record.

Tag Tab (SWIFT Record)

The following table describes the properties for the Tag tab on SWIFT Record Properties dialog box:

Property	Description
Tag	Type the name of the SWIFTNet field tag (for example, 93C).

Tag Tab (VLD)

The following table describes the properties on this tab apply to the Tag tab on the Variable Length Delimited Data Record Properties dialog box:

Property	Description
Tag	String value of the field defined to be the tag, which is used during translation to identify the record.
Field #	Field number that contains the tag from a list of possible field numbers.

Tag Tab (XML)

The following table describes the properties on the Tag tab for the XML File Properties and XML Element Properties dialog boxes:

Property	Description
Tag	Attribute identification tag, as it appears in the XML document. The translator validates the tag against the characters that XML permits for element names. An XML tag must start with a letter, an underscore (_), or a colon (:), followed by valid XML name characters. The translator uses the attribute name by default.
Allow ANY content	If selected, the translator ignores ANY content when processing the map. **Note:** Do not map any child elements, pcdata, or attributes after the element with ANY content specified otherwise the translator will generate an error.
Defined as abstract	Indicates (if selected) that the element is an abstract type. **Note:** This check box must be cleared to enable the map to compile successfully.

Tree Tab

The following table describes the properties for the Tree tab on the Preferences dialog box, which enables you to set global viewing options

Property	Description
These setting customize the look of the file format tree	Enable you to specify whether you want the group, record (segment), or field (element) descriptions displayed.
Colours	Accesses the Colours dialog box to enable you to change colors for map components.
Font	Accesses the Font dialog box to enable you change fonts in the map.

Type Tab (SWIFTNet)

The following table describes the properties on the Type tab for the SWIFTNet Composite Properties dialog box:

Property	Description
Choice (A | B)	Subordinate objects of the composite represent a choice (a disjunction between) the subordinate objects.
Sequence (A , B)	Subordinate objects of the composite represent a sequence of the subordinate objects.

Type Tab, Attributes (XML)

The following table describes the properties on this Tag tab for the XML Attribute Properties dialog box:

Property	Description
Attribute Type	Type of data that can be used in this attribute. This table describes the attribute types.

Attribute	Description
CDATA	Character data (a string of characters).
ENUMERATED	Value must match a value in the associated code list, and all values in the code list must match the NMTOKEN production, as defined by the XML specification. To use an enumerated attribute, you must also create a code list and use a code list standard rule with the attribute.
ID	Valid and unique identifier.
IDREF	Reference to a unique identifier.
IDREFS	List of references to unique identifiers.
NMTOKEN	Value follows the rules specified in XML for name tokens.
NMTOKENS	List of name tokens.

Property	Description
Implied	Attribute is optional. If no value is set, the document is still considered valid.
Required	Attribute is required. If no value is set, the document is not valid.
Default Exists	Default value exists for this attribute. You must define the default value. If the incoming data does not contain a value for this attribute, the translator creates it with the default value.
Fixed	Default value of this attribute is fixed (cannot be changed). You must define the default value. If the incoming data does not match this value, the document is not valid.
Default value	Default value for the attribute.

Type Tab, Content Particles (XML)

The following table describes the properties on the Type tab for the XML Content Particle Properties dialog box:

Property	Description
Choice (A | B)	Subordinate objects of the content particle represent a choice (a disjunction between) the subordinate objects.
Sequence (A , B)	Subordinate objects of the content particle represent a sequence of the subordinate objects.
All (A, B) | (B, A)	Subordinate objects may appear once or not at all, and in any order. **Note:** This parameter appears for XML content particles only.

Validation Tab (EDI Composites and SWIFT Composites)

The following table describes the property on the Validation tab for the Composite Properties dialog box:

Property	Description
Mandatory composite	For EDI composites, whether this composite must be present according to the standard. For SWIFTNet, whether some or all of the SWIFT subfields or components belonging to the composite must exist.

Validation Tab

The following table describes the properties on the Validation tab for the Element Properties, Field Properties, Variable Length Delimited Data Fields, CII TFDs, XML Pcdata, XML Attributes, SWIFT Fields, and SQL Fields:

Property	Description
Mandatory field	Select this check box to indicate that the map component is required. **Note:** For HIPAA, select this check box if the element is required (designated as R in the HIPAA standard) and leave the check box clear if the element is designated as situational (S).
Minimum length	Minimum number of characters in the map component.

Property	Description
Maximum length	Maximum number of characters of the map component.
Data-type	Type of data for this map component. Valid values are: ◆ String - Alphanumeric ◆ Number - Numeric, real, overpunched (not used for SWIFTNet), or packed (for Positional only) ◆ Date/time - Date or time ◆ Bin Data - binary data (only available if you select "Binary" on the Special tab of the EDI Segment Properties dialog box) ◆ Bin Length - length of binary data (only available if you select "Binary" on the Special tab of the EDI Segment Properties dialog box) **Note:** 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.
Data format	How the data in the map component will be formatted. Depending on the data type you selected, you can either: ◆ For the String data-type, select a syntax token to denote that this map component must be formatted as the syntax token dictates. **Note:** Free Format indicates that any characters are acceptable in the field. The translator does not check the characters for compliance. ◆ For Number or Date/Time data-type, select the data format from a list.
Data is Read/written as Raw Bytes	Type of byte ordering: ◆ Little-endian byte ordering specifies that the least significant character is first. Intel and Windows use this order. ◆ Big-endian byte ordering specifies that the most significant character is first. UNIX and Java use this order. **Note:** Not used for SWIFTNet.

Gentran Integration Suite Map Editor Guide

Property	Description
Positive Number Format	How the Map Editor should handle the use and generation of a plus sign. By default, the Positive Number Format section will display the global option that you configured on the Standard Formats tab of the Preferences dialog (see *Standard Formats Tab* on page 336), and you can change the parameter to a setting that will only affect the map component you are currently accessing when the data-type of the map component is Number.
	Note: This parameter is not valid for SQL fields, packed decimal numeric fields, or SWIFTNet fields. This parameter is only enabled if the data-type of the map component is Number and the data format is *not* a packed decimal format.
	Valid choices are:
	◆ Use properties file setting (specifies that the rule or rules configured in the customer_overrides.properties file should be followed for this map component)
	◆ Don't allow/generate '+' prefix (specifies that regardless of whether the Preference dialog box (Standard Formats tab) or the customer_overrides.properties file contains a rule or rules to allow the "+" prefix, the plus sign will not be allowed for this map component)
	◆ Allow/generate '+' prefix (specifies that regardless of whether the Preference dialog box (Standard Formats tab) or the customer_overrides.properties file indicates that the "+" prefix is not allowed in maps, the rule or rules will be overridden and the plus sign will be allowed to be used and generated for this map component when the data-type of the field is Number)

Version Tab

The following table describes the properties for the Version tab on the Preferences dialog box, which enables you to set a global option to specify when map version numbers are incremented:

Note:	You can also set and increment the map version manually on the Map Details dialog box.

Note:	The auto-increment function updates the minor version number of the map on the Map Details dialog box, up to 255. If the minor version number exceeds 255, the system updates the minor version number to zero and increases the major version number.

Property	Description
Only when manually changed	The minor version number of the map is only updated when you do so manually on the Map Details dialog box.
Ask whether to increment when saving a map	When you save a map, you are prompted with a message box asking whether you want to increment the minor version on the Map Details dialog box.
Ask whether to increment when compiling a map	When you compile a map, you are prompted with a message box asking whether you want to increment the minor version on the Map Details dialog box.
Always increment when saving a map	When you save a map the minor version on the Map Details dialog box is incremented.
Always increment when compiling a map	When you compile a map the minor version on the Map Details dialog box is incremented.

Map Editor Migration Information

This section contains information that is useful if you are migrating from Gentran:Server for Windows or Gentran:Server for UNIX to Gentran Integration Suite.

This section covers the following topics:

✦ Map Conversion Utilities

✦ Extended Rules

✦ User Exits

✦ Select and Update Standard Rules

✦ NCPDP Standard

✦ Transaction Data File (TDF)

✦ ODBC

✦ Differences Between the Gentran Integration Suite Translator and the Gentran:Server for Windows Translator

Map Conversion Utilities

Gentran Integration Suite provides you with utilities that enable you to execute the conversion of Gentran:Server for UNIX, Gentran:Server for iSeries, and Gentran:Basic for zSeries maps to the Gentran Integration Suite format from within the Map Editor. The conversion process builds the structure of the two sides of the map, but does not convert links (with the exception of Gentran:Server for UNIX conversions, in which you can specify that simple links are converted), standard and extended rules, and conditional relationships.

You can also translate Gentran:Server for iSeries, and Gentran:Basic for zSeries maps through the command line. For more information about how to use the map conversion utilities or convert maps from the command line, see Appendix F, *Map Conversion*.

Extended Rules

Gentran Integration Suite extended rule support differs slightly from that of Gentran:Server for Windows and Gentran:Server for UNIX. This section contains specific details about the differences and lists the extended rules that are not supported in Gentran Integration Suite.

Caution: If your map contains an incorrect or unsupported extended rule, the translator generates a compile error.

For more information, see:

Wild Blocks

Wild blocks are records in a map that can match any data. They were used in Gentran:Server for Windows to provide pass-through functionality for early build and break maps. Wild blocks are not supported in Gentran Integration Suite because Readblock and Writeblock extended rules are much more efficient.

If you have a map that uses wild blocks, replace them with Readblock and Writeblock extended rules. For more information about using these extended rules, see *readblock* on page 234 and *writeblock* on page 248.

Readbyte and Writebyte

Readbyte and Writebyte extended rules were supported in Gentran:Server for Windows and Gentran:Server for UNIX, but they are not supported in Gentran Integration Suite because the translator does not perform byte-level input and output. To support international character encodings, only character-level input and output is performed.

If you have a map which uses the readbyte and writebyte extended rules, replace them with readblock and writeblock extended rules. For more information about using these extended rules, see *readblock* on page 234 and *writeblock* on page 248.

Fseek and Ftell

Fseek and Ftell extended rules were supported in Gentran:Server for Windows and Gentran:Server for UNIX, but they are not supported in Gentran Integration Suite because Java streams do not provide the necessary functionality.

If you have a map which uses the fseek and ftell extended rules, in most cases (for example, when you want the translator to look for a specified block and then seek back to the start) you can replace the fseek and ftell extended rules with the unreadblock extended rule. See *unreadblock* on page 246 for more information about using this extended rule.

Other Unsupported Extended Rules

If your map contains an extended rule that is a holdover from another Sterling Commerce product and is not supported in Gentran Integration Suite, that rule will cause a compile error in your map. To eliminate the compile error, remove the unsupported extended rule from your map.

The following table contains a list of extended rules that were used in the Gentran:Server for Windows and Gentran:Server for UNIX products, are not supported in Gentran Integration Suite, and the reason the extended rule was not implemented in Gentran Integration Suite:

Extended Rule Not Supported in Gentran Integration Suite	Product this Rule was Supported In	Reason this Extended Rule is not Implemented in Gentran Integration Suite
INSERT	Gentran:Server for Windows	Used with the Gentran database, which is inaccessible from Gentran Integration Suite.
AUDITLOG	Gentran:Server for Windows	Used with the Gentran:Server Audit Log, which is inaccessible from Gentran Integration Suite.
WINEXEC	Gentran:Server for Windows	Called directly to the Win32 WinExec API.
CREATEOBJECT	Gentran:Server for Windows	Required ActiveX support.
DELETEOBJECT	Gentran:Server for Windows	Required ActiveX support.
QUERYOBJECT	Gentran:Server for Windows	Required ActiveX support.
GETIID	Gentran:Server for Windows	Required ActiveX support.
BYTEVECTOR	Gentran:Server for Windows	Required ActiveX support.
EXEC	Gentran:Server for UNIX	Used with the C++ UNIX translator only.
PARAM	Gentran:Server for Windows and Gentran:Server for UNIX	Not required in Gentran Integration Suite.
PARAMCOUNT	Gentran:Server for Windows and Gentran:Server for UNIX	Not required in Gentran Integration Suite.
SETPARAM	Gentran:Server for Windows and Gentran:Server for UNIX	Not required in Gentran Integration Suite.

User Exits

This table contains a matrix of the varying levels of support for user exits across Sterling Commerce product offerings:

Product	User Exit Support
Gentran:Server for Windows	Supports calling ActiveX Automation Servers using Microsoft Component Object Model (COM) technology.
Gentran:Server for UNIX	Supports calling C libraries through a fixed Application Program Interface (API).
Gentran Integration Suite	Supports calling Java objects using Java Reflection APIs.

For more information, see *User Exit Migration Tip* on page 352.

User Exit Migration Tip

If your map contains a user exit, create a Java class that does the same thing as the user exit and then use Gentran Integration Suite extended rule capabilities to call the Java class.

For more information about Gentran Integration Suite extended rules, see Chapter 9, *Using Extended Rules*.

Select and Update Standard Rules

Gentran:Server for Windows supports the use of Select and Update standard rules to provide access to the internal Gentran:Server database. Gentran Integration Suite does support the Select and Update standard rules, but their scope is to provide access to information in the Gentran Integration Suite system; these standard rules cannot provide access to a Gentran database.

Caution: The translator does not process any incorrect standard rules.

For more information about using the Select and Update standard rules in Gentran Integration Suite, see *Using the Select Standard Rule* on page 171 and *Using the Update Standard Rule* on page 180.

For more information, see:

Tracking Migration Tip

If you are migrating from Gentran:Server for Windows to Gentran Integration Suite, and your maps use Select and Update standard rules which refer to columns of tables in the Gentran:Server database for Tracking purposes, you can replace the old standard rules with a corresponding standard rule, selecting either Process Data or Correlation table as the table to be updated. For more information about using the Select and Update standard rules in Gentran Integration Suite, see *Using the Select Standard Rule* on page 171 and *Using the Update Standard Rule* on page 180.

Trading Partner Code List Migration Tip

If you are migrating from Gentran:Server for Windows to Gentran Integration Suite, and your maps use a Select standard rule to get information from Trading Partner Code Lists, you can manually create a new code list in Gentran Integration Suite and then replace the Select standard rule with one that does the select by Sender ID or Receiver ID.

For more information about using the Select standard rule in Gentran Integration Suite, see *Using the Select Standard Rule* on page 171.

Document Name Migration Tip

If you are migrating from Gentran:Server for Windows to Gentran Integration Suite and your maps use an Update standard rule to update the database with the document name, you can modify the Update standard rule to update the Correlation table with the document name, so you can then search the Correlation Search interface for the name you specified.

For more information about using the Update standard rule in Gentran Integration Suite, see *Using the Update Standard Rule* on page 180.

Synonym Table by In Value

If you are migrating from Gentran:Server for UNIX to Gentran Integration Suite and your maps use a Select standard rule (Synonym Table by In Value) to access the Gentran:Server

for UNIX synonym table and search a specified table column ("Out") to match the data in the current map component. When a match is found, the translator returns the contents of that synonym table row into the field you specify.

In Gentran:Server for UNIX, this function enables you to configure cross-referencing so you can convert your values (for example, a part number) to your trading partner's values during outbound processing, and to convert your trading partner's values to your values during inbound processing.

> **Note:**   This rule is not supported in Gentran Integration Suite—it enables easier migration of maps from Gentran:Server for UNIX to Gentran Integration Suite by enabling you to view the rule in the your maps.

For more information about using the Select standard rule in Gentran Integration Suite, see *Using the Select Standard Rule* on page 171.

Synonym Table by Out Value

If you are migrating from Gentran:Server for UNIX to Gentran Integration Suite and your maps use a Select standard rule (Synonym Table by Out Value) to access the Gentran:Server for UNIX synonym table and search a specified table column ("In") to match the data in the current map component. When a match is found, the translator returns the contents of that synonym table row into the field you specify.

In Gentran:Server for UNIX, this function enables you to configure cross-referencing so you can convert your values (for example, a part number) to your trading partner's values during outbound processing, and to convert your trading partner's values to your values during inbound processing.

> **Note:**   This rule is not supported in Gentran Integration Suite—it enables easier migration of maps from Gentran:Server for UNIX to Gentran Integration Suite by enabling you to view the rule in the your maps.

For more information about using the Select standard rule in Gentran Integration Suite, see *Using the Select Standard Rule* on page 171.

NCPDP Standard

NCPDP is the National Council of Prescription Drug Programs standard for the electronic exchange of documents. NCPDP is supported in the Gentran:Server for Windows and Gentran:Server for UNIX products but Gentran Integration Suite does *not* support the NCPDP standard.

Transaction Data File (TDF)

TDF is a proprietary Sterling Commerce data format originally used by Gentran:Basic for DOS. The transaction data file (TDF) button is disabled in Gentran Integration Suite. TDF is supported in the Gentran:Server for Windows product. The TDF format is not supported in Gentran Integration Suite.

ODBC

If you were using ODBC maps with Gentran:Server for Windows, to convert the maps to use with Gentran Integration Suite you must recompile the maps in the Map Editor on a machine that has the necessary ODBC data source.

Also, you must have a JDBC driver installed in Gentran Integration Suite for the database and set up a pool in Gentran Integration Suite with the pool name matching the name of the ODBC data source. At runtime, Gentran Integration Suite accesses the database using the pool you configured.

Differences Between the Gentran Integration Suite Translator and the Gentran:Server for Windows Translator

There are inherent differences between the Gentran Integration Suite translator and the Gentran:Server for Windows translator. In both Gentran Integration Suite and Gentran:Server for Windows, you can specify a Record Length and/or Record Delimiters on the Positional File Properties dialog box (Record tab). The Record Length and Record Delimiters parameters notify the translator how many bytes should be read from the input stream to create one positional record. These parameters are not mutually exclusive.

It is important to note that the Gentran Integration Suite and Gentran:Server for Windows translators may not treat the Record Length and Record Delimiters parameters in the same manner, depending on which parameters you specify.

The following table describes the differences between the two translators and how they handle the Record Length and Record Delimiters parameters:

Parameter(s) specified on Positional Record Properties dialog box (Records tab)	How the Gentran Integration Suite translator handles the specified information	How the Gentran:Server for Windows translator handles the specified information
Record Length *only*	The translator attempts to read Record Length bytes from the input stream and ignores any operating system default line separators, stopping only when it encounters the end of the input stream.	The translator attempts to read Record Length bytes from the input stream but stops if it encounters the operating system default line separators or the end of the input stream.

Parameter(s) specified on Positional Record Properties dialog box (Records tab)	How the Gentran Integration Suite translator handles the specified information	How the Gentran:Server for Windows translator handles the specified information
Record Length *and* Record Delimiters	The translator attempts to read up to the Record Length bytes from the input stream but stops if it encounters the specified Record Delimiters or the end of the input stream. The Record Delimiters take precedence over the Record Length parameter.	The translator attempts to read up to the Record Length bytes from the input stream but stops if it encounters the specified Record Delimiters or the end of the input stream. The Record Delimiters take precedence over the Record Length parameter.
Record Delimiters *only*	The translator continues to read from the input stream until it either encounters the Record Delimiters or the end of the input stream.	The translator continues to read from the input stream until it either encounters the Record Delimiters or the end of the input stream.

Using Indexes in the Map Editor and Translator

This section contains detailed information about how to use extended rule indexes in the Map Editor and translator and uses detailed examples to illustrate the proper use of indexes.

See Chapter 9, *Using Extended Rules* for more information about extended rules.

This section covers the following topics:

✦ Introduction to Indexes

✦ Simple Example of Using Indexes

✦ Complex Example of Using Indexes

✦ Using Indexes with an XML File Format

Introduction to Indexes

To write valid extended rules, you must understand how to use indexes. Without indexes, it is possible that a rule which moves data from one field to another may result in lost data.

An index is a number that is used within an extended rule only when referencing a field contained by a repeating record or group. This number is used to specify the exact occurrence (iteration) of the repeating group or record that you want the translator to access.

> **Note:** When you use indexes in the Map Editor, you must always initialize them. The translator does not initialize any variables defined in a map.

An index is a valuable part of a Fully Qualified Reference format (FQR format). When you write an extended rule that references a field, you must clarify for the translator exactly which field you need it to access. The FQR format helps to remove ambiguity and enables the author of the extended rule to be extremely explicit about where the translator can find the field they are referencing. The following figure shows the FQR format:

$GroupName[index1][index2].#FieldName
```
   1              2                3
```

The FQR format defines how a field must be referenced within an extended rule and provides three important pieces of information to the translator:

1. **Group Name** – The name of the repeating map component that contains the field. In a nested group structure, this is the name of the innermost repeating group or record containing the field.

2. **Index** – An index consists of a number enclosed in square brackets. The number represents the specific occurrence of data you want to access within the repeating group or record. If the field to be accessed is in a nested group structure, you must provide an index for each level of nesting.

3. **Field Name** – The name of the field containing the data to be accessed.

As an analogy, you can think of specifying an index in terms of choosing a floor number in an elevator—there are many floors in a building, and the elevator does not know where you intend to go, unless you tell it by pushing the button for the appropriate floor number. In a similar way, you can think of a repeating record as a building with many floors—each floor is a different occurrence of data in that repeating record. The translator must be informed which occurrence of the record to access to retrieve the appropriate data in the same way you must indicate which floor you intend to visit to an elevator. The index is analogous to the floor number.

For example, you create a map to process inbound orders and you want to write an extended rule to save the shipping address into a string variable. This map uses a field named **CompanyAddress**, which is contained by a repeating record named **CompanyInfo** that can repeat up to ten times. You know that the third occurrence of the **CompanyInfo** record always contains the shipping address for the parts order. Therefore, the extended rule you write using the FQR format follows:

```
string [64] strAddress;
strAddress = $CompanyInfo[3].#CompanyAddress;
```

Sometimes it is appropriate to use an Abbreviated Reference Format (ABR format) when accessing a field:

```
#FieldName
   |_____|
        1
```

The ABR format defines how a field must be referenced within an extended rule and provides only one piece of information to the translator:

✦ **Field Name** – The name of the field containing the data to be accessed.

The ABR format consists of a field name only. The ABR format must only be used when you write an extended rule at the field level that references a field within the current record. An example of an extended rule using the ABR format follows:

```
#TempAddress = #CompanyAddress;
```

The following table lists many scenarios in which you would use an extended rule to reference a field, and notes whether the FQR format must be used in that particular situation:

Rule Level	Map Object	Use FQR Format
Field	Referencing current field or another field within the current record	No
Field	Referencing a field outside the current record	Yes
Record	Referencing a field within the current record	No
Record	Referencing a field outside the current record	Yes
Group	Referencing a field within the current group	Yes
Group	Referencing a field outside the current group	Yes
Session	Referencing a field anywhere in the map	Yes

For more information, see *Background Information* on page 360.

Background Information

The translator works in the most efficient manner possible. When the translator is processing the Output side of the map and encounters a repeating structure (record or group) that does not contain any data, it detects no further data for that repeating structure and moves on to the next map component. If the translator did not function in this manner, it would have to read through the maximum number of loops permitted for that repeating structure, which would lengthen the time necessary to process maps.

When the translator is processing a map, it creates a data storage area based on the structure of the Input side of the map (the *source* side of the map), because the Input side of the map is processed first. Therefore, extended rules address the map based on the hierarchy of the Input side. When the end of the Input structure is reached, the translator processes the Output side of the map.

When you use extended rules, you must be careful to always address the Input side of the map so the translator can locate the map object that the rule accesses. From the Input side of a map, you have access to the entire file structure. From the Output side of a map, extended rules only have access to the current record (on the Output side) and the entire Input side of the map.

Simple Example of Using Indexes

In this example, you are receiving a repeating segment that contains notes (the **NTE** segment). You must map the contents of the **Notes** element (the **NTE02** (**0003**) element) to a repeating Output record (**Notes**), but only if they have a qualifier of Y.

Following is a sample of the repeating segment being received in the input file:

```
NTE*Y*Must include this line in the output file~
NTE*N*Do not need this line in the output file~
NTE*N*Do not need this line in the output file~
NTE*Y*Must include this line in the output file~
NTE*Y*Must include this line in the output file~
```

In the map, you have created a temporary input repeating segment (**TEMP_NTE**) to receive the notes for only those segments that have a qualifier of Y in the **NTE01** (**0363**) element. Then you add a simple link from the **Notes** element in the **TEMP_NTE** segment to the repeating Output record (**Notes**), as displayed in the following figure:
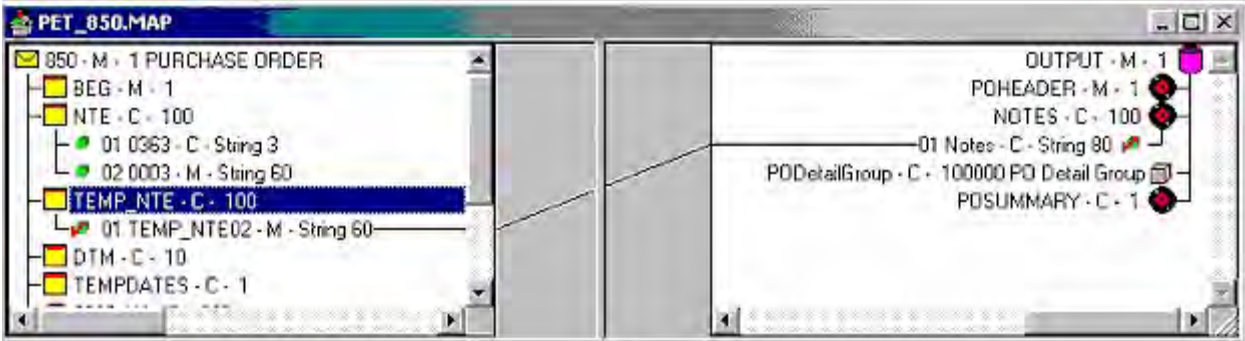
For more information, see:

✦ *Extended Rule without Indexes (Simple Example)* on page 361

✦ *Adding Indexes (Simple Example)* on page 363

✦ *Translation with Indexes (Simple Example)* on page 364

Extended Rule without Indexes (Simple Example)

Add the following extended rule to the On End Extended rule for the **NTE** segment:

```
IF #0363 = "Y" then
Begin
  $TEMP_NTE.#TEMP_NTE02 = #0003
End
```

After running translation, the output file contains the following data for the Notes segment:

```
Notes Must include this line in the output file
```

There is only one segment that contains the required data, but the input file contains three segments that must be written to the Output file. The Output file only contains one segment because the extended rule did not make use of an index to track which specific occurrence of the temporary segment to write the data.

Arrays

A repeating structure is represented internally (in the translator) as an array. An *array* groups a number of items into a larger unit. The items in an array are accessed by an index number. When you place data into an array, you use an index to make sure that each consecutive occurrence of an array is populated, and thus there will never be an instance of the array that does not contain data, and this prevents data loss.

Translation

The following is a step-by-step explanation of how the translator processed the map using the preceding rule.

1. The translator initiates the **Input** side of the map.

   a. The **first NTE** segment is processed.

   > **Note:** The translator reads the first occurrence of the **NTE** segment. The rule finds the qualifier Y in the **NTE01** element, so it maps the contents of the **NTE02** element to the first occurrence of the **TEMP_NTE** segment.

   b. The **second NTE** segment is processed.

   > **Note:** The translator is reading the second occurrence of the **NTE** segment. The rule does not find the qualifier of Y, so the translator does not write any data to the second occurrence of the **TEMP_NTE** segment.

   c. The **third NTE** segment is processed.

   > **Note:** The translator is reading the third occurrence of the **NTE** segment. The rule does not find the qualifier of Y, so the translator does not write any data to the third occurrence of the **TEMP_NTE** segment.

   d. The **fourth NTE** segment is processed.

   > **Note:** The translator reads the fourth occurrence of the **NTE** segment. The rule finds the qualifier Y in the **NTE01** element, so it maps the contents of the **NTE02** element to the fourth occurrence of the **TEMP_NTE** segment.

   e. The **fifth NTE** segment is processed.

   > **Note:** The translator reads the fifth occurrence of the **NTE** segment. The rule finds the qualifier Y in the **NTE01** element, so it maps the contents of the **NTE02** element to the fifth occurrence of the **TEMP_NTE** segment.

2. The translator initiates the **Output** side of the map.

   a. The translator processes the **first** occurrence of the **TEMP_NTE** segment to verify whether there is any data to be linked across to the Output side of map.

   > **Note:** The **first** occurrence of the **TEMP_NTE** segment contains data, so the translator passes that data across to the **Output** segment.

b. The translator processes the **second** occurrence of the **TEMP_NTE** segment to verify whether there is any data to be linked across.

> **Note:** There is no data in the **second** occurrence of the **TEMP_NTE** segment, so the translator is finished processing the TEMP_NTE segment and moves to the next segment to process it.

The following table is an example of the array the translator would create for the **TEMP_NTE** segment using the extended rule that does *not* use indexes:

Occurrence	Data Generated
1	Must include this line in the output file.
2	
3	
4	Must include this line in the output file.
5	Must include this line in the output file.

The following table demonstrates that an extended rule must use an index so that the data is written to each occurrence of the array, consecutively:

Occurrence	Data Generated
1	Must include this line in the output file.
2	Must include this line in the output file.
3	Must include this line in the output file.

Adding Indexes (Simple Example)

To receive the proper output, you must modify the extended rule. The following example contains the extended rule, which has been modified to use indexes:

```
If #0363 = "Y" then
Begin
  $TEMP_NTE[nte_cnt].#TEMP_NTE02 = #0003
  nte_cnt = nte_cnt + 1;
End
```

The preceding extended rule consists of the following parts:

✦ The index (**nte_cnt**) follows the name of the repeating structure (**TEMP_NTE**) and is enclosed in brackets.

✦ In the preceding extended rule, **nte_cnt** is the index, and it was declared and initialized on the Input On Begin extended rule, using the following declaration and initialization rule:

```
Integer nte_cnt;
nte_cnt = 1;
```

✦ The index is declared on the Input (file format) level because it must be available to be used on the **NTE** segment. You cannot declare the index on the **NTE** segment itself, because that would cause the translator to re-declare the index each time an **NTE** segment is received.

✦ By declaring it on the Input level, the index is ready for use and in scope for the entire Input side of the map.

✦ The rule adds one to the index each time the **If** statement condition is met, so it does not create empty occurrences of the **TEMP_NTE** segment. When the map is processed with the preceding rule, the correct output is received.

Translation with Indexes (Simple Example)

This section explains how the translator processes the map using the preceding extended rule:

1. The translator initiates the input side of the map.

    a. The **first NTE** segment is processed as follows:

        • The **nte_cnt** index is currently set to 1.

        • The rule found the qualifier Y in the **NTE01**. The value in the **nte_cnt** index is 1, so the data is written to the **first** occurrence of the **TEMP_NTE** segment.

        • The index is then incremented by one.

    b. The **second NTE** segment is processed as follows:

        • The **nte_cnt** index is currently set to 2.

        • The rule did not find a Y qualifier, so the translator does not write anything to the **second** occurrence of the **TEMP_NTE** segment.

        • The index is *not* incremented because the statement was not processed.

    c. The **third NTE** segment is processed as follows:

        • The **nte_cnt** index is currently set to 2.

        • The rule did not find a Y qualifier, so the translator does not write anything to the **third** occurrence of the **TEMP_NTE** segment.

        • The index is *not* incremented because the statement was not processed.

    d. The **fourth NTE** segment is processed as follows:

        • The **nte_cnt** index is currently set to 2.

        • The rule found the qualifier Y in the **NTE01**. The value in the **nte_cnt** index is 2, so the data is written to the **second** occurrence of the **TEMP_NTE** segment.

        • The index is then incremented by one.

      e.   The **fifth NTE** segment is processed as follows:

- The **nte_cnt** index is currently set to 3.
- The rule found the qualifier Y in the **NTE01**. The value in the **nte_cnt** index is 3, so the data is written to the **third** occurrence of the **TEMP_NTE** segment.
- The index is then incremented by one.

2. The translator initiates the output side of the map.

      a.   The translator processes the **first** occurrence of the **TEMP_NTE** segment to verify whether there is any data to be linked across to the Output side of map.

> **Note:** The **first** occurrence of the **TEMP_NTE segment** contains data, so the translator passes that data across to the **Output** segment.

      b.   The translator processes the **second** occurrence of the **TEMP_NTE** segment to verify whether there is any data to be linked across to the Output side of map.

> **Note:** The **second** occurrence of the **TEMP_NTE** segment contains data, so the translator passes that data across to the **Output** segment.

      c.   The translator processes the **third** occurrence of the **TEMP_NTE** segment to verify whether there is any data to be linked across to the Output side of map.

> **Note:** The **third** occurrence of the **TEMP_NTE** segment contains data, so the translator passes that data across to the **Output** segment.

      d.   The translator processes the **fourth** occurrence of the **TEMP_NTE** segment to verify whether there is any data to be linked across to the Output side of map.

> **Note:** The **fourth** occurrence of the **TEMP_NTE** segment does not contain data, so the translator is finished processing the TEMP_NTE segment.

Complex Example of Using Indexes

In this example, on the Input side of the map you receive an **SAC** segment that repeats 25 times. The SAC segment is a child to a repeating **SLN** group. The SLN group is a child to a repeating **IT1** group, which puts the SAC segment three hierarchical levels deep on the Input side of the map.

Additionally, you only want to map the contents of the **SAC** segment *if* the **SAC05** (Price) element is *not* equal to zero. Finally, the Output structure of the map is at the same hierarchical level as the input structure.

Following is a sample of the repeating structure that you receive in the input file:

```
IT1*1*****SH*Service is Requested
SLN*1**O
SAC*S**AB**0****EA*0***Service**Service
SAC*S**AB**500.00****EA*20***Service**Service
SAC*S**AB**0****EA*23***Service**Service
SAC*S**AB**121.11****EA*10***Service**Service
SLN*2**O
SAC*S**AB**0****EA*50***Service**Service
SAC*S**AB**250****EA*2***Service**Service
IT1*2*****SH*Service is Requested
SLN*1**O
SAC*S**AB**0****EA*24***Service
SAC*S**AB**45.67****EA*200***Service
SAC*S**AB**0****EA*78***Service**Service
SAC*S**AB**222.74****EA*103.789***Service
```

On the Input side of the map, you have created a temporary repeating segment (**TEMP_SAC**) to receive the SAC information for those segments where the Price is not equal to zero, and you have linked the **TEMP_SAC** segment to the Output side of the map.

Now, you want the translator to create the following array:

Loop Hierarchical Structure	Occurrence	Data
First IT1, First SLN, First TEMP_SAC	1,1,1	SAC*S**AB**500.00****EA*20***Service**Service
First IT1, First SLN, Second TEMP_SAC	1,1,2	SAC*S**AB**121.11****EA*10***Service**Service
First IT1, Second SLN, First TEMP_SAC	1,2,1	SAC*S**AB**250****EA*2***Service**Service
Second IT1, First SLN, First TEMP_SAC	2,1,1	SAC*S**AB**45.67****EA*200***Service
Second IT1, First SLN, Second TEMP_SAC	2,1,2	SAC*S**AB**222.74****EA*103.789***Service

For more information, see:

Map Setup (Complex Example)

The following figure indicates where the SAC segment is located in the map:
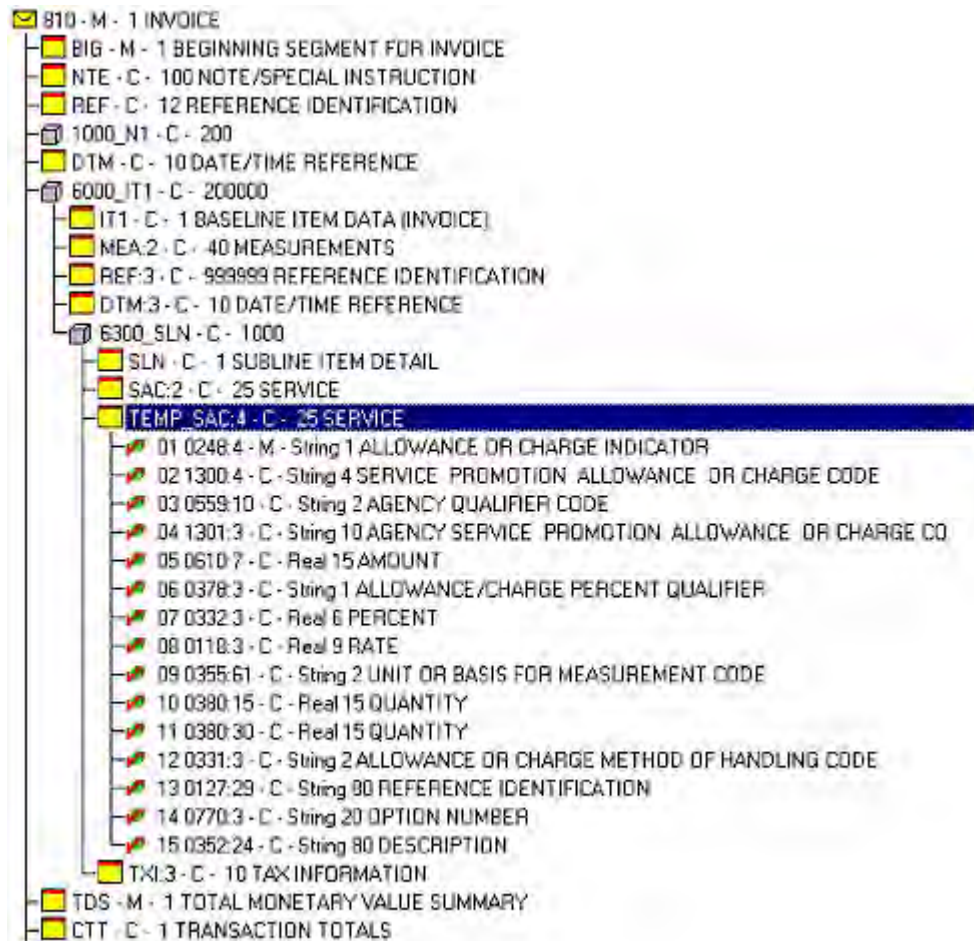
Extended Rule without Indexes (Complex Example)

Add the following On End extended rule to the **SAC** segment:

```
IF #0610:2 != 0 THEN
BEGIN
$TEMP_SAC:4.#0248:4 = #0248:3;
$TEMP_SAC:4.#1300:4 = #1300:3;
$TEMP_SAC:4.#0559:10 = #0559:9;
$TEMP_SAC:4.#1301:3 = #1301:2;
$TEMP_SAC:4.#0610:7 = #0610:2;
$TEMP_SAC:4.#0378:3 = #0378:2;
$TEMP_SAC:4.#0332:3 = #0332:2;
$TEMP_SAC:4.#0118:3 = #0118:2;
$TEMP_SAC:4.#0355:61 = #0355:54536;
$TEMP_SAC:4.#0380:15 = #0380:84;
$TEMP_SAC:4.#0380:30 = #0380:168;
$TEMP_SAC:4.#0331:3 = #0331:2;
$TEMP_SAC:4.#0127:29 = #0127:571;
$TEMP_SAC:4.#0770:3 = #0770:2;
$TEMP_SAC:4.#0352:24 = #0352:19;
END
```

Now, after the translator runs, the Output file contains no data from the SAC segment, but there are five valid SAC segments in the input file. The reason the Output file does not contain the SAC segment information is because the extended rule did not make use of indexes to track the specific occurrence of the **TEMP_SAC** segment to which it must write the data.

Arrays

A repeating structure is represented internally (in the translator) as an array. An *array* groups a number of items into a larger unit. The items in an array are accessed by an index number. When you place data into an array, you use an index to make sure that each consecutive occurrence of an array is populated, and thus there will never be an instance of the array that does not contain data, and this prevents data loss.

Translation

The following is a step-by-step explanation of how the translator processed the map using the preceding rule:

1. The translator initiates the **Input** side of the map.
2. The **first IT1** segment is processed.
   a. The **first SLN** segment is processed.
   b. The first SAC segment is processed as follows:
      - The **If** statement is evaluated false because the value in the **SAC05** element is equal to zero.
      - The translator is on the **first** occurrence of the **SAC** segment, so it does not write any data to the **first** occurrence of the **TEMP_SAC** segment.

    c. The **second SAC** segment is processed as follows:

- The **If** statement is evaluated to true because the value in the **SAC05** is equal to 500.00 (more than zero).
- The translator is on the **second** occurrence of the **SAC** segment, so it writes the contents of the **SAC** segment to the **second** occurrence of the **TEMP_SAC** segment.

    d. The **third SAC** segment is processed as follows:

- The **If** statement is evaluated false because the value in the **SAC05** element is equal to zero.
- The translator is on the **third** occurrence of the **SAC** segment, so it does not write any data to the **third** occurrence of the **TEMP_SAC** segment.

    e. The **fourth SAC** segment is processed as follows:

- The **If** statement is evaluated to true because the value in the **SAC05** is equal to 112.11.
- The translator is on the **fourth** occurrence of the **SAC** segment, so it writes the contents of the **SAC** segment to the **fourth** occurrence of the **TEMP_SAC** segment.

    f. The **second SLN** segment is processed.

    g. The **first SAC** segment is processed as follows:

- The **If** statement is evaluated false because the value in the **SAC05** element is equal to zero.
- The translator is on the **first** occurrence of the **SAC** segment, so it does not write any data to the **first** occurrence of the **TEMP_SAC** segment.

    h. The **second SAC** segment is processed as follows:

- The **If** statement is evaluated to true because the value in the **SAC05** is equal to 250.
- The translator is on the **second** occurrence of the **SAC** segment, so it writes the contents of the **SAC** segment to the **second** occurrence of the **TEMP_SAC** segment.

3. The second **IT1** segment is processed.

    a. The first **SLN** segment is processed.

    b. The first **SAC** segment is processed as follows:

- The **If** statement is evaluated false because the value in the **SAC05** element is equal to zero.
- The translator is on the **first** occurrence of the **SAC** segment, so it does not write any data to the **first** occurrence of the **TEMP_SAC** segment.

    c. The **second SAC** segment is processed as follows:

- The **If** statement is evaluated to true because the value in the **SAC05** is equal to 45.67.
- The translator is on the **second** occurrence of the **SAC** segment, so it writes the contents of the **SAC** segment to the **second** occurrence of the **TEMP_SAC** segment.

d.   The **third SAC** segment is processed as follows:

- The **If** statement is evaluated false because the value in the **SAC05** element is equal to zero.

- The translator is on the **third** occurrence of the **SAC** segment, so it does not write any data to the **third** occurrence of the **TEMP_SAC** segment.

e.   The **fourth SAC** segment is processed as follows:

- The **If** statement is evaluated to true because the value in the **SAC05** is equal to 222.74.

- The translator is on the **fourth** occurrence of the **SAC** segment, so it writes the contents of the **SAC** segment to the **fourth** occurrence of the **TEMP_SAC** segment.

4.   The translator initiates the **Output** side of the map.

5.   The translator processes the **first** occurrence of the **TEMP_SAC** segment in the **first SLN** group of the **first IT1** group to verify whether there is any data to be linked across to the Output side of the map.

- There is no data in the first occurrence of the **TEMP_SAC** segment so the translator is finished processing the TEMP_SAC segment and moves on to the next segment.

6.   The translator processes the **first** occurrence of the **TEMP_SAC** segment in the **second SLN** group of the **first IT1** group to verify whether there is any data to be linked across to the Output side of the map.

- There is no data in the first occurrence of the **TEMP_SAC** segment so the translator is finished processing the TEMP_SAC segment and moves on to the next segment.

7.   The translator processes the **first** occurrence of the **TEMP_SAC** segment in the **first SLN** group of the **second IT1** group to verify whether there is any data to be linked across to the Output side of the map.

- There is no data in the first occurrence of the **TEMP_SAC** segment so the translator is finished processing the TEMP_SAC segment and moves on to the next segment.

The following table is an example of the array the translator will create for the **TEMP_SAC** segment using the extended rule that does not use indexes:

Loop Hierarchical Structure	Occurrence	Data
First IT1, First SLN, First TEMP_SAC	1,1,1	
First IT1, First SLN, Second TEMP_SAC	1,1,2	SAC*S**AB**500.00****EA*20***Service**Service
First IT1, First SLN, Third TEMP_SAC	1,1,3	
First IT1, First SLN, Fourth TEMP_SAC	1,1,4	SAC*S**AB**121.11****EA*10***Service**Service
First IT1, Second SLN, First TEMP_SAC	1,2,1	

Loop Hierarchical Structure	Occurrence	Data
First IT1, Second SLN, Second TEMP_SAC	1,2,2	SAC*S**AB**250****EA*2***Service**Service
Second IT1, First SLN, First TEMP_SAC	2,1,1	
Second IT1, First SLN, Second TEMP_SAC	2,1,2	SAC*S**AB**45.67****EA*200***Service
Second IT1, First SLN, Third TEMP_SAC	2,1,3	
Second IT1, First SLN, Fourth TEMP_SAC	2,1,4	SAC*S**AB**222.74****EA*103.789***Service

Adding Indexes (Complex Example)

To receive the proper output, you must modify the extended rule. You declared and initialized the index Z on the Input On Begin extended rules to track which occurrence of the TEMP_SAC segment should have data written to it.The following example contains the the extended rule on the SAC On End rules has been modified to utilize the Z index:

```
IF #0610:2 != 0 THEN
BEGIN
$TEMP_SAC:4[Z].#0248:4 = #0248:3;
$TEMP_SAC:4[Z].#1300:4 = #1300:3;
$TEMP_SAC:4[Z].#0559:10 = #0559:9;
$TEMP_SAC:4[Z].#1301:3 = #1301:2;
$TEMP_SAC:4[Z].#0610:7 = #0610:2;
$TEMP_SAC:4[Z].#0378:3 = #0378:2;
$TEMP_SAC:4[Z].#0332:3 = #0332:2;
$TEMP_SAC:4[Z].#0118:3 = #0118:2;
$TEMP_SAC:4[Z].#0355:61 = #0355:54536;
$TEMP_SAC:4[Z].#0380:15 = #0380:84;
$TEMP_SAC:4[Z].#0380:30 = #0380:168;
$TEMP_SAC:4[Z].#0331:3 = #0331:2;
$TEMP_SAC:4[Z].#0127:29 = #0127:571;
$TEMP_SAC:4[Z].#0770:3 = #0770:2;
$TEMP_SAC:4[Z].#0352:24 = #0352:19;
Z = Z + 1;
END
```

However, when you compile the preceding extended rule, you receive the following error for several lines of the rule: *insufficient indexes to access group TEMP_SAC:4*. You receive this error because the **TEMP_SAC** segment is a child to the **SLN** group, and the SLN group is a child to the **IT1** group, and thus you need an index for each group. In this case, because the **IT1** and **SLN** groups are repeating groups, the rule cannot use an index of 1 to reference them, as noted in the following incorrect example:

```
$TEMP_SAC:4[1][1][Z].#0248:4 = #0248:3;
```

Using an index of 1 in this instance will only enable the translator to access the first occurrence of the **IT1** and **SLN** groups. We need the indexes for the **IT1** group and the **SLN** group to keep track of which occurrence the translator is currently processing for both groups.

The following example contains the extended rule, which has been modified to use an index for each group (**IT1**, **SLN**, and **TEMP_SAC**):

```
IF #0610:2 != 0 THEN
BEGIN
  $TEMP_SAC:4[X][Y][Z].#0248:4 = #0248:3;
  $TEMP_SAC:4[X][Y][Z].#1300:4 = #1300:3;
  $TEMP_SAC:4[X][Y][Z].#0559:10 = #0559:9;
  $TEMP_SAC:4[X][Y][Z].#1301:3 = #1301:2;
  $TEMP_SAC:4[X][Y][Z].#0610:7 = #0610:2;
  $TEMP_SAC:4[X][Y][Z].#0378:3 = #0378:2;
  $TEMP_SAC:4[X][Y][Z].#0332:3 = #0332:2;
  $TEMP_SAC:4[X][Y][Z].#0118:3 = #0118:2;
  $TEMP_SAC:4[X][Y][Z].#0355:61 = #0355:54536;
  $TEMP_SAC:4[X][Y][Z].#0380:15 = #0380:84;
  $TEMP_SAC:4[X][Y][Z].#0380:30 = #0380:168;
  $TEMP_SAC:4[X][Y][Z].#0331:3 = #0331:2;
  $TEMP_SAC:4[X][Y][Z].#0127:29 = #0127:571;
  $TEMP_SAC:4[X][Y][Z].#0770:3 = #0770:2;
  $TEMP_SAC:4[X][Y][Z].#0352:24 = #0352:19;
Z = Z + 1;
END
```

The extended rule consists of the following parts:

✦ In the preceding extended rule, the **X** index is used to keep track of the **IT1** group and the **Y** index is used to keep track of the **SLN** group. Both these indexes were declared and initialized to 1 on the Input level On Begin rule.

✦ Also, you have placed an On End extended rule on the **IT1** group to add 1 to the **X** index. The **X** index must be incremented to account for when the translator receives a new **IT1** group. For example:

```
X = X + 1;
```

✦ You have also placed an On End extended rule on the **SLN** group to add 1 to the **Y** index. The **Y** index must be incremented to account for when the translator receives a new SLN group. For example:

```
Y = Y + 1;
```

✦ The extended rule on the On End of the **SAC** segment adds 1 to the **Z** index only when the **If** statement is evaluated as true, so it is not creating empty occurrences of the **TEMP_SAC** segment.

However, when you process the map with the preceding rule, you still do not receive the correct output. This time you do receive data from the **SAC** segment, but you only receive the first two **SAC** segments in which the **SAC05** element does not equal zero (that is, in which the **If** statement is evaluated as true).

The following table is an example of the array the translator will create for the **TEMP_SAC** segment using the index configuration described previously:

Loop Hierarchical Structure	Occurrence	Data
First IT1, First SLN, First TEMP_SAC	1,1,1	SAC*S**AB**500.00****EA*20***Service**Service
First IT1, First SLN, Second TEMP_SAC	1,1,2	SAC*S**AB**121.11****EA*10***Service**Service
First IT1, Second SLN, First TEMP_SAC	1,2,3	SAC*S**AB**250****EA*2***Service**Service
Second IT1, First SLN, First TEMP_SAC	2,3,4	SAC*S**AB**45.67****EA*200***Service
Second IT1, First SLN, Second TEMP_SAC	2,3,5	SAC*S**AB**222.74****EA*103.789***Service

Translation with Incomplete Indexes (Complex Example)

This section explains how the translator would process the Output side of the map with the data in the previous table:

1. The translator initiates the Output side of the map.

   a. The translator processes the first occurrence of the **TEMP_SAC** segment in the first **SLN** group of the first **IT1** group to verify whether there is any data to be linked across to the Output side of the map.

      • The first occurrence of the **TEMP_SAC** segment does contain data, so the translator links this information to the Output side.

      • The extended rule wrote the data to the proper occurrence in the array (1,1,1), as noted in the preceding table.

   b. The translator processes the second occurrence of the **TEMP_SAC** segment in the first **SLN** group of the first **IT1** group to verify whether there is any data to be linked across to the Output side of the map.

      • The second occurrence of the **TEMP_SAC** segment does contain data, so the translator links this information to the Output side.

      • The extended rule wrote the data to the proper occurrence in the array (1,1,2), as noted in the preceding table.

   c. The translator processes the third occurrence of the **TEMP_SAC** segment in the first **SLN** group of the first **IT1** group to verify whether there is any data to be linked across to the Output side of the map.

      • The third occurrence of the **TEMP_SAC** segment does not contain data, so the translator is finished processing the **TEMP_SAC** segment for this iteration of the **SLN** group.

    d. The translator processes the first occurrence of the **TEMP_SAC** segment in the second **SLN** group of the first **IT1** group to verify whether there is any data to be linked across to the Output side of the map.

- The first occurrence of the **TEMP_SAC** segment does not contain data, so the translator is finished processing the **TEMP_SAC** segment for this iteration of the **SLN** group.

- The extended rule wrote the data to the *improper* occurrence in the array (1,2,3), as noted in the preceding table, which is why the translator did not find the data (because it was looking for the data in occurrence 1,2,1).

    e. The translator processes the first occurrence of the **TEMP_SAC** segment in the first **SLN** group of the second **IT1** group to verify whether there is any data to be linked across to the Output side of the map.

- The first occurrence of the **TEMP_SAC** segment does not contain data, so the translator is finished processing the **TEMP_SAC** segment for this iteration of the **SLN** group.

- The extended rule again wrote the data to the *improper* occurrence in the array (2,3,5), as noted in the preceding table, which is why the translator did not find the data (because it was looking for the data in occurrence 2,1,1).

After running translation with the modified map, only two occurrences of the SAC segment were received, rather than the expected five segment, because the indexes for the **SLN** group and the **TEMP_SAC** group were not reset on their parent structure.

Note: When you are working with nested structures and using indexes to map data to temporary structures, the child indexes must be reset to 1 on the start of their parent structure.

In reviewing the array built by the translator in the table on page 373, we can see that a missing occurrence of data happened when the translator wrote the third iteration of data to the **TEMP_SAC** segment. When the **If** statement found the third **SAC** segment, which did not have a value of zero in the **SAC05**, the data was written to the third occurrence of the **TEMP_SAC** segment (in the second occurrence of the **SLN** segment in the first occurrence of the **IT1** segment). This is the missing occurrence of data. The next section explains how to use complete indexes so there are no occurrences of missing data in the map.

Map Setup Using Complete Indexes (Complex Example)

The On End extended rule on the **SAC** segment that was built in the previous section is correct. However, you did not configure the map to reset the child indexes, which caused the improper output.

Resetting Indexes to 1

The **TEMP_SAC** segment is a child to the **SLN** segment, so you must reset the **Z** index to 1 on the On Begin of the **SLN** group. For example:

```
Z = 1;
```

The **SLN** group is a child to the **IT1** group, so you must reset the **Y** index to 1 on the On Begin of the **IT1** group. For example:

```
Y = 1;
```

The index used for the **IT1** group does not need to be reset, because the **IT1** group is the first parent in the nesting of groups. If you did reset the index for the **IT1** group, the rule on the **SAC** segment would overwrite data already written to the **TEMP_SAC** segment.

Creating the Array

The following table describes how the translator will create the array after you added the rules to reset the **Y** and **Z** indexes:

Loop Hierarchical Structure	Occurrence	Data
First IT1, First SLN, First TEMP_SAC	1,1,1	SAC*S**AB**500.00****EA*20***Service**Service
First IT1, First SLN, Second TEMP_SAC	1,1,2	SAC*S**AB**121.11****EA*10***Service**Service
First IT1, Second SLN, First TEMP_SAC	1,2,1	SAC*S**AB**250****EA*2***Service**Service
Second IT1, First SLN, First TEMP_SAC	2,1,1	SAC*S**AB**45.67****EA*200***Service
Second IT1, First SLN, Second TEMP_SAC	2,1,2	SAC*S**AB**222.74****EA*103.789***Service

Now you are properly using indexes in the map, and when translation is run, the information from all five **SAC** segments is received in the Output file, as expected.

Using Indexes with an XML File Format

Working with indexes within an XML File Format in a map is similar to using them with other file formats: Delimited, Positional, SQL, and so forth. The one main difference is this: with an XML file format, every XML element and Content Particle in the hierarchy—down to the element being referenced by the rule—requires an index because the translator views elements and content particles as groups.

Note: You do not need an index for the XML File Format Object (Root Element).

For more information, see *Example of Using Indexes with XML File Format* on page 375.

Example of Using Indexes with XML File Format

You are receiving XML data and, in the detail section (**LineItem** element), you only need the information mapped to the output file *if* the QuantityShipped value is greater than zero.

In the map, you created a temporary input repeating structure (**Temp_LineItem**) to receive the information for only those iterations of the **LineItem** element in which the QuantityShipped value is greater than zero. Also, you linked elements in the **Temp_LineItem** element structure to the appropriate Output segment.

Map Setup

You must declare an index because **Temp_LineItem** is a repeating structure, and the map must write the data consecutively to the repeating iterations of **Temp_LineItem** to verify that no data is missed when the Output segment is created during translation.
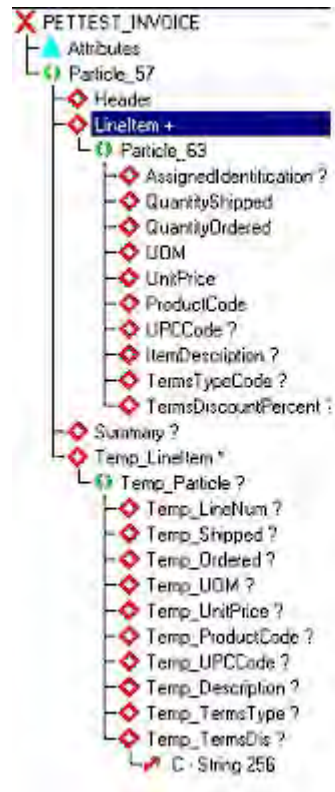
For the Input Level (PETTEST_INVOICE) On-Begin extended rules, you declared and initialized an index, as the following example indicates:

```
Integer X;
X = 1;
```

Also, you added the following an On-End extended rule for the **LineItem** element:

```
If $QuantityShipped.#QuantityShipped > 0 Then
Begin
    $Temp_LineNum[X].#Temp_LineNum =
$AssignedIdentification.#AssignedIdentification;
    $Temp_Shipped[X].#Temp_Shipped = $QuantityShipped.#QuantityShipped;
    $Temp_Ordered[X].#Temp_Ordered = $QuantityOrdered.#QuantityOrdered;
    $Temp_UOM[X].#Temp_UOM = $UOM.#UOM;
    $Temp_UnitPrice[X].#Temp_UnitPrice = $UnitPrice.#UnitPrice;
    $Temp_ProductCode[X].#Temp_ProductCode = $ProductCode.#ProductCode;
    $Temp_UPCCode[X].#Temp_UPCCode = $UPCCode.#UPCCode;
    $Temp_Description[X].#Temp_Description = $ItemDescription.#ItemDescription;
    $Temp_TermsType[X].#Temp_TermsType = $TermsTypeCode.#TermsTypeCode;
    $Temp_TermsDis[X].#Temp_TermsDis =
$TermsDiscountPercent.#TermsDiscountPercent;
    X = X + 1;
End
```

After you compile this extended rule, you receive the following compilation errors:
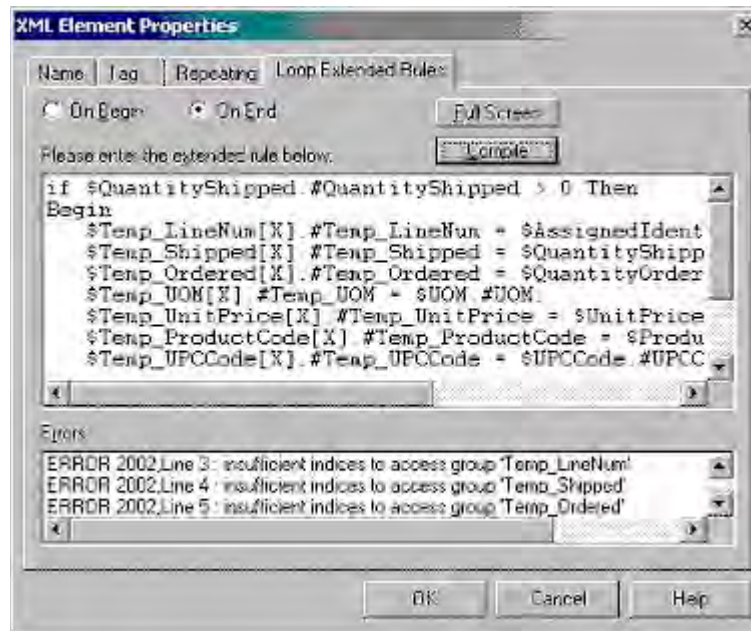
Why Were These Errors Received?

You received the preceding errors because you did not provide indexes for every content particle and element in the hierarchical map structure, down to the elements you referenced.

So, to assign the value of the elements in the **LineItem** elements to the temporary elements in the **Temp_LineItem** element, you need to use four indexes for each assignment statement in this rule, as indicated in the following example:
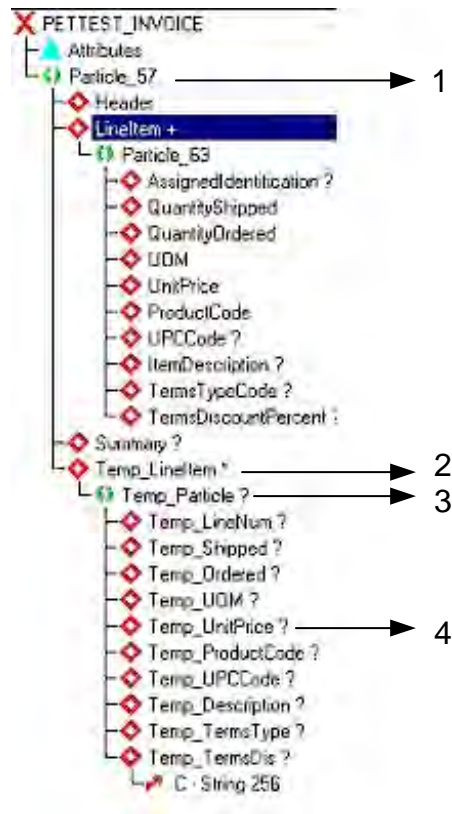
So, you now know that you need four indexes for each assignment statement in the rule, but not all the elements and content particles in this hierarchical structure repeat, and thus you will need to address the non-repeating elements and content particles separately, as noted in the subsequent topic.

How to Use an Index for Elements and Content Particles that Do Not Repeat

The content particles and elements that do not repeat must use an index of [1], because there is only one occurrence of each of these structures.

However, when supplying an index for each content particle and element in the hierarchical structure to the element being referenced, the index list of the rule can become quite lengthy.

Caution: It is very important that you verify the indexes are in the correct order when you write the rule, or you will not receive the correct output.

Order of Index Placement

Indexes are *always* listed in order, starting from the **Root** Element, and continuing successively down to the element you are referencing in the rule. The following table illustrates the order in which the indexes must be placed for each assignment statement in the extended rule:

Order	Element or Content Particle	Index Used	Why that Index is Used
1	Particle_57	[1]	Non-Repeating Content Particle
2	Temp_LineItem	[X]	Repeating Element
3	Temp_Particle	[1]	Non-Repeating Content Particle
4	<Temporary Element Name>	[1]	Non-Repeating Element

Now that you established the number of indexes that are needed, you can apply the entire rule, as indicated in the following example:

```
If $QuantityShipped.#QuantityShipped > 0 Then
Begin
   $Temp_LineNum[1][X][1][1].#Temp_LineNum =
$AssignedIdentification.#AssignedIdentification;
   $Temp_Shipped[1][X][1][1].#Temp_Shipped =
$QuantityShipped.#QuantityShipped;
   $Temp_Ordered[1][X][1][1].#Temp_Ordered =
$QuantityOrdered.#QuantityOrdered;
   $Temp_UOM[1][X][1][1].#Temp_UOM = $UOM.#UOM;
   $Temp_UnitPrice[1][X][1][1].#Temp_UnitPrice = $UnitPrice.#UnitPrice;
   $Temp_ProductCode[1][X][1][1].#Temp_ProductCode =
$ProductCode.#ProductCode;
   $Temp_UPCCode[1][X][1][1].#Temp_UPCCode = $UPCCode.#UPCCode;
   $Temp_Description[1][X][1][1].#Temp_Description =
$ItemDescription.#ItemDescription;
   $Temp_TermsType[1][X][1][1].#Temp_TermsType =
$TermsTypeCode.#TermsTypeCode;
   $Temp_TermsDisc[1][X][1][1].#Temp_TermsDis =
$TermsDiscountPercent.#TermsDiscountPercent;
   X = X + 1;
End
```

If you review the sample data file for the preceding scenario in *Sample: Repeating Structure Received for XML Inbound Scenario* on page 381, you will note that the third, fourth, and seventh occurrences of the **LineItem** element contain a value of zero for the **QuantityShipped**. This is important because if the extended rule you create does not contain indexes, when the translator reads the third occurrence of the array and determines it does not contain data, it assumes the remaining occurrences of the array are empty, and thus the translator concludes that its processing of the **DET** Output segment is complete and any data that is present in the remaining occurrences of the array is not mapped to the output side of the map.

The following table is an example of the array that the translator creates for the **Temp_LineItem** element using the extended rule with the index:

Occ	Temp _ LineNum	Temp _ Shipped	Temp _ Ordered	Temp _ UOM	Temp _ Unit Price	Temp _ Product Code	Temp _ UPCCode	Temp _ Description	Temp _ Terms Type	Temp _ Terms Dis
1	00001	100	100	EA	100	32166	1236547894	This is the 1st item description	XX	0
2	00002	50	100	EA	25.5	12354	321321123	This is the 2nd item description	10	0
3	00003	5	5	EA	500	33245	2222222222 22	This is the 3rd item description	10	0
4	00004	250	250	EA	0.5	77511	987654321	This is the 4th item description	10	0

Note: The "Occ" column indicates in which occurrence of the array the data is stored, and the remaining column headings indicate each temporary element in the **Temp_LineItem** structure.

The following is the output file you receive after translating the map:

```
HDRPETXML   P1.0   INV    P
BEGINV5008   Invoic1260588259950510222222222233333333333      VENDOR Name                         VENDOR ADDRESS 1
CM2INV5008   Invoic1260588259TEXTTEXTTEXT
DETINV5008   Invoic1260588259000001 100   100   100    EA 32166     1236547894   THIS IS THE 1ST ITEM DESCRIPTIXX00
DETINV5008   Invoic1260588259000002 50    100   25.5   EA 12354     321321123    THIS IS THE 2ND ITEM DESCRIPTI1000
DETINV5008   Invoic1260588259000005 5     5     500    EA 33245     222222222222 THIS IS THE 5TH ITEM DESCRIPTI1000
DETINV5008   Invoic1260588259000006 250   250   0.5    EA 77511     987654321    THIS IS THE 6TH ITEM DESCRIPTI1000
SUNINV5008   Invoic1260588259913900      4
```

This time you received the proper output because the extended rule wrote the necessary information down to the **Temp_LineItem** structure consecutively. This prevented missing occurrences of data in the array. If the extended rule did not use indexes, the third, fourth, and seventh occurrences of the array would be empty. This would cause the Output file to contain only the first two occurrences of the **DET** segment on the output side of the map, because when the translator reads the third occurrence of the array and determines it does not contain data, it assumes the remaining occurrences of the array are empty. Thus, the translator concludes that its processing of the **DET** segment is complete and any data that is present in the remaining occurrences of the array is not mapped to the output side of the map.

Sample: Repeating Structure Received for XML Inbound Scenario

The following is a sample of the repeating structure that is received in the Input file for this XML Scenario:

```
<LineItem>
  <AssignedIdentification>00001</AssignedIdentification>
  <QuantityShipped>100</QuantityShipped>
  <QuantityOrdered>100</QuantityOrdered>
  <UOM>EA</UOM>
  <UnitPrice>100.00</UnitPrice>
  <ProductCode>32166</ProductCode>
  <UPCCode>1236547894</UPCCode>
  <ItemDescription>THIS IS THE 1ST ITEM DESCRIPTION</ItemDescription>
  <TermsTypeCode>XX</TermsTypeCode>
  <TermsDiscountPercent>0</TermsDiscountPercent>
</LineItem>
<LineItem>
  <AssignedIdentification>00002</AssignedIdentification>
  <QuantityShipped>50</QuantityShipped>
  <QuantityOrdered>100</QuantityOrdered>
  <UOM>EA</UOM>
  <UnitPrice>25.50</UnitPrice>
  <ProductCode>12354</ProductCode>
  <UPCCode>321321123</UPCCode>
  <ItemDescription>THIS IS THE 2ND ITEM DESCRIPTION</ItemDescription>
  <TermsTypeCode>10</TermsTypeCode>
  <TermsDiscountPercent>0</TermsDiscountPercent>
</LineItem>
<LineItem>
  <AssignedIdentification>00003</AssignedIdentification>
  <QuantityShipped>0</QuantityShipped>
  <QuantityOrdered>100</QuantityOrdered>
  <UOM>EA</UOM>
  <UnitPrice>7.50</UnitPrice>
  <ProductCode>98754</ProductCode>
  <UPCCode>12365478989</UPCCode>
  <ItemDescription>THIS IS THE 3RD ITEM DESCRIPTION</ItemDescription>
  <TermsTypeCode>10</TermsTypeCode>
  <TermsDiscountPercent>0</TermsDiscountPercent>
</LineItem>
<LineItem>
  <AssignedIdentification>00004</AssignedIdentification>
  <QuantityShipped>0</QuantityShipped>
  <QuantityOrdered>10</QuantityOrdered>
  <UOM>EA</UOM>
  <UnitPrice>100.00</UnitPrice>
  <ProductCode>21222</ProductCode>
  <UPCCode>222222222222</UPCCode>
  <ItemDescription>THIS IS THE 4TH ITEM DESCRIPTION</ItemDescription>
  <TermsTypeCode>10</TermsTypeCode>
  <TermsDiscountPercent>0</TermsDiscountPercent>
</LineItem>

//CONTINUED ON NEXT PAGE
```

```
<LineItem>
  <AssignedIdentification>00005</AssignedIdentification>
  <QuantityShipped>5</QuantityShipped>
  <QuantityOrdered>5</QuantityOrdered>
  <UOM>EA</UOM>
  <UnitPrice>500.00</UnitPrice>
  <ProductCode>33245</ProductCode>
  <UPCCode>222222222222</UPCCode>
  <ItemDescription>THIS IS THE 5TH ITEM DESCRIPTION</ItemDescription>
  <TermsTypeCode>10</TermsTypeCode>
  <TermsDiscountPercent>0</TermsDiscountPercent>
</LineItem>
<LineItem>
  <AssignedIdentification>00006</AssignedIdentification>
  <QuantityShipped>250</QuantityShipped>
  <QuantityOrdered>250</QuantityOrdered>
  <UOM>EA</UOM>
  <UnitPrice>0.50</UnitPrice>
  <ProductCode>77511</ProductCode>
  <UPCCode>987654321</UPCCode>
  <ItemDescription>THIS IS THE 6TH ITEM DESCRIPTION</ItemDescription>
  <TermsTypeCode>10</TermsTypeCode>
  <TermsDiscountPercent>0</TermsDiscountPercent>
</LineItem>
<LineItem>
  <AssignedIdentification>00007</AssignedIdentification>
  <QuantityShipped>0</QuantityShipped>
  <QuantityOrdered>175</QuantityOrdered>
  <UOM>EA</UOM>
  <UnitPrice>69.99</UnitPrice>
  <ProductCode>22887</ProductCode>
  <UPCCode>123456789</UPCCode>
  <ItemDescription>THIS IS THE 7TH ITEM DESCRIPTION</ItemDescription>
  <TermsTypeCode>10</TermsTypeCode>
  <TermsDiscountPercent>0</TermsDiscountPercent>
</LineItem>
```

Hierarchical Levels (HL) in Maps

This section contains detailed information about how to use accumulators in the Hierarchical Level (HL) segment, as well as how to set up the accumulators.

For more information about accumulators, see Chapter 8, *Using Standard Rules*.

This section covers the following topics:

✦ Introduction to Using Accumulators in the HL Segment

✦ Example of Using Accumulators in the HL Segment

Introduction to Using Accumulators in the HL Segment

Many Gentran Integration Suite users send out Advanced Ship Notices (856 documents). Some users have data from their applications that do not contain the HL information. Gentran Integration Suite users can set up accumulators in the map to populate the appropriate fields in the HL and CTT segments. You can also use accumulators to populate the HL information if you need to manipulate data on the input side of the map and the HL information from your application is no longer correct.

Example of Using Accumulators in the HL Segment

This section contains the creating process for an 856 document that contains four HL Levels: Shipment, Order, Pack, and Item.

For more information, see:

✦ *Scenario* on page 384

✦ *Creating the Proper HL Structure in the Map* on page 385

✦ *Creating the HL Accumulators* on page 387

✦ *Shipment HL Segment* on page 388

Scenario

In this scenario, we assume that no other accumulators are used anywhere else in the 856 map, and we will reference accumulators **0**, **1**, **2**, **3**—using a specific accumulator for each HL Level:

✦ Accumulator 0 is used on all HL segments to populate the HL01 value

✦ Accumulator 1 is used on the Order HL segment to populate the HL02 value

✦ Accumulator 2 is used on the Pack HL segment to populate the HL02 value

✦ Accumulator 3 is used on the Item HL segment to populate the HL02 value

The HL segment contains the following fields:

✦ HL01–HIERARCHICAL ID NUMBER

✦ HL02–HIERARCHICAL PARENT ID NUMBER

✦ HL03–HIERARCHICAL LEVEL CODE

✦ HL04–HIERARCHICAL CHILD CODE

Note: This scenario does not include the procedure for populating the HL04 element.

The **HL01 element (Hierarchical ID Number)** identifies the HL segment numerically in the EDI document. The field starts at one and increments by one for each subsequent HL segment that is created. The accumulator **0** is used for the HL01 element. Each HL segment (Shipment, Order, Pack, and Item) uses the same accumulator for the HL01 element, and the map increments the accumulator and then uses the accumulator for each HL01 element.

The **HL02 element (Hierarchical Parent ID)** identifies the Parent HL segment to which the current HL segment belongs. The accumulator **1** is used for the HL02 in the Order HL segment, and this element is not populated in the Shipment HL segment because it is the first HL segment in the document.

The **HL03 element (Hierarchical Level Code)** indicates the level of the HL segment (Shipment, Order, Pack or Item) by using a code. **S** indicates the Shipment Level, **O** indicates the Order level, **P** indicates the Pack level, and **I** indicates the Item level. The map uses constants to populate this element.

For each HL, except for the innermost nested HL (which is Item in this scenario), the map also moves the value contained in accumulator **0**, located on the HL01 element, into the accumulator that is used for the HL02 element in the next HL segment.

For example, the standard rule on the HL01 element in the Shipment HL segment increments accumulator **0** by one and places that value in the HL01 element. Then, the rule moves that same value into accumulator **1** for use on the HL02 element in the Order HL segment.

Creating the Proper HL Structure in the Map

This section assumes that you created an Import map with an ANSI 856 structure on the output side.

To create the proper HL structure for this map:

1. Select the HL group in the map.

2. From the **Edit** menu, select **Copy**.

3. Select the last segment or group in the HL group.

4. Paste the copied HL group.

Note: If the last object in the HL group is a group, you are prompted to select at which level you want the HL group pasted. Select **Paste after the selected item (at the same level)** and click **OK**.

The map structure now has the original HL group that contains a child HL group. The following figure demonstrates how the map looks after you complete these steps (the child HL group is selected):
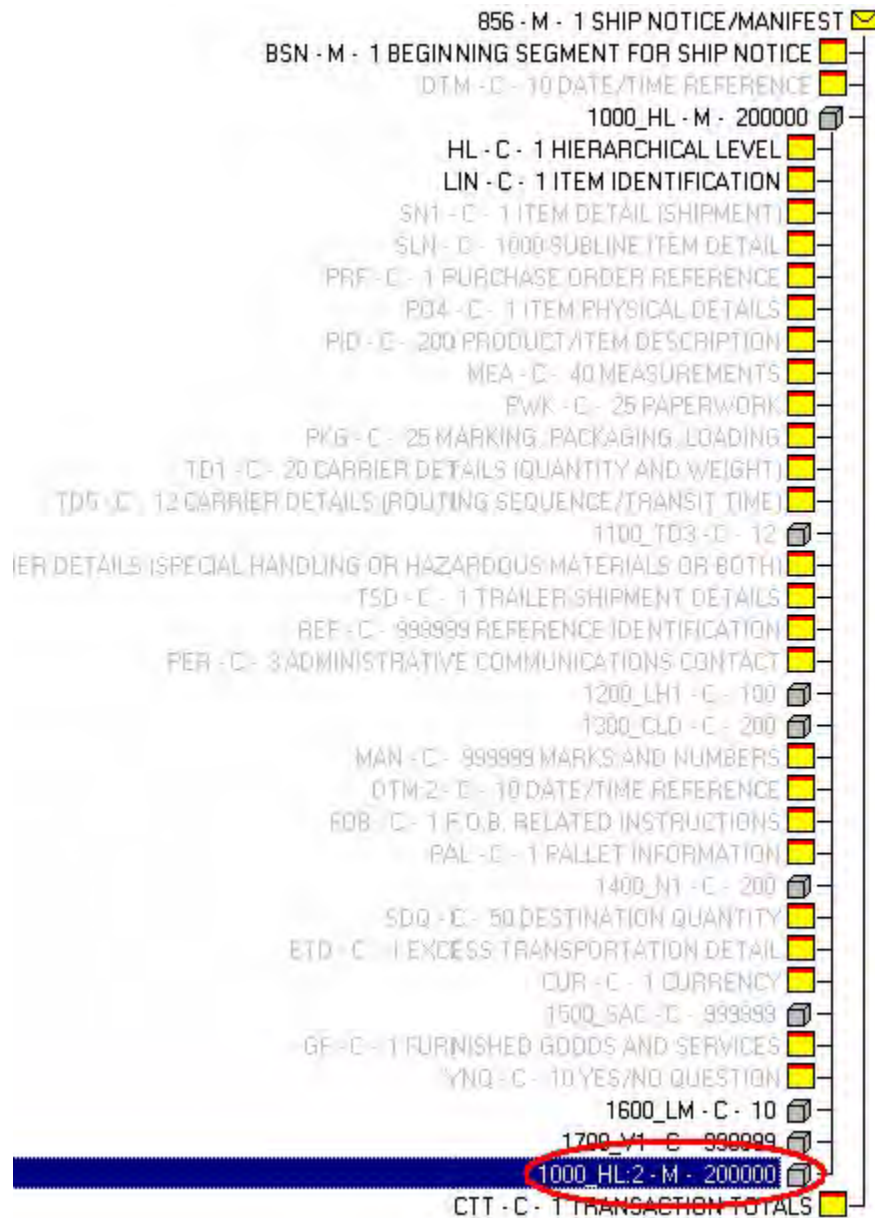
Repeat this process for each HL level that is needed, remembering to paste each new HL group as a child of the previously pasted HL group.

Activate the necessary groups and segments needed for each HL group after you create the necessary structure.

Note: If the hierarchical levels (that is, shipment, order, and so forth) are singly occurring groups, the option **Promote records to parent** (on the Looping tab of the Group Properties dialog box), which is only available for singly occurring groups, can be selected. This option causes the translator to treat the records as if they are one level higher in the hierarchy.

Only select this option if your input structure is at the same hierarchical level as the output structure.

For more information about descriptions of dialog box components, see Appendix B, *Map Editor Properties.*

Creating the HL Accumulators

All accumulators must be initialized because the translator does not do so at run-time. Therefore, you must place a standard rule on an element in the BSN segment that zeros all the accumulators that are used. This prevents invalid data in the accumulators when they are referenced later in the map.

Initializing the Accumulators

To initialize the HL accumulators:

1. In the Map Editor, right-click an element that does not currently have a standard rule on it.

   The Element Properties dialog box opens.

2. Click the **Standard Rule** tab to access the standard rule options.

3. From the standard rule list, select **Use Accumulator**.

4. Click **New**.

   The Edit Accumulator Entry dialog box opens to create a new calculation for this field.

5. From the **Primary Accumulator** list, select **0**.

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

6. In the **Name** box, type the accumulator a name, to use as a descriptive alias that enables you to differentiate what the accumulators you create are used for.

7. From the **First** list, select **Zero primary**.

8. In the **Edit Accumulator** dialog box, click **OK** to add the accumulator.

9. In the **Element Properties** dialog box, click **OK** to add the standard rule to the element.

10. Complete the preceding steps for each accumulator you are using. For step 5, select the next sequential accumulator (that is, if you have already used accumulator **0** the next accumulator is **1**).

The following figure shows what the Standard Rule should look like after all accumulators are initialized:
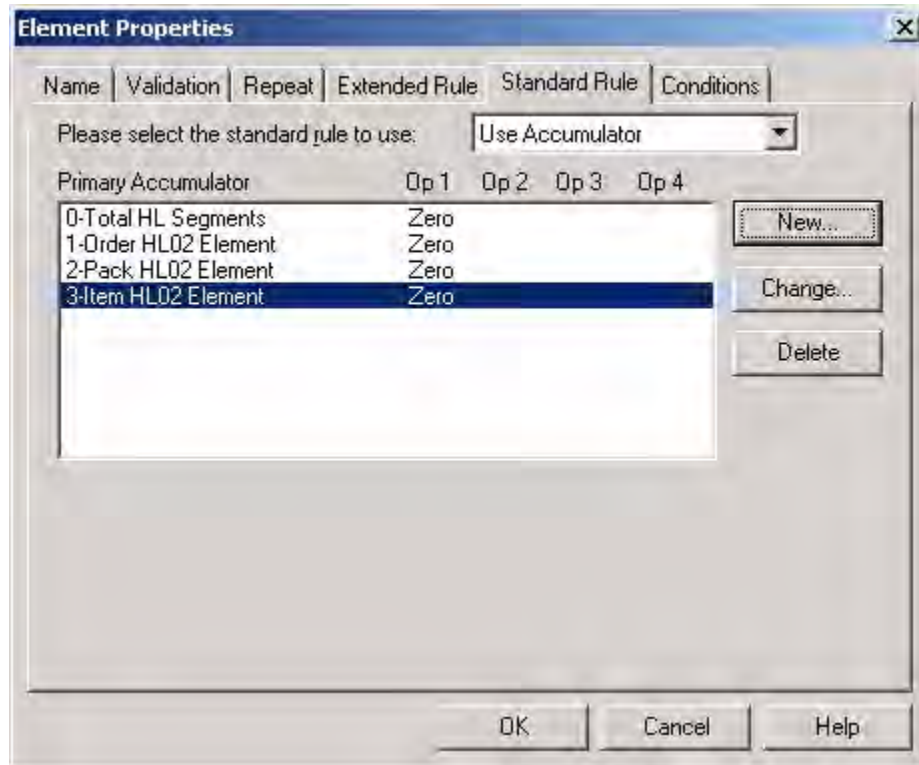
Shipment HL Segment

Now you must create the accumulators for the elements in the Shipment HL segment.

Creating the HL01 Element Accumulators

To create the HL01 element accumulators:

1. In the Map Editor, right-click the **HL01** element and select **Properties** from the shortcut menu.

    The Element Properties dialog box opens.

2. Click the **Standard Rule** tab to access the standard rule options.

3. From the standard rule list, select **Use Accumulator**.

4. Click **New**.

   The Edit Accumulator Entry dialog box opens to create a new calculation for this field.

5. From the **Primary Accumulator** list, select **0**. This is the accumulate that you created in step 5 in *Initializing the Accumulators* on page 387.

6. From the **First** list, select **Increment primary**.

   Increment primary is the first operation performed. The value in the accumulator increments by one for each iteration of the current group.

7. From the **Second** list, select **Use primary**.

   Use primary is the second operation performed, after the first operation is completed. The current value of the accumulator is loaded into the Assigned Identification field.

8. In the **Edit Accumulator** dialog box, click **OK** to add the accumulator.

9. Click **New**.

   The Edit Accumulator Entry dialog box opens to create a new calculation for this field.

10. From the **Primary Accumulator** list, again select **0**.

11. From the **First** list, select **Move primary to alternate**.

12. From the **Alternate Accum** list, select **1**.

13. In the **Edit Accumulator** dialog box, click **OK** to add the accumulator.

The Element Properties dialog box should look like the following figure:
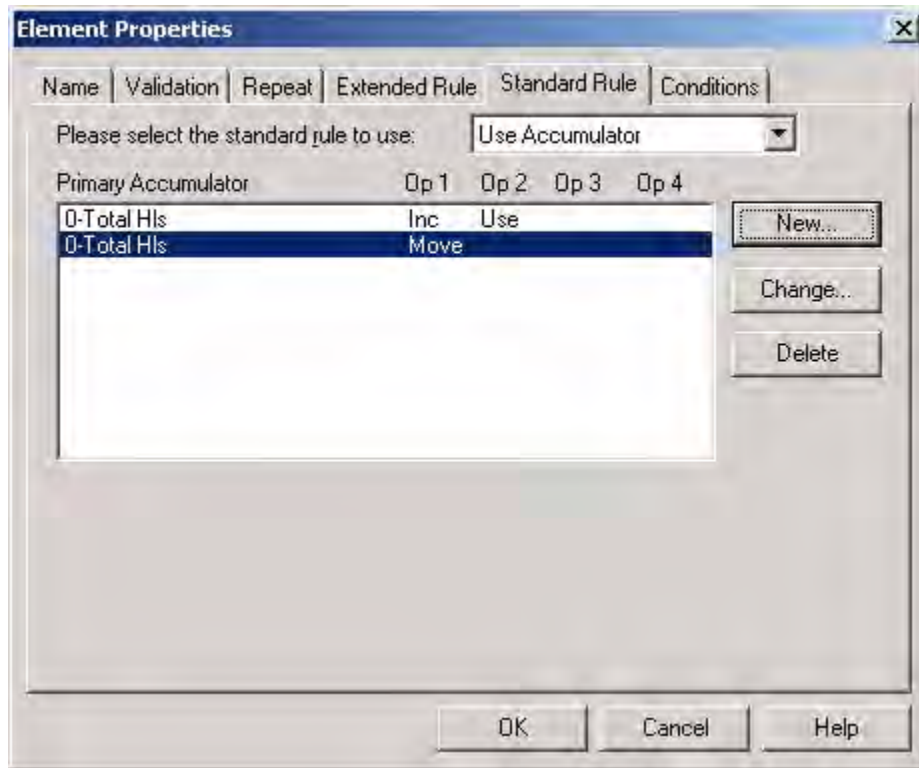
14. In the **Element Properties** dialog box, click **OK** to add the standard rule to the Line Items field.

Changing the HL03 Element

To change the properties of the HL03 element:

1. In the Map Editor, right-click the **HL03** element and select **Properties** from the shortcut menu.

The Element Properties dialog box opens.

2. Apply a standard or extended rule that places the constant **S** in this element.

For more information about standard rules, see Chapter 8, *Using Standard Rules*. For more information about extended rules, see Chapter 9, *Using Extended Rules*.

Order HL Segment

Now you must create the accumulators for the elements in the Order HL segment.

Creating the HL01 Element Accumulators

To create the HL01 element accumulators:

1. In the Map Editor, right-click the **HL01** element and select **Properties** from the shortcut menu.

   The Element Properties dialog box opens.

2. Click the **Standard Rule** tab to access the standard rule options.

3. From the standard rule list, select **Use Accumulator**.

4. Click **New**.

   The Edit Accumulator Entry dialog box opens to create a new calculation for this field.

5. From the **Primary Accumulator** list, select **0**.

6. From the **First** list, select **Increment primary**.

7. From the **Second** list, select **Use primary**.

8. In the **Edit Accumulator** dialog box, click **OK** to add the accumulator.

9. Click **New**.

   The Edit Accumulator Entry dialog box opens to create a new calculation for this field.

10. From the **Primary Accumulator** list, again select **0**.

11. From the **First** list, select **Move primary to alternate**.

12. From the **Alternate Accum** list, select **2**.

13. In the **Edit Accumulator** dialog box, click **OK** to add the accumulator.

14. In the **Element Properties** dialog box, click **OK** to add the standard rule to the Line Items field.

Creating the HL02 Element Accumulator

To create the HL02 element accumulators:

1. In the Map Editor, right-click the **HL02** element and select **Properties** from the shortcut menu.

   The Element Properties dialog box opens.

2. Click the **Standard Rule** tab to access the standard rule options.

3. From the standard rule list, select **Use Accumulator**.

4. Click **New**.

   The Edit Accumulator Entry dialog box opens to create a new calculation for this field.

5. From the **Primary Accumulator** list, select **1**.

6. From the **First** list, select **Use primary**.

7. In the **Edit Accumulator** dialog box, click **OK** to add the accumulator.

8. In the **Element Properties** dialog box, click **OK** to add the standard rule to the Line Items field.

Changing the HL03 Element

To change the properties of the HL03 element:

1. In the Map Editor, right-click the **HL03** element and select **Properties** from the shortcut menu.

    The Element Properties dialog box opens.

2. Apply a standard or extended rule that places the constant **O** in this element.

    For more information about standard rules, see Chapter 8, *Using Standard Rules*. For more information about extended rules, see Chapter 9, *Using Extended Rules*.

Pack HL Segment

Now you must create the accumulators for the elements in the Pack HL segment.

Creating the HL01 Element Accumulators

To create the HL01 element accumulators:

1. In the Map Editor, right-click the **HL01** element and select **Properties** from the shortcut menu.

    The Element Properties dialog box opens.

2. Click the **Standard Rule** tab to access the standard rule options.

3. From the standard rule list, select **Use Accumulator**.

4. Click **New**.

    The Edit Accumulator Entry dialog box opens to create a new calculation for this field.

5. From the **Primary Accumulator** list, select **0**.

6. From the **First** list, select **Increment primary**.

7. From the **Second** list, select **Use primary**.

8. In the **Edit Accumulator** dialog box, click **OK** to add the accumulator.

9. Click **New**.

    The Edit Accumulator Entry dialog box opens to create a new calculation for this field.

10. From the **Primary Accumulator** list, again select **0**.

11. From the **First** list, select **Move primary to alternate**.

12. From the **Alternate Accum** list, select **3**.

13. In the **Edit Accumulator** dialog box, click **OK** to add the accumulator.

14. In the **Element Properties** dialog box, click **OK** to add the standard rule to the Line Items field.

Creating the HL02 Element Accumulator

To create the HL02 element accumulators:

1. In the Map Editor, right-click the **HL02** element and select **Properties** from the shortcut menu.

    The Element Properties dialog box opens.

2. Click the **Standard Rule** tab to access the standard rule options.

3. From the standard rule list, select **Use Accumulator**.

4. Click **New**.

    The Edit Accumulator Entry dialog box opens to create a new calculation for this field.

5. From the **Primary Accumulator** list, select **2**.

6. From the **First** list, select **Use primary**.

7. In the **Edit Accumulator** dialog box, click **OK** to add the accumulator.

8. In the **Element Properties** dialog box, click **OK** to add the standard rule to the Line Items field.

Changing the HL03 Element

To change the properties of the HL03 element:

1. In the Map Editor, right-click the **HL03** element and select **Properties** from the shortcut menu.

    The Element Properties dialog box opens.

2. Apply a standard or extended rule that places the constant **P** in this element.

    For more information about standard rules, see Chapter 8, *Using Standard Rules*. For more information about extended rules, see Chapter 9, *Using Extended Rules*.

Item HL Segment

Now you must create the accumulators for the elements in the Pack HL segment.

Creating the HL01 Element Accumulators

To create the HL01 element accumulators:

1. In the Map Editor, right-click the **HL01** element and select **Properties** from the shortcut menu.

    The Element Properties dialog box opens.

2. Click the **Standard Rule** tab to access the standard rule options.

3. From the standard rule list, select **Use Accumulator**.

4. Click **New**.

   The Edit Accumulator Entry dialog box opens to create a new calculation for this field.

5. From the **Primary Accumulator** list, select **0**.

6. From the **First** list, select **Increment primary**.

7. From the **Second** list, select **Use primary**.

8. In the **Edit Accumulator** dialog box, click **OK** to add the accumulator.

9. In the **Element Properties** dialog box, click **OK** to add the standard rule to the Line Items field.

Creating the HL02 Element Accumulator

To create the HL02 element accumulators:

1. In the Map Editor, right-click the **HL02** element and select **Properties** from the shortcut menu.

   The Element Properties dialog box opens.

2. Click the **Standard Rule** tab to access the standard rule options.

3. From the standard rule list, select **Use Accumulator**.

4. Click **New**.

   The Edit Accumulator Entry dialog box opens to create a new calculation for this field.

5. From the **Primary Accumulator** list, select **3**.

6. From the **First** list, select **Use primary**.

7. In the **Edit Accumulator** dialog box, click **OK** to add the accumulator.

8. In the **Element Properties** dialog box, click **OK** to add the standard rule to the Line Items field.

Changing the HL03 Element

To change the properties of the HL03 element:

1. In the Map Editor, right-click the **HL03** element and select **Properties** from the shortcut menu.

   The Element Properties dialog box opens.

2. Apply a standard or extended rule that places the constant **I** in this element.

   For more information about standard rules, see Chapter 8, *Using Standard Rules*. For more information about extended rules, see Chapter 9, *Using Extended Rules*.

CTT Segment

The CTT01 must be populated with the total number of HL segments created in the document. To populate this element in the map, use the accumulator **0** because it already contains the total number of HL segments.

Creating the CTT01 Element Accumulator

To create the CTT01 element accumulators:

1. In the Map Editor, right-click the **CTT01** element and select **Properties** from the shortcut menu.

   The Element Properties dialog box opens.

2. Click the **Standard Rule** tab to access the standard rule options.

3. From the standard rule list, select **Use Accumulator**.

4. Click **New**.

   The Edit Accumulator Entry dialog box opens to create a new calculation for this field.

5. From the **Primary Accumulator** list, select **0**.

6. From the **First** list, select **Use primary**.

7. In the **Edit Accumulator** dialog box, click **OK** to add the accumulator.

8. In the **Element Properties** dialog box, click **OK** to add the standard rule to the Line Items field.

Visual Representation of the Hierarchical Levels

The following figure was created from the sample EDI file located in *Sample EDI File* on page 396:
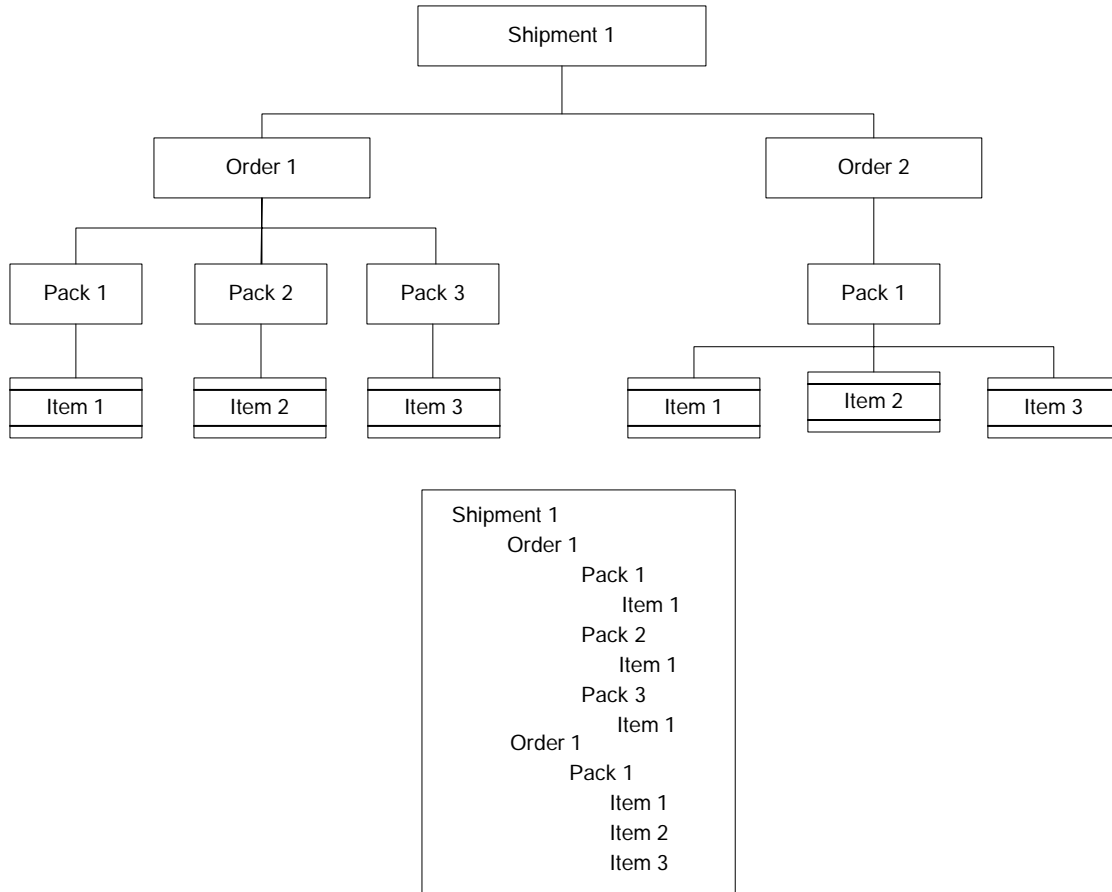
Sample EDI File

The following EDI file is a sample that enables you to see where the HL segment information is located.

```
BSN*00*000*19990106*11:28:19~

HL*1**S~

REF*SI*0000001017~

HL*2*1*O~

PRF*123456789~

REF*OQ*1254123-C~

REF*SI*0000001017~
```

```
DTM*086*20020115~
N1*SH*Sterling Commerce~
N3*4600 Lakehurst Ct~
N4*Dublin*OH*43016*USA~
HL*3*2*P~
PO4*1*****30*LB***15*10*3*IN~
HL*4*3*I~
LIN**IN*NOTEBOOK #2*LT*NONE*SN*SYS8-1475~
SN1**1*EA~
PID*F****System 8000 Notebook, 32MB RAM~
REF*SI*005865238~
REF*OQ*1254123-C~
HL*5*2*P~
PO4*1*****30*LB***15*10*3*IN~
HL*6*5*I~
LIN**IN*EXCEL~
SN1**6*EA~
PID*F****Software~
REF*SI*58794312558~
REF*OQ*SOFT3256~
HL*7*2*P~
PO4*1*****30*LB***15*10*3*IN~
HL*8*7*I~
SN1**2*EA~
PID*F****600 dpi Laser Printer~
REF*SI*05987642341~
REF*OQ*1254123-A~
HL*9*1*O~
PRF*987654351~
REF*OQ*EAS1008~
REF*SI*55522214521~
DTM*086*20020115~
N1*SH*Sterling Commerce~
N3*4600 Lakehurst Ct~
N4*Dublin*OH*43016*USA~
HL*10*9*P~
```

```
PO4*2*****300*LB***15*10*3*IN~
HL*11*10*I~
SN1**3*EA~
PID*F****Software~
REF*SI*2222333645~
REF*OQ*SOFT3256~
HL*12*10*I~
SN1**3*EA~
PID*F****Monitor 20 in., 1600x1200, .28~
REF*SI*0000001019~
REF*OQ*MON5421~
HL*13*10*I~
SN1**3*EA~
PID*F****Keyboard~
REF*SI*0000001019~
REF*OQ*KEY548795~
REF*SI*0000001019~
CTT*13~
```

Map Conversion

This section contains detailed information about how to use the Gentran Integration Suite map conversion utilities.

This section covers the following topics:

✦ About Map Conversion
✦ Converting Gentran:Server for UNIX Maps to Gentran Integration Suite
✦ Converting Gentran:Server for iSeries Maps to Gentran Integration Suite
✦ Converting Gentran:Basic for zSeries Maps to Gentran Integration Suite
✦ Converting Maps Using the Command Line

About Map Conversion

The Gentran Integration Suite Map Conversion utilities enable you to quickly and easily convert maps created using other Sterling Commerce products to Gentran Integration Suite maps. The map conversion functionality enable you to import maps into the Map Editor from the original Gentran:Server for UNIX, Gentran:Server for iSeries, and Gentran:Basic for zSeries map files.

Note: For Gentran:Server for iSeries and Gentran:Basic for zSeries maps, the map links, standard rules, extended rules, conditional relationships, and user exits are not converted by the functionality. You must add these items in to the converted map.

For Gentran:Server for iSeries, the conversion utility only works for applications and maps that are created as internally defined files. The conversion does not work for any application or map that was created as "externally described" or created with a logical file.

For Gentran:Server for UNIX maps, the simple links (unless you specify that they *should* be converted), standard rules, extended rules, conditional relationships, and user exits are not converted by the functionality. You must add these items in to the converted map.

You can also convert Gentran:Server for iSeries and Gentran:Basic for zSeries maps using the command line. To use the command line conversion switches, see *Converting Maps Using the Command Line* on page 403.

Converting Gentran:Server for UNIX Maps to Gentran Integration Suite

Gentran Integration Suite contains a map conversion wizard to enable you to easily import Gentran:Server for UNIX maps into the Map Editor, converting them to the Gentran Integration Suite format.

> **Note:**   This utility only converts maps created with the Gentran:Server for UNIX Visual Mapper version 5.x and above.

For other tips about migrating from Gentran:Server for UNIX to Gentran Integration Suite, see Appendix C, *Map Editor Migration Information*.

To convert Gentran:Server for UNIX maps:

1.   Select **File > Import > Gentran:Server for UNIX Maps** and click **Next**.

2.   Type the **name** of the Gentran:Server for UNIX Visual Mapper .map or .vmp map that you want to convert (or click **Browse** to locate the map or maps), and click **Next**.

> **Note:**   If you are converting a map created using Gentran:Server for UNIX version 5.1, 5.2, or 5.3, select a .vmp file. If you are converting a map created using Gentran:Server for UNIX version 6.1, select a .map file.

> **Tip:**   If you click **Browse**, you can select multiple maps

3.   Type the **name** of the destination folder where you would like the converted map to be written (or click **Browse** to locate the destination folder), and click **Next**.

4.   If you do not want the Map Editor to import the map links, clear the **Create links within imported maps** check box.

> **Note:**   Only simple links are converted.

5.   If you do not want the newly converted map to be automatically opened in the Map Editor, clear the **Open imported maps in Map Editor** check box.

6.   Click **Finish**. The utility converts the specified map and (if designated) opens the map in Gentran Integration Suite. If the file name already exists in the specified directory, the Map Editor renames the file with a numeral appended to the name, and saves the file. For example, if ORDER.map already exists in the specified directory, Map Editor renames the imported file ORDER1.map, or if ORDER1.map already exists in the directory, Map Editor renames the map ORDER2.map.

7. Add the necessary map links, standard rules, extended rules, conditional relationships, and user exits to the converted map.

    For more information about adding additional map links, see *Creating Simple Links* on page 57.

    For more information about adding standard rules, see Chapter 8, *Using Standard Rules*.

    For more information about adding extended rules, see Chapter 9, *Using Extended Rules*.

    For more information about adding conditional relationships, see *Defining and Modifying Relational Conditions* on page 76.

    For more information about adding user exits, see Chapter 11, *Using User Exits*.

Converting Gentran:Server for iSeries Maps to Gentran Integration Suite

Gentran Integration Suite contains a map conversion wizard to enable you to easily import Gentran:Server for iSeries maps into the Map Editor, converting them to the Gentran Integration Suite format.

Note: This utility only converts *internally described* applications and maps copied from Gentran:Server for iSeries, including Gentran:Server for AS/400 3.0 and 3.1, Gentran:Server iSeries 3.2, and Gentran:Server for iSeries 3.3.

To convert Gentran:Server for iSeries maps:

1. In Gentran:Server for iSeries, prepare the Gentran:Server for iSeries application and map files you want to convert by performing the steps listed in *Gentran:Server for iSeries* on page 406.

2. After you copy the files to your machine, select **File > Import > Gentran:Server iSeries Maps** and click **Next**.

3. Type the **source folder name** of the folder where the exported iSeries maps are located (or click **Browse** to locate the source folder), and click **Next**.

4. Type the **destination folder name** of the destination folder where you would like the converted maps to be written (or click **Browse** to locate the destination folder), and click **Next**.

5. Select the maps you want to import and click **Next**. By default, all maps in the source folder are selected.

6. If you do not want the newly converted map or maps to be automatically opened in the Map Editor, clear the **Open imported maps in Map Editor** check box.

7. Click **Finish**. The utility converts the maps in the specified folder and (if designated) opens the maps in Gentran Integration Suite. The Map Editor creates two copies of each imported map, one for receiving (renamed with an R appended to the file name) and one for sending (renamed with a S appended the file name). For example, if you import a file named INVOICE, the Map Editor creates one map named INVOICER (for receiving) and another named INVOICES (for sending).

8. Add the necessary map links, standard rules, extended rules, conditional relationships, and user exits to the converted maps.

    For more information about adding map links, see *Creating Simple Links* on page 57.

    For more information about adding standard rules, see Chapter 8, *Using Standard Rules*.

    For more information about adding extended rules, see Chapter 9, *Using Extended Rules*.

    For more information about adding conditional relationships, see *Defining and Modifying Relational Conditions* on page 76.

    For more information about adding user exits, see Chapter 11, *Using User Exits*.

Converting Gentran:Basic for zSeries Maps to Gentran Integration Suite

Gentran Integration Suite contains a map conversion wizard to enable you to easily import Gentran:Basic for zSeries maps and convert them to the Gentran Integration Suite format.

> **Note:**   This utility only converts maps *exported* from Gentran:Basic for zSeries version 6.0 or 6.1 (Gentran:Basic for MVS 6.0 or Gentran:Basic for OS/390 6.1).

To convert Gentran:Basic for zSeries maps:

1. In Gentran:Basic for zSeries, export the maps you want to convert to a single folder.

    For information about exporting maps, see *Converting Maps Using the Command Line* on page 403.

2. Select **File > Import > Gentran:Basic zSeries Maps** and click **Next**.

3. From the **Product Version** list, select the zSeries product version used to create and export the maps.

4. Type the **source folder name** of the folder where the exported zSeries maps are located (or click **Browse** to locate the source folder), and click **Next**.

5. Type the **destination folder name** of the destination folder where you would like the converted maps to be written (or click **Browse** to locate the destination folder), and click **Next**.

6. Select the maps you want to import and click **Next**. By default, all maps in the source folder are selected.

7. If you do not want the newly converted map to be automatically opened in the Map Editor, clear the **Open imported maps in Map Editor** check box.

8. Click **Finish**. The utility converts the maps in the specified folder and (if designated) opens the maps in Gentran Integration Suite. The Map Editor creates two copies of each imported map, one for receiving (renamed with an R appended to the file name) and one for sending (renamed with a S appended the file name). For example, if you import a file named INVOICE, the Map Editor creates one map named INVOICER (for receiving) and another named INVOICES (for sending).

9. Add the necessary map links, standard rules, extended rules, conditional relationships, and user exits to the converted maps.

For more information about adding map links, see *Creating Simple Links* on page 57.

For more information about adding standard rules, see Chapter 8, *Using Standard Rules*.

For more information about adding extended rules, see Chapter 9, *Using Extended Rules*.

For more information about adding conditional relationships, see *Defining and Modifying Relational Conditions* on page 76.

For more information about adding user exits, see Chapter 11, *Using User Exits*.

Converting Maps Using the Command Line

You can also convert Gentran:Server for UNIX, Gentran:Server for iSeries, and Gentran:Basic for zSeries maps using the command line.

The zSeries or iSeries map files must be named:

✦ APPTLR.izm

✦ APPHDR.izm

✦ APPRCT.izm

✦ APPFLD.izm

✦ TRNHDR.izm

✦ TRNSEG.izm

✦ TRNELE.izm

A very specific set of steps must be followed to create these files on the Gentran:Basic for zSeries and Gentran:

Server for iSeries systems. For information about how to create these files, see *Creating the zSeries and iSeries Map Files* on page 404.

The topics in this section are:

✦ Creating the zSeries and iSeries Map Files

◆ Syntax

◆ Examples

Creating the zSeries and iSeries Map Files

This section contains the steps you need to follow to create the necessary map files prior to executing the command line map conversion utility.

Gentran:Basic for zSeries 6.1

To create the map files for Gentran:Basic for zSeries version 6.1:

1. Back up the following VSAM files to a sequential format to convert your application definitions:

   ◆ GENTRAN.V6X1.VSAM.APPL.HEADER

   ◆ GENTRAN.V6X1.VSAM.APPL.RECORD

   ◆ GENTRAN.V6X1.VSAM.APPL.FIELD

   ◆ GENTRAN.V6X1.VSAM.APPL.LINK

2. Use ASCII mode to FTP the sequential files listed above from the mainframe to your Windows computer. These files must be named as follows:

   ◆ APPHDR.izm

   ◆ APPRCT.izm

   ◆ APPFLD.izm

   ◆ APPTLR.izm

3. Back up the following VSAM files to a sequential format to convert your maps:

   ◆ GENTRAN.V6X1.VSAM.TRANS.HEADER

   ◆ GENTRAN.V6X1.VSAM.TRANS.SEGMENT

   ◆ GENTRAN.V6X1.VSAM.TRANS.ELEMENT

4. Use ASCII mode to FTP the sequential files listed above from the mainframe to your Windows computer. These files must be named as follows:

   ◆ TRNHDR.izm

   ◆ TRNSEG.izm

   ◆ TRNELE.izm

Gentran:Basic for zSeries 6.0

Mapping and Application Definition Conversion

If you are migrating your maps and applications from a Gentran:Basic 6.0 environment to a Gentran Integration Suite environment, you must run utility programs to clean up filler

space in these mapping files in order successfully transfer the files to your Windows computer and run the Gentran Integration Suite map conversion process.

The first step to converting your maps and application structures to a Gentran Integration Suite format is to copy your VSAM transaction map and application files to a sequential format. If you are running Gentran:Basic 6.0, you can run the two jobs provided. These jobs copy the necessary files to a sequential format and also initialize all filler areas within the records of these files.

Job - INMAP60

Job INMAP60 executes program EBDINIM. This program reads the VSAM transaction mapping files, initializes all filler space in the records, and writes out sequential versions of the transaction mapping files. These sequential transaction mapping files should be used to transfer to your Windows computer for the Gentran Integration Suite mapping conversion process.

You must specify whether you want all your transaction maps initialized at one time or if you want to select a specific map or a set of maps (a range of maps based on the range of keys in the mapping files). Please review the comments in the INMAP60 JCL to determine how to initialize all transaction maps or a selection of transaction maps.

Job - INAPP60

Job INAPP60 executes program EBDINIA. This program reads in the VSAM application files, initializes all filler space in the records, and writes out sequential versions of the application files. These sequential application files should be used to transfer to your Windows computer for the Gentran Integration Suite mapping conversion process.

You must specify whether you want all your application definitions initialized at one time or if you want to select a specific application or a set of application definitions (a range of applications based on the range of keys in the application files). Please review the comments in the INAPP60 JCL to determine how to initialize all applications definitions or a selection of application definitions.

To create the map files for Gentran:Basic for zSeries version 6.0:

Note: The JCL/programs to run INMAP60 and INAPP60 are not part of the Gentran:Basic product. Contact Gentran:Basic customer support to receive these programs.

---

1. Run INAPP60.
2. Use ASCII mode to FTP the sequential files generated by INAPP60 from the mainframe to your Windows computer. These files must be named as follows:
   - APPHDR.izm
   - APPRCT.izm
   - APPFLD.izm
   - APPTLR.izm
3. Run INMAP60.

4. Use ASCII mode to FTP the sequential files generated by INMAP60 from the mainframe to your Windows computer. These files must be named as follows:

   ◆ TRNHDR.izm

   ◆ TRNSEG.izm

   ◆ TRNELE.izm

Gentran:Server for iSeries

You must perform the following steps on the iSeries machine prior to running the conversion on your Windows computer. Step 6 and 7 take the necessary map, application definition files for Gentran:Server for iSeries 3.0, 3.1, 3.2, and 3.3, unpacks the fields and puts them into the format needed to complete any GIS conversions.

To create the map files for Gentran:Server for iSeries version 3.0, 3.1, 3.2, and 3.3:

1. Run the **CNVTGIS** conversion utility to prepare the files to imported. For information about this conversion utility, see the **CNVTGIS.doc** instructions located in the iSeries folder on your product CD or perform the following steps.

2. Create a library to hold the conversion utilities on your iSeries by keying in the following command and the press **ENTER**:

```
CRTLIB CNVTGIS
```

   If the library already exists, key the following command instead and then press **ENTER**:

```
CLRLIB CNVTGIS
```

3. Create a save file in the library specified above by typing **CRTSAVF FILE(CNVTGIS/CNVTGIS)**

4. Copy the save file from the *<installdir>* to the newly created save file, using the following commands:

```
CPYFRMSTMF FROMSTMF('<installdir>/cnvtgis.savf')
TOMBR('/QSYS.LIB/CNVTGIS.LIB/CNVTGIS.FILE') MBROPT(*REPLACE)
```

5. Restore the conversion objects with the following command: **RSTLIB SAVLIB(CNVTGIS) DEV(*SAVF) SAVF(CNVTGIS/CNVTGIS)**

6. Add the CNVTGIS library to your library list with the following command: **ADDLIBLE CNVTGIS**

7. Run the copy utility to prepare your files for the conversion; to do so, type **CPYTOGIS** and press **F4**.

8. Type the parameters and press **Enter**. Press **F1** on any parameter to access further information for that parameter.

9. After the CPYTOGIS is completed, copy the APPFLD, APPHDR, APPRCT, APPTLR, TRNELE, TRNHDR, TRNSEG files from the IFS directory keyed on the CPYTOGIS command to a folder on your machine.

10. Once the files are copied, rename the 7 files so that they have an IZM file extension. For example, rename APPFLD to APPFLD.izm (with the .izm extension).

Syntax

From the command line, use this syntax to convert maps from the command line:

```
MAPPER.EXE -m version mapLocation [SingleMapName[S|R]]

Values for iSeries and zSeries:
          version     -   Enter the version of Gentran in which the maps were
                          created. Valid values for zSeries are 6.0 and 6.1.
                          Valid value for iSeries is 3.0, 3.1, 3.2, or 3.3.

          mapLocation  -  Enter the full path to the directory on Windows
                          where the Gentran zSeries or iSeries maps reside.
                          The path should include a wildcard for the file
                          name. The directory should only contain the
                          maps to be converted and no other files.

          [SingleMapName] Name of a single map you wish to convert.

          [S|R]       -   Direction of map ('S'=Send, 'R'=Recv)

Value for UNIX only:
          -U          -   UNIX conversion, followed by a fully qualified file
                          path or by a file path and *.vmp or *.map.
```

Examples

This is an example in which one zSeries map only is converted. The translation name is **TDANA001DV** and the direction is **Receive**.

```
mapper.exe -m 6.1 c:\mfmaps\*.izm TDANA100DVR
```

This is an example in which all of the maps in the specified directory are converted:

```
mapper.exe -m 6.1 c:\mfmaps\*.izm
```

COBOL Copybook Conversion for Use with Map Editor

This section covers the following topics:

✦ Overview
✦ Checklist of User Tasks
✦ Creating a Map Using a COBOL Copybook
✦ Troubleshooting for COBOL Copybook Conversions

Overview

Gentran Integration Suite allows you to select a COBOL copybook file layout (in ASCII text) as input to the Map Editor for the positional side of a map. The conversion of COBOL copybook is accomplished using the third-party open source tool CB2XML (Copy-Book-to-XML), which is a COBOL copybook to XML converter. CB2XML accepts a COBOL copybook file layout as input and returns an XML file as output, and the Map Editor then converts the intermediate XML file to the positional format.

> **Note:** The Map Editor does not convert any programming or data manipulations (equivalent to the Gentran Integration Suite standard and extended rules); just the file layout.

Prerequisite Knowledge and Tools

The following knowledge and tools are necessary to use COBOL copybooks with Gentran Integration Suite:

✦ The audience using this software should be familiar with Gentran Integration Suite, their operating system, and the use of COBOL copybooks.

Note: This documentation is not intended to explain COBOL or COBOL copybooks.

✦ You must have a Java Runtime Environment version 1.4.2_06 installed on the same machine as Map Editor. You can download the 1.4.2_06 version of JRE from:
http://java.sun.com/products/archive/

If you do not have JRE version 1.4.2_06 installed on the same machine as Map Editor, you will receive a message instructing you to install JRE version 1.4.2_06 and try the COBOL copybook conversion again. The message you receive if you do not have either installed is:

```
Error: Required JRE version 1.4.2_06 was not found. Please
install the proper version.
```

Note: The Java JDK contains the correct version of JRE. However, if you install only JRE, it will use less disk space than installing JDK.

✦ You must have the Microsoft XML Core Services (MSXML) 4.0 installed on the same machine as Map Editor. You can download this version of MSXML from:
http://www.microsoft.com/downloads/details.aspx?FamilyID=3144B7
2B-B4F2-46DA-B4B6-C5D7485F2B42&displaylang=en

If you do not have Microsoft XML Core Services (MSXML) 4.0 installed, you will receive a message instructing you to install MSXML 4.0 on the same machine as Map Editor and try the COBOL copybook conversion again.

✦ The CB2XML tool is installed (by default, unless you specify another location) in the following subfolder along with the necessary files when you install the Map Editor on your machine: **Program Files\Sterling Commerce\Map Editor\3rdParty\cb2xml**

Note: The cb2xml child directories are set by the Gentran Integration Suite install and cannot be changed.

✦ When the conversion of the COBOL copybook is complete and Map Editor has created the positional side of your map, it leaves behind the intermediate XML file (using the same name of your COBOL copybook with the extension .XML appended) in the same folder as the source file. You can view this XML file in Internet Explorer.

Note: You can use this intermediate XML file to create a schema, if you wish, or you can just delete it if you have no use for it.

Notes to be Aware of Prior to Implementing COBOL Copybook

Conversion

In the Gentran Integration Suite implementation of COBOL copybook conversions, the following notes apply:

- ◆ The description of data in a COBOL copybook contains hierarchical levels. To preserve these levels, the Map Editor converts the top levels into groups. Additionally, records are created when necessary to preserve the necessary map component hierarchy. If a record is created to contain a field, the Map Editor uses the field name and appends the suffix "_record" to use as the name of the generated record. This concept is illustrated in the diagram below:

<u>COBOL Copybook</u>   <u>Map Editor</u>

Record   Group

→ Field   → Record (named
                                 [FIELDNAME_record])

→ Record   → Field
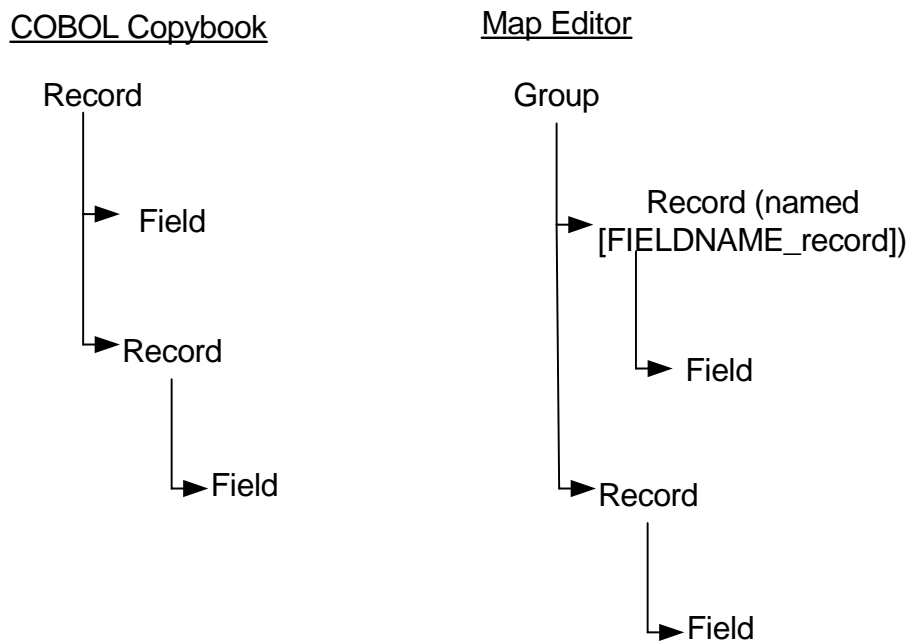
→ Field   → Record

                                → Field

- ◆ When possible, comments from the COBOL copybook are added to the Additional Notes section of the Name tab of the applicable map component Properties dialog box (this may be Group, Positional Record, or Field Properties).

This diagram illustrates an example of an Additional Notes section that contains a comment from the original copybook:
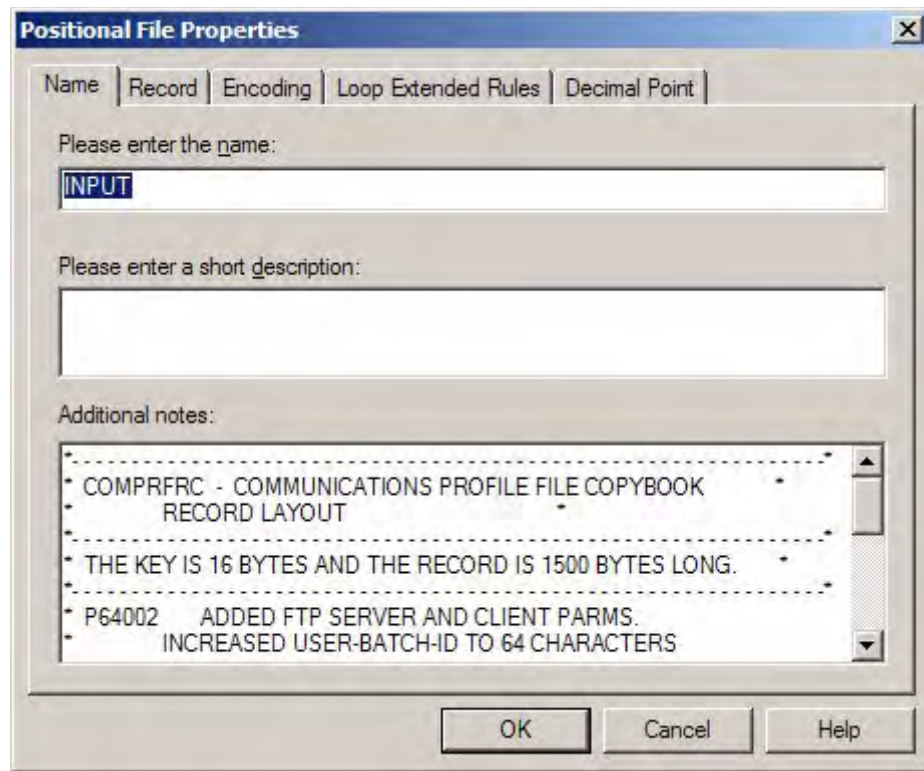
Note: The maximum length allowed for any Map Editor Description is 71 characters. To configure CB2XML to allow characters to be converted from COBOL to XML beyond this character limit in the COBOL source file, you need to create a **cb2xml.properties** file. See *Creating a CB2XML.properties File if Descriptions Exceed the Maximum Length* on page 418.

◆   You are notified during the Map Wizard conversion of the COBOL copybook if redefine statements are present in the copybook file. REDEFINE is a COBOL term indicating that the parameter is redefining data space. Additionally, this information is added to the Description section of the Name tab of the applicable map component Properties dialog box (this may be Group, Positional Record, or Field Properties). When a map component is redefined, the description ***REDEFINED*** is added to the map component. When a map component redefines another map component, the description **REDEFINES: [NAME_OF_MAP_COMPONENT]** is added to the map component. This

diagram illustrates an example of what redefined map components would look like in a map:
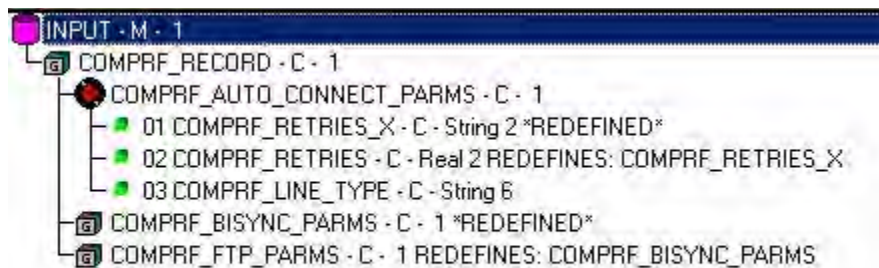
Note: The maximum length allowed for any Map Editor Description is 71 characters. To configure CB2XML to allow characters to be converted from COBOL to XML beyond this character limit in the COBOL source file, you need to create a **cb2xml.properties** file. See *Creating a CB2XML.properties File if Descriptions Exceed the Maximum Length* on page 418.

✦ In general, the Map Editor converts COBOL copybook data types to Map Editor data types as described in this table:

COBOL Copybook Data Type	Map Editor Data Type
Packed decimal number	Packed decimal number
String	String
Implied decimal number	Implied decimal number
Real decimal number	Real decimal number
Signed zone decimal number	Overpunched number
Computation-3	Packed decimal number

However, the supported data types in the Map Editor are string, number, and date/time (the formats that are supported for positional file formats). See *Formatting Data in Fields* for a complete description of these data formats. Therefore, if an unsupported data format is present in the copybook file (such as the COMP format), the Map Editor converts it to the string format and indicates this in the Description and Additional Notes section of the Name tab of the applicable map component Properties dialog box (this may be Group, Positional Record, or Field Properties). This diagram illustrates

an example of Description and Additional Notes sections that contain information about an unsupported data format from the original copybook:
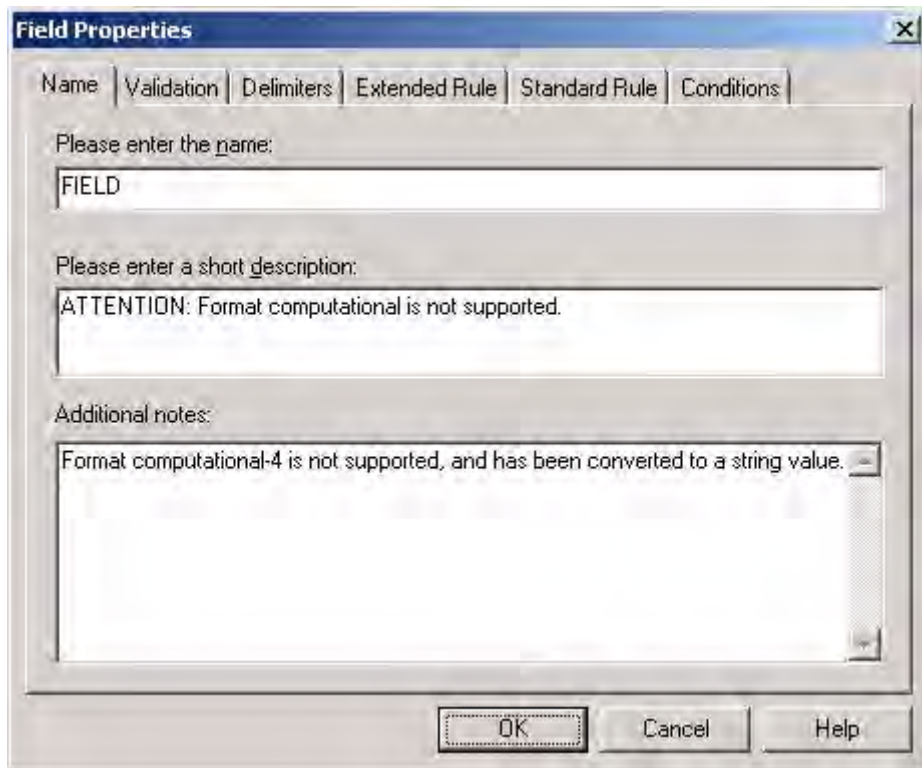
Note: The maximum length allowed for any Map Editor Description is 71 characters. To configure CB2XML to allow characters to be converted from COBOL to XML beyond this character limit in the COBOL source file, you need to create a **cb2xml.properties** file. See *Creating a CB2XML.properties File if Descriptions Exceed the Maximum Length* on page 418.

✦ The following are examples of formats that may appear in a COBOL copybook that are not supported in Gentran Integration Suite:

- ◆ Computational (COMP)
- ◆ Computational-4 (COMP-4)
- ◆ Binary
- ◆ Index
- ◆ Pointer
- ◆ Computational-1 (COMP-1)
- ◆ Computational-2 (COMP-2)

◆ Exponential numerics

Note: Any fields in the COBOL copybook that use Computational-3 (COMP-3) format are automatically converted to the appropriate Gentran Integration Suite number (packed decimal) format.

✦ The Map Editor only needs the record layout of your copybook file.

✦ The Map Editor only converts the copybook file layout and does *not* convert any standard or extended rules.

✦ The input accepted by Map Editor is a complete COBOL copybook file layout in ASCII text.

✦ Any dashes in the COBOL copybook are converted to underscores by the Map Editor.

✦ You must follow the rules of COBOL (that is, your copybook must not contain errors or Gentran Integration Suite will be unable to parse it correctly).

Checklist of User Tasks

The following is a checklist of the tasks you will need to complete to convert a COBOL copybook to create the positional side of a map in Map Editor:

Task Number	Description of Task	Completed
1	Ensure that you have a complete COBOL copybook and have downloaded the copybook from your system (via FTP) to the machine on which Map Editor is installed. **Note:** The Map Editor only processes the record layout of the copybook file, so you should only provide the record layout.	**X**
2	Create a map using a COBOL copybook. See *Creating a Map Using a COBOL Copybook* on page 416.	
3	If you have unsupported data formats or REDEFINES in your copybook, their description by Map Editor may exceed the limit allowed. If this occurs, you need to create a file named **cb2xml.properties**. See *Creating a CB2XML.properties File if Descriptions Exceed the Maximum Length* on page 418.	
4	Set record tags for each record on the Positional Record Properties dialog box, delete the accompanying field, and run Auto Position on the fields in each record.	
5	Map redefined components, using conditions or other mapping operations.	
6	Select the appropriate data format for map components from the COBOL copybook that were in an unsupported format and were automatically converted to the string format.	
7	Create any other necessary standard rules, extended rules, and links.	
8	Check the format of the completed map to ensure that it mirrors your COBOL copybook.	

Creating a Map Using a COBOL Copybook

Each map has two sides. Each side represents a data format; the input side is what you are translating from and the output side is what you are translating to. Each data format has specific map components.

> **Note:**   If an error occurs while you are creating a map, you need to check your COBOL copybook for errors.

To create a map:

1. From the Map Editor **File** menu, select **New**.

2. In the New Map wizard, answer the following questions and then click **Next**.

    ◆   What kind of map are you creating?

        Accept the default, Sterling Integrator.

    Caution: Always accept the Sterling Integrator default when you are using the Gentran Integration Suite product.

    Note: The other map options enable you to import maps from other Sterling Commerce products in order to convert them to Sterling Integrator-type maps.

    ◆   What is the name of the map?

        Type the unique name of the map. The Map Editor adds the default .mxl extension.

    ◆   What is your name?

        Type your name if it differs from the user name prompted by the New Map wizard. The New Map wizard displays Input Format fields. You must complete the format of the input side of the map. This is the format of the data that is translated by Gentran Integration Suite.

3. In the Input Format window, specify how you want to define the data format by selecting one of the following:

    ◆   Create a new data format using this syntax

        Select a format. If you are converting a COBOL copybook for the input side of the map, select the **Positional** format and click **Customize**.

    ◆   Load the data format from a saved definition

        Type the path and file name of the saved definition (.ddf or .ifd extension).

        Click **Browse** to display the **Open File Definition** dialog box.

    Note: During the conversion process, if the wizard encounters any REDEFINE statements, you are notified that redefine statements are present in the converted copybook file. Additionally, you are notified if the wizard encounters any unsupported data types (such as exponential numbers or binary data).

4. If you are converting a COBOL copybook for the input side of the map, select the name of the COBOL copybook file (using **Browse**) or type a URL pointing to the file and click **Next**.

5. Click **Finish**.

6. In the Output Format window, specify how you want to define the data format by selecting one of the following:

   ◆ Create a new data format using this syntax

     Select a format. If you are converting a COBOL copybook for the output side of the map, select the **Positional** format and click **Customize**.

   ◆ Load the data format from a saved definition

     Type the path and file name of the saved definition (.ddf or .ifd extension).

     Click **Browse** to display the **Open File Definition** dialog box.

   Note: During the conversion process, if the wizard encounters any REDEFINE statements, you are notified that redefine statements are present in the converted copybook file. Additionally, you are notified if the wizard encounters any unsupported data types (such as exponential numbers or binary data).

7. If you are converting a COBOL copybook for the output side of the map, select the name of the COBOL copybook file (using **Browse**) or type a URL pointing to the file and click **Next**.

8. Click **Finish**.

9. Click **Finish** to create the map. The map opens in the Map Editor window.

   Note: If the map contains REDEFINE statements, a message box is displayed noting that the REDEFINES statements present in the file, [filename], are converted but will require manual attention when the new map wizard is complete.

10. In the Map Editor, select **File** > **Save** to save the map. Do not use spaces or apostrophes in the map name. To save a map as a .map file, select **File > Save As** and then select **Source Maps (*.map)** from the **Save as type** list.

Tasks you Must Complete After Creating a Map with a Copybook

After you have created a map using a COBOL copybook, you must complete the following tasks:

✦ Set record tags for each record on the Positional Record Properties dialog box, delete the accompanying field, and Auto Position the fields in each record. See *Modifying Records After a COBOL Copybook Conversion* on page 419.

✦ Map redefined components, using conditions or other mapping operations. See *Using Standard Rules*, *Using Extended Rules,* and *Defining and Modifying Relational Conditions*.

✦ Select the appropriate data format for map components from the COBOL copybook that were in an unsupported format and were automatically converted to the string format. See *Formatting Data in Fields*.

♦ Create any other necessary standard rules, extended rules, and links. See *Creating Simple Links*, *Using Standard Rules*, and *Using Extended Rules*.

♦ Check the format of the completed map to ensure that it mirrors your COBOL copybook.

Creating a CB2XML.properties File if Descriptions Exceed the Maximum Length

The maximum length allowed for any Map Editor Description (used if there are REDEFINES and unsupported data formats in the copybook file) is 71 characters. To configure CB2XML to allow characters to be converted from COBOL to XML beyond this character limit in the COBOL source file, you need to create a file named **cb2xml.properties** that contains a **column.end** property that will be used when CB2XML is invoked to convert the copybook. When CB2XML is invoked, it searches for the cb2xml.properties file and, if CB2XML finds this file, it uses the properties in the file to specify to CB2XML how wide the lines are for conversion to XML and subsequent import into the Map Editor.

You will want to create a **cb2xml.properties** file if, for example, the Map Editor Description values are *less than* 71 characters and appear to have been truncated (or are incomplete). If this occurs, it would be appropriate for you to create a **cb2xml.properties** file to increase the number of characters per line that is read by CB2XML. However, if the Map Editor Description values are already 71 characters in length and appear to have been truncated, you would not create a **cb2xml.properties** file because the Map Editor maximum Description length has already been reached (and thus the **cb2xml.properties** file would be superfluous).

Note: Regardless of what you set the column.end parameter to in the cb2xml.properties file, Map Editor uses only the first 71 characters for the Description or Additional Notes parameters.

To create a **cb2xml.properties** file:

1.   Using a text editor, create a file named **cb2xml.properties**.

2.   Save this file in the same directory as the COBOL copybook file that you will convert and import.

3.   Add the following contents to the properties file:

```
column.start=6
column.end=XX
```

Note: Change XX to specify to CB2XML how wide the lines are that it is converting from copybook source to XML.

4.   Save the **cb2xml.properties** file in the same directory where the COBOL copybook source file you want to convert is located.

Modifying Records After a COBOL Copybook Conversion

For each record on the positional side of your map created using a COBOL copybook, you need to set the record tags for each record on the Positional Record Properties dialog box, delete the accompanying field (containing the original tag), and Auto Position the fields in each record.

Note: You must either specify the start position of each field or use the Auto Position function; otherwise, the translator will not be able to read or write the record correctly.

To complete the tasks necessary on each record on the positional side of your map:

1. In the Map Editor, right-click the first positional record. From the shortcut menu, select **Edit Fields**.

    The Positional Field Editor dialog box opens.

2. Note the name, start position (Start), and length (Len) of the field in this record that currently contains the record tag, and click **OK**.

3. Right-click the first positional record again. From the shortcut menu, select **Properties**.

    The Positional Record dialog box opens.

4. Select the **Tag** tab to access the tag options.

5. In the **Tag** box, type the record tag (for example, **__ABC**). This is the name of original field that you noted in step 2.

6. In the **Position** box, type the start position of the record tag. This is the start position that you noted in step 2.

7. In the **Length** box, type the length of the record tag. This is the length that you noted in step 2.

8. Click **OK** to close the Positional Record Properties dialog box.

9. Right-click the first positional record again, and from the shortcut menu select **Edit Fields**.

    The Positional Field Editor dialog box opens.

10. Click **Auto Position**.

    Auto Position automatically calculates the start position in the record of each field, using the criteria that each field is positioned directly after the previous field and is of the length specified in the Max Length box. Click **Yes** to acknowledge the warning message that fields are sequenced in order.

    Note: The Auto Position function is valid only if you define a record tag and if you define every field in the record in the sequence that each field occurs. It is generally recommended that you run Auto Position, but there are circumstances in which it is not recommended. For example, if you have a record that starts at position 155 and contains 5 fields of 5 characters each, if you run Auto Position on this record it will reset the starting position of the first field from 155 to 1.

---

11. Click **OK** to close the Positional Field Editor dialog box.

12. Repeat steps 1 through 11 for each record on the positional side of your map.

Troubleshooting for COBOL Copybook Conversions

Why is a field not appearing in my map?

The most probable explanation is that this field is defined in your COBOL copybook in a column beyond the accepted range of 7-72.

Check the intermediate XML file using Internet Explorer (the intermediate XML file will be located in the same directory as the source copybook and will be named the same as the source file with a .XML extension. If the field definition is outside the 7-72 range, edit it so it falls within that range and recreate your map.

Index

A

abbreviated reference format 359

abstract element
  and product 133
  definition 131

accum function, extended rule 160, 211

accumulator operations
  add primary to alternate 161
  decrement primary 160
  divide by primary 161
  divide primary by alternate 161
  divide primary by field 161
  hash sum in primary 161
  increment primary 160
  load primary 161
  modulo primary with alternate 161
  modulo with field 161
  modulo with primary 161
  move primary to alternate 161
  multiply primary by alternate 161
  multiply with primary 161
  negate primary 161
  sum in primary 160
  use primary 161
  zero primary 161

accumulators, using 159

activation of map components 71

add primary to alternate, Use Accumulator standard rule 161

alphabetical language reference
  accum 211
  atoi 208, 212
  aton 208, 213
  begin 213
  break 214
  cerror 208, 214
  collate 219
  concat 206, 221
  continue 222

count 209, 222
date 223
delete 209, 224
do 248
else 228
empty 209, 225
end 225
eof 226
exist 209, 226
get 204, 227
if 228
index 229
left 206, 230
len 208, 230
messagebox 231
mid 206, 232
ntoa 208, 232
numerrors 233
occurrencetotal 233
readblock 210, 234
resetoccurrencetotal 236
right 206, 237
select 237
set 204, 238
sort 239
strdate 206, 239
strstr 207, 241
sum 241
sumtotal 242
then 228
trim 207, 243
trimleft 207, 244
trimright 207, 245
unreadblock 210, 246
update 246
while 248
writeblock 210, 248

array 361, 368

array, extended rules 188

assignment statement
  datetime expression 202
  numeric expression 202

string expression 202

atoi function, extended rule 208, 212

aton function, extended rule 208, 213

attribute
  container object 131, 150
  creating 150
  managing 149
  object 150

attribute container, definition 131

auto get next cursor, SQL 121

auto get next row, SQL 120

Auto Position function
  definition 92, 419
  using 90, 92, 419

Auto Trim function 75

auto-incrementing maps, customizing 29

B

begin function, extended rule 213

binary data segments
  example 80
  using 80

break function, extended rule 214

Business Processes, enabling maps for 274, 277

C

cerror function, extended rule 208, 214

character encoding, definition 105

character range 40

Check Database Consistency function 127

checking in
  map versions 257, 276
  maps 274
  translation object 274
  XML encoder object 274

checking out
  maps 258, 277
  translation object 258, 277
  XML encoder object 258, 277

CII

data attributes 105
  encoding default 105
  importing maps from Gentran:Server for
      Windows 106
  preserving leading spaces 106
  specifying character sets 105

CII data format
  control tags 103
  creating layout from standard 103
  encoding 105
  format of message 102
  loop 104
  mapping to positional data format 106
  overview 101

CII documents, mapping 101

CII File root element 102

code list
  copying and pasting 169
  defining 168
  definition 166
  deleting 168
  exporting 169
  importing 169
  mapping code item descriptions 170
  mapping entry descriptions 170
  trading partner 172
  validating 170

codes, using 166

collate function, extended rule 219

Colours function, Map Editor 27

command line
  list of numeric map types 62
  using to change map type 61
  using to compile maps 60
  using to save maps 61

compatible rule execution 36

compile error messages 281

Compile function 59

compiled map 17

compiling maps
  in Map Editor 59
  using the command line 60

compiling, XML encoder object 63

D

E

exist function, extended rule 209, 226

export, map 260, 279

exporting
  resource tags 261
  resources 261, 262

expression 189

extended rule
  alphabetical language reference 210
  commands and functions 195, 210
  compatible execution with Gentran:Server for
      Windows 36
  conditional logic 204
  data types supported 188
  datetime expression 203
  declarations section 188
  defining 192
  defining a map component rule 193
  field level 190
  fseek migration information 350
  ftell migration information 350
  keywords 194
  migration information 350
  On Begin 190
  operators 196
  pre-session 189
  processing 189
  processing diagram 192
  readbyte migration information 350
  session 193
  statements section 189
  string conditions and functions 205
  symbols 197
  unsupported 351
  wild blocks 350
  writebyte migration information 350

F

field
  definition 98, 112
  positional 86

field-level extended rule 190

format specifiers, strdate, extended rule 240

fseek, migration information 350

ftell, migration information 350

fully qualified reference format 358

G

Generate Fields function
  use in Input Records, SQL 123
  use in Output Records, SQL 123

Generate UBFI function 82

Generic Envelope service
  available field names 251

Gentran:Server
  for Windows, compatible rule execution 36
  map type changed to Sterling Integrator using the
      command line 61
  maps, compiled using the command line 61
  maps, saved using the command line 61
  migration information 349
  TFD, using 106
  UNIX maps, using the command line 62
  Windows, changing the map function 36
  Windows, importing CII maps 106
  Windows, importing maps from 36
  Windows, using DBCS 90
  Windows, using record delimiters 89

Gentran:Server for UNIX
  synonym table by in value 174
  synonym table by out value 175

Gentran:Server maps, converting 400, 401, 402

get function, extended rule 204, 227

global colors, customizing 27

global confirmations, customizing 29

global display options, customizing 27

global fonts, customizing 28

group
  definition 68, 86, 98, 102, 110
  promoting 39

H

hash sum in primary, Use Accumulator standard
      rule 161

hash total, calculating 163

hierarchical levels (HL)
  examples of use 383
  introduction to 383
  using in maps 383

N

O

P

Q

overview 109

SQL documents, mapping 109

SQL Manager
about 112

SQL root element 110

standard imports 263

standard rule
Loop Count 159
Select 171
System Variable 155
Update 180
Use Accumulator 159
Use Code 166
Use Constant 156
using 155

standard version
update to another 72

statement record
definition 111
managing 116
Test SQL function 118

statements section, extended rule 189

strdate function, extended rule 206, 239

strdate, format specifiers 240

string
conditions and functions, extended rule 205
expression, assignment statement 202
supported data type 188
type 40

strstr function, extended rule 207, 241

sum function, extended rule 241

sum in primary, Use Accumulator standard rule 160

sumtotal function, extended rule 242

symbols
extended rule 197
syntax examples 197

synonym table by in value
definition 174
using 174

synonym table by out value
definition 175
using 175

syntax token
Asian languages 43
character range 40, 44
definition 40
deleting 44
double-byte character set (DBCS) 43
using in CII 106
Western European languages 42

System Variable standard rule 155

T

TDF, migration information 355

temporary fields
creating 93
when to use 93
where to use 93

temporary records
caveats that apply to 93
creating 93
when to use 93
where to use 93

temporary storage map components, using 24

Test Connection function, using in SQL to test data sources 116

Test SQL function, using to test statement records 118

TFD
properties 104
tags 104
types 104

then function, extended rule 228

time, extended rule syntax 204

tracking, migration tip 353

trading partner code list
available field names 250
definition 172
mapping 178
migration tip 353
unmapping 178
using 172

transaction data file (TDF) 23, 85

transaction data file (TDF), migration information 355

transaction register
available field names 251

U

V