User's Guide

July 18, 2002

PatternStream User's Guide

Finite Matters Ltd.
Suite B
3064 River Road West
Goochland, Virginia 23063
USA

July 18, 2002

Contents

Chapter 1: Introduction

Prerequisites for understanding the PatternStream application

You'll understand more quickly the way the PatternStream application works if you have some knowledge (or access to knowledge) of several areas:

- You need to be very familiar with the database that contains the data you want to extract. You should have basic knowledge of SQL syntax and should understand any limitations of the *ODBC*-based (open database connectivity) database your company uses. If you don't know these things, you should have access to someone in your company, such as a database administrator *(DBA),* who can help you construct queries and resolve any problems with your data.
- You need to understand Adobe® FrameMaker® well enough to design a template that will produce a document that meets your requirements. You must understand FrameMaker concepts such as flows, master pages, table formatting, and how paragraph and character styles are used.
- Although not required, you will find it much easier to understand the PatternStream application if you have some kind of computer programming background, especially object-oriented programming. The PatternStream application is a logic-driven application. If you've studied programming, you'll find it easier to learn how to construct a pattern set.

The PatternStream application

You use the PatternStream database publishing application to publish paper and electronic documents—including catalogs, directories, manuals, reports, and other pattern-based information—from a variety of databases and other data sources.

PatternStream is currently structured as an extension (or plug-in) to the FrameMaker application (see Figure 1 on page 12). After PatternStream is properly installed, you automatically load it when you start FrameMaker. PatternStream uses FrameMaker as a kind of back end formatting engine. Essentially, it extracts data from external data sources in a logically structured way and inserts this data into a FrameMaker document as text, creating paragraphs, tables and other structural elements as directed. It is as if the PatternStream module is building, or composing, the document similar to the way a manual user of

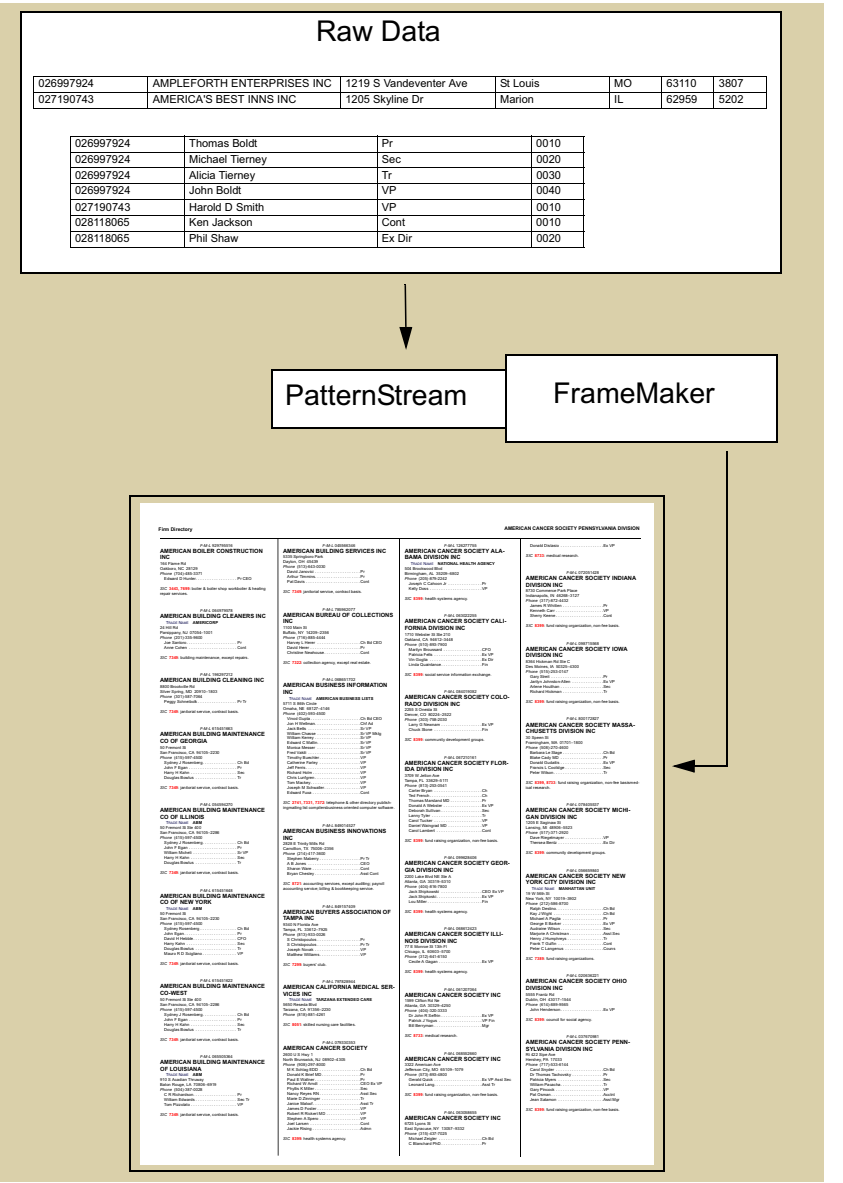


Figure 1: Basic PatternStream Architecture.

FrameMaker would do it. PatternStream is a graphical programming language designed for this type of data-driven composition. In order to provide as much flexibility as possible, you have access to a large collection of object types that are divided into ten distinct classes. The objects derived from these classes cooperate very closely with each other to collectively form the logic that is contained in a pattern set. These classes are summarized in Table 1.

Table 1: Object Classes in the PatternStream Application.

Object	Description
Pattern	Used to represent a collection of structural elements (paragraphs, table rows, etc.) that repeats. They are denoted in the PatternStream interface by the yellow folder icon. Patterns are associated with *queries* and contain *targets*.
Target	This is the largest class of objects. They are used to create the structural elements (tables, paragraphs, etc.) that are associated with patterns. They do not directly create textural content, but some targets can use attached string templates for this purpose.
Query	Query objects extract data from an outside data source as a series of records. Every pattern must be associated with a query object before it can invoke the targets it contains. A query can be executed as many times as the logic of the document layout requires.
Variable	*Variables* are object used to hold the data extracted by the queries you construct. They can also be used as iteration counters and can be assigned values just like a normal programming variable in more standard languages, such as C and Visual Basic.
Connection object	For every data source (ODBC database, tab-delimited ASCII file, etc.) from which you wish to extract information, you must create at least one connection object. Query objects use these connection objects to establish a transaction with the external data source.

Table 1: Object Classes in the PatternStream Application. (continued)

Object	Description
String Templates	These objects are used to create an instance of text. String Templates are composed of one or more segments, each individual segment creating a logical portion of the text.
Transforms	Objects that change the current value of a variable. Typical use of transforms is to perform simple mathematical operations on numeric variables or to perform string manipulations on text data.
Conditions	Condition objects evaluate to either TRUE or FALSE. They are used to make the generation of structural elements in the output document conditional on whether various data criteria are met.
Parameter Lists	Collections of variables used for specific purposes. For example, *argument lists*, which are a type of *parameter list,* can be used to pass values to a FrameScript.
Extensions	Miscellaneous objects or extensions to the basic PatternStream application. These objects include lookup tables (arrays), customized extensions written in the C++ language, FrameScript scripts, etc.

Types of Database Publishing

The term Database Publishing covers a wide diversity of techniques used to programmatically build output documents that are data-driven. Because this covers a wide variety of applications, it is important to distinguish among them the type of problem PatternStream was designed to solve.

Forms

This type of database publishing is a generalization of the paradigm used for typical report generation. These techniques are characterized by a pre-formatted background and specific areas within this background that are designated for variable data. The areas, or frames, can be either fixed in size or allowed to grow to accommodate the data. Because the formatting requirements for each instance of the form are minimal, this technique is very fast and usually easy to setup. On

the other hand, they have only a limited ability to handle large changes in the quantity of data for each page and varying data that has a complex structure. Examples of this type of system are PDF Forms and Crystal Reports.

Data Driven Publishing

These systems treat a document as a collection of fragments that must be assembled at the time of publishing. These fragments are managed through a customized database which, along with their location and type, can also contain layout information. Typically in these systems, each fragment is handled separately, with the layout being worked out by a page designer. More regular data, such as tables, are implemented using placeholders to determine the location of the table. The table is then generated either on demand or when the final document is published. These applications use a database publishing model derived from document management systems. Large portions of the application are usually concerned with workflow issues such as check in and checkout of fragments and the reuse of fragments in multiple documents. They are also designed on the assumption that a large number of users will be working on a document simultaneously.

Data Driven Composition

This is the database publishing paradigm that PatternStream was designed for. It assumes that the document to be published is composed of regular layout patterns and once these patterns are worked out, the entire document can be published at once. The scope of these patterns and subpatterns can range from a series of documents with similar structure to text fragments with a repeating pattern of data. These systems view the output document the same way a manual user of a desktop publishing application would view it.

Sometimes, an entire paragraph is built from variable data elements and constant text strings, including punctuation. In Figure 2, we have a hypothetical database where a particular person ID is associated with a series of states or provinces. The desired output is a sentence that lists all of the states that a person is registered to lobby in. For data driven composition, the configuration file (in PatternStream, these are called pset files) must contain, along with the SQL query itself, directions for building the entire sentence.This would include the logic for the placement of the commas, the substitution of a comma for the word “and” before the last state, and of course, a period at the end.

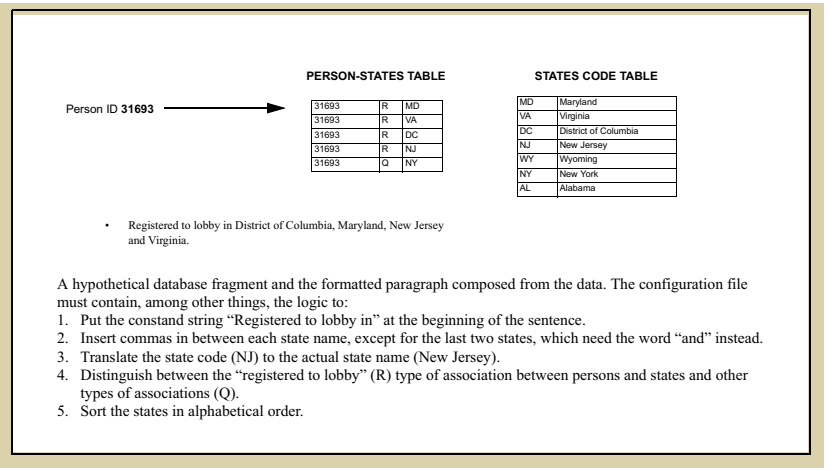


Figure 2: Data Driven Composition.

The Pattern Set Hierarchy

PatternStream uses a unique approach to solving the problem of data-driven composition by viewing a document first as two separate hierarchies, and then providing a method of merging these two disparate hierarchies together.

The Data Hierarchy

The first hierarchy deals with the logical structure of the data and how it is presented. Usually the layout logic reflects the underlying data structure, since it is the layout that visually organizes the information you wish to present. In Figure 3 on page 17 is a fragment of a directory published from a relational database and beside it are the nested select statements that were needed to extract that data. Notice how the layout reflects the logical organization of the data, which is mirrored in the nested sequence of the queries. The top level query retrieves information on each political action committee for a given company (represented by the COMP_ID variable). There is then a sub-query that retrieves contribution data for each political party for the current company and committee (represented by the COMP_ID and PAC_ID variables). At the same level, another query retrieves all of the political parties for which large contributions were made to individual legislators. For each party, a sub-query is performed for the senate members and then the house members. Notice how the layout style (hopefully) makes clear the relationship between all of this textural information.

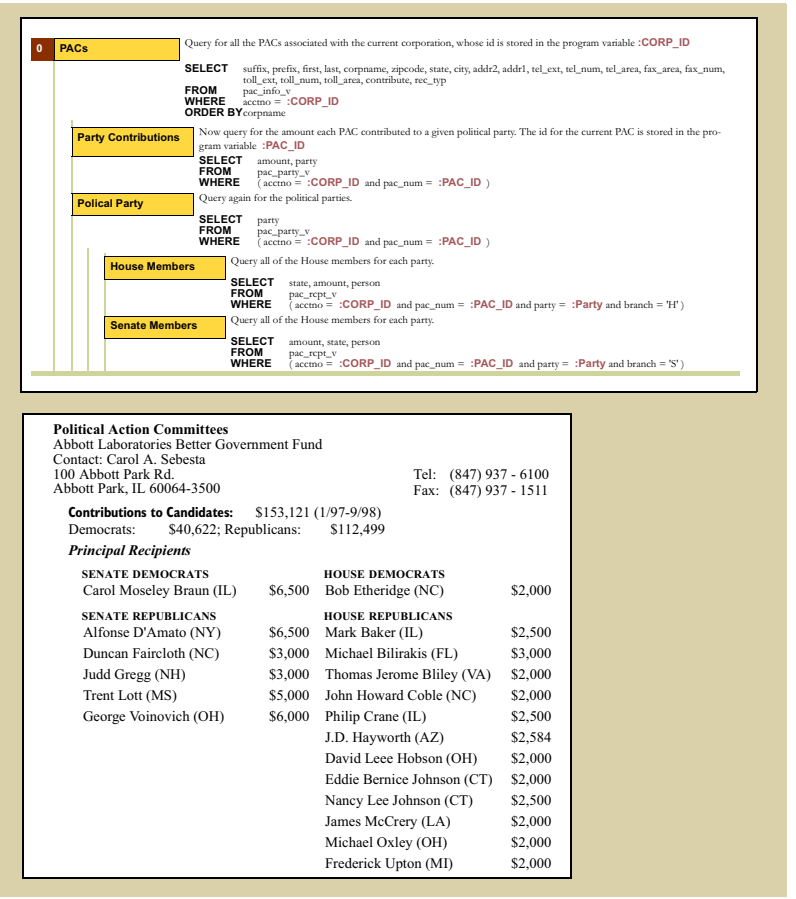


Figure 3: Data Hierarchy.

The Document Hierarchy

Documents can also be viewed as a logical hierarchy of structural elements; documents contain pages, pages contain paragraphs and tables, tables contain rows and rows contain cells. An example of document being viewed in this manner would be the DTD of an XML document type. In Figure 4, we have the same directory fragment, this time viewed as a collection of structural elements. Although the highlighted information represents only one level of the data hierarchy (it being retrieved by the top level query) notice how, for layout purposes, it has been implemented with multi-level structural elements, including paragraphs embedded in table cells.

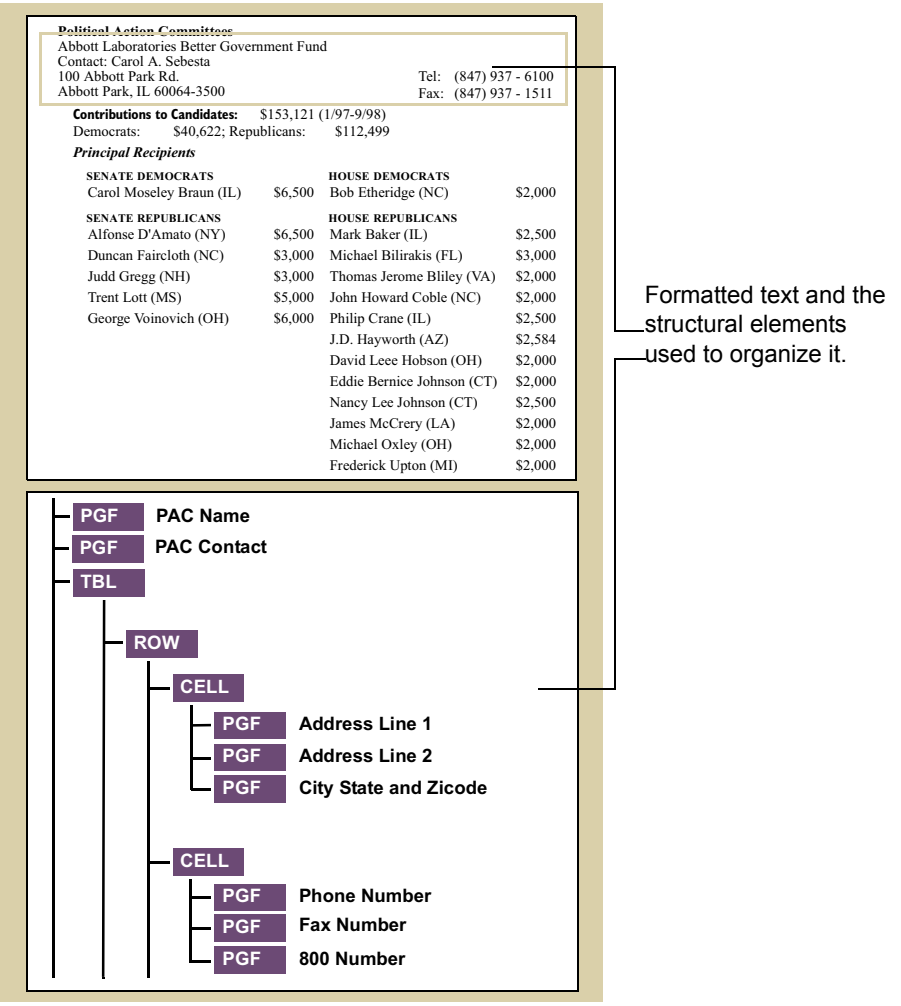


Figure 4: Document Hierarchy.

The Merged Hierarchy

In general, as the above example illustrates, the levels in the data hierarchy do NOT match one-for-one with the levels in the document hierarchy. PatternStream is built around a paradigm that uses both these natural ways of viewing the output, and provides a means of merging these two different views together. The method for doing this is a construct call the Pattern Set Hierarchy. Below is

the portion of a PatternStream configuration file (called a PSet file) that was used to generate the fragments above.

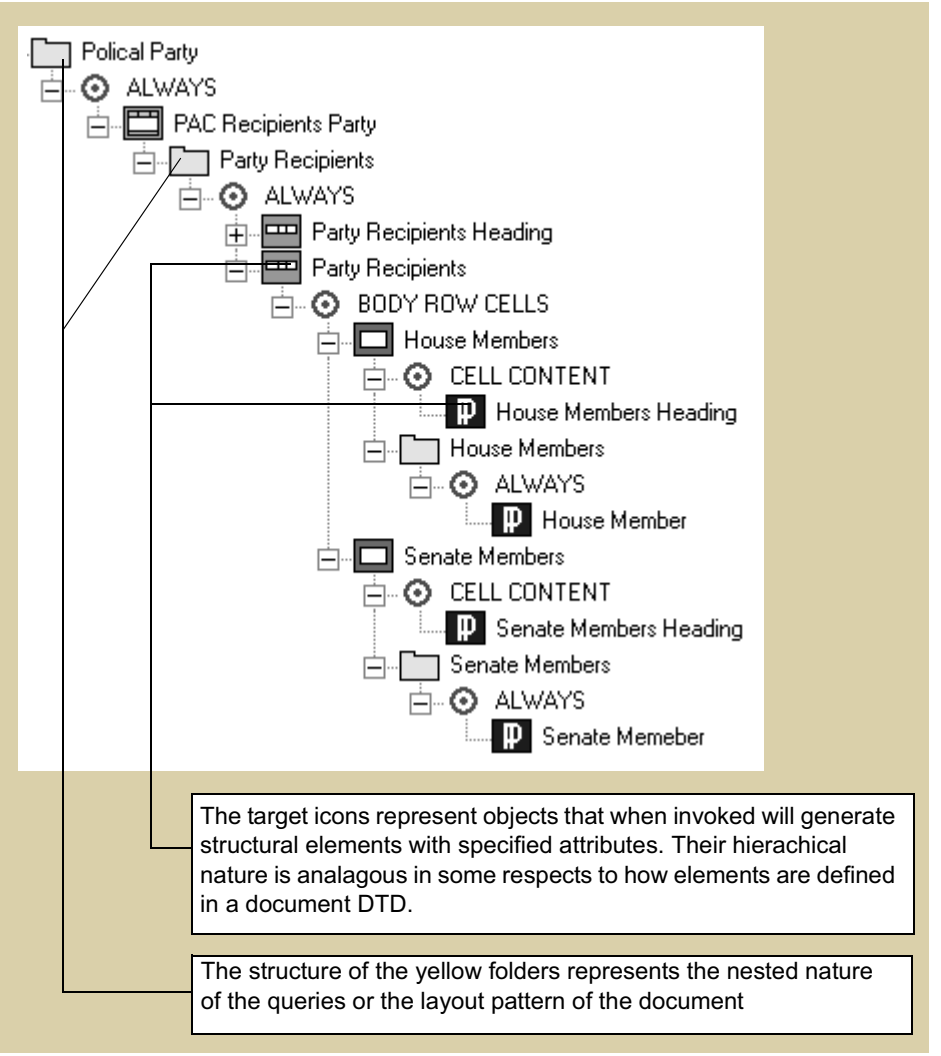


Figure 5: PSet Hierarchy.

Notice how the nested structure of the queries, as represented by the pattern objects (or the yellow folders) is made to mesh with the hierarchical structure of the target objects, which are used to generate instances of FrameMaker struc-

tural elements, such as tables, rows, and paragraphs. It is this merging of the two hierarchies that is the hardest thing to conceptualize when building a data driven composition project. And it is the PSet Hierarchy that not only makes this conceptualization easier but allows PatternStream to provide a graphical environment that uses re-locatable objects (patterns and targets) to build this hierarchy. Sub-queries, and the elements they create, can be moved inside and outside other structural elements, modifying the layout logic, by just clicking on the mouse a few times rather then having to rearrange subroutine calls and rewriting sections of code.

Chapter 2: PatternStream Basics

This chapter explains some of the basic ideas needed to work with PatternStream and additionally, provides a description of how the user interface is organized. As each major interface component is explained, the functionality it provides is also explained. PatternStream is an extension of the FrameMaker application. It uses FrameMaker as the formatting engine and acts as an intermediary between this formatting engine and an external data source, most likely one or more relational databases. Therefore, in order to launch PatternStream, you must first launch FrameMaker.

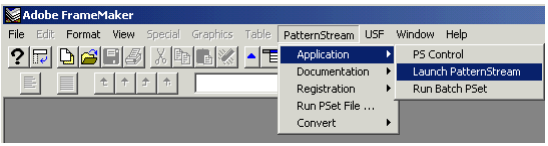


Figure 6: PatternStream menu in the FrameMaker Menu Bar.

When you launch FrameMaker, you will notice the PatternStream menu in the FrameMaker menu bar (Figure 6). This provides the primary access to the PatternStream application. It is important to realize that when you are running PatternStream you are also running FrameMaker.

Launching PatternStream

To launch the PatternStream application do the following:

1. First launch the FrameMaker application.
2. Select the PatternStream menu, select Application and then select Launch PatternStream.
3. This brings up the PatternStream main dialog, shown in Figure 7 on page 22.

Opening a PSet File

Once you have launched PatternStream, the next thing to do is to open a pset file that already exits. For this, we will open one of the examples that were installed along with the PatternStream application. It is recommended that you have these examples installed since they illustrate of many of the principles and techniques used in developing a pattern set. To open a pset file do the following:

1 Select the PSet menu from the PatternStream main dialog.:

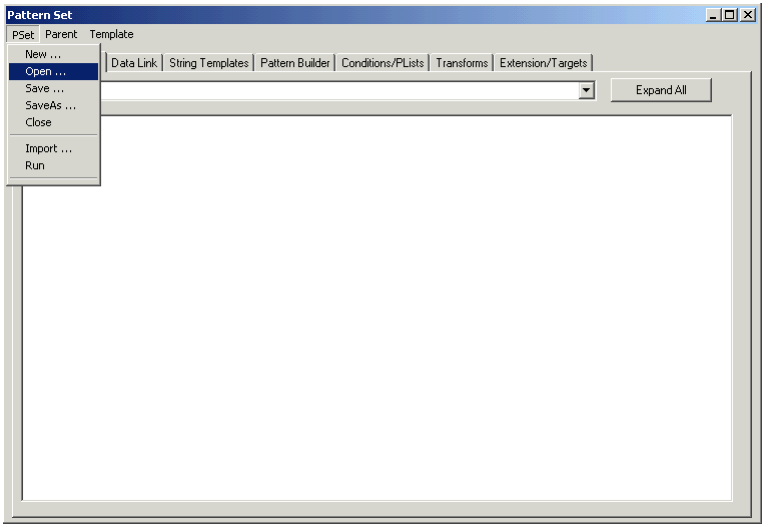


Figure 7: Main PatternStream Dialog.

2 Select Open from the PSet drop-down menu. This brings up the Open File dialog.

3 Set the dialog to point to the directory containing the directory examples. For a standard install, this would be in Program Files\Finite Matters\PatternStream 2.0\Examples\Directories\PsetFiles.

Note: For this first time, it is recommended that you select the file GoochlandPhoneBook.pst.

Running a PSet File

Once you have opened a pset file, then the next basic operation you might want to perform is to run it. Select the PSet menu from the PatternStream main dialog and choose Run from the drop-down menu. The current pattern set will then generate a short three page phone directory.

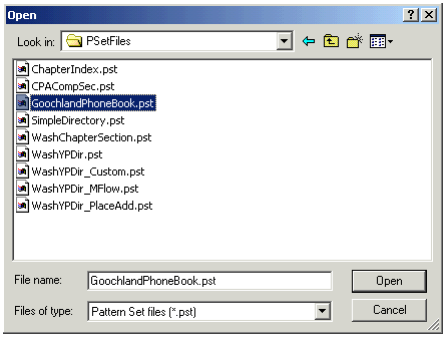


Figure 8: Open File Dialog.

The PS Control Bar

Assuming that the above example worked correctly, you should have noticed a few things. The first thing was that you could see the document being composed line by line and that it seemed to take a long time (fortunately is only three pages). The second, more important point is that once you selected Run, you lost control of FrameMaker/PatternStream and had to wait until the pattern set finished before regaining control. In order to regain control without having to wait for the pattern set to finish (or using Ctrl-Alt-Del in the usual Windows way), there is a separate tool called the PatternStream Control Panel which can be used to stop a pattern set once it has been started.

Besides displaying the PatternStream main dialog, the Launch PatternStream command also starts this PS Control Bar, if it has not already been started. If the control bar is not started when you initially launch PatternStream, you can start it manually by doing the following:

1. Select the PatternStream menu from the FrameMaker menu bar.
2. Select the Application submenu, then select PS Control.

3 This launches the PS Control Panel shown in Figure 9.

Figure 9: PatternStream Control Panel.

The main purpose of the control bar is to stop a pattern set file after you have started it running. Following the steps for running a pset file again, but this time after waiting only a few seconds, click on the Stop button on the PS Control Panel. Notice how the current pattern set stops running and you can work in FrameMaker/PatternStream again.

The Development Cycle

The previous little example illustrates one part a cycle that will be repeated over and over again when building a pattern set. One of the main advantages of have PatternStream run as a direct extension of FrameMaker is that you can immediately see the results of any recent changes by just selecting run, waiting a few seconds, then clicking on the Stop button in the PatternStream Control Panel. Data driven composition is an inherently iterative process. It is virtually impossible to workout all of the data extraction logic (i.e., SQL queries), the layout logic and the formatting characteristics completely in advance, except for the simplest of cases. Therefore the best approach

is to make a few changes, run the pattern set to see how those changes look, make any corrections that are necessary, then start the cycle again.

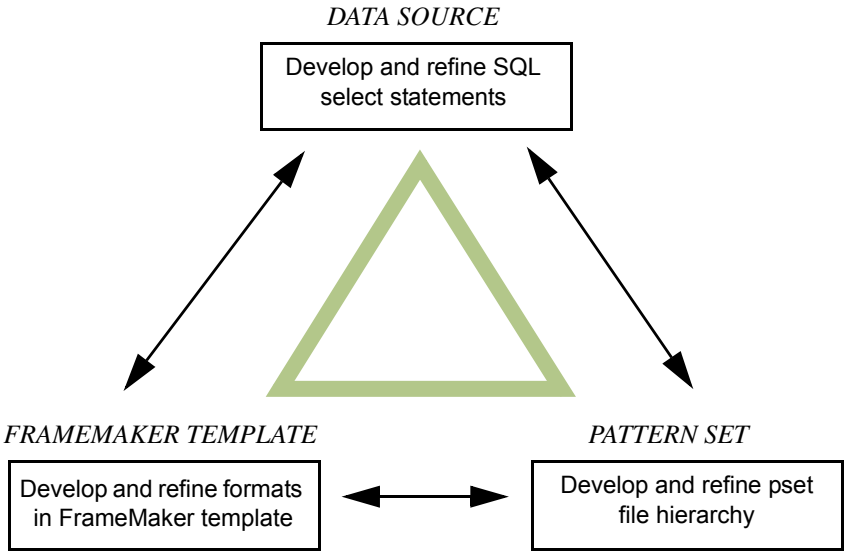


Figure 10: Cycle of pattern set construction

Figure 10, shows this development cycle in more detail. Each corner of the triangle represents a major component of a typical PatternStream project.

The Data Source

The data source component is everything needed to retrieve the data. This entails working out the different SQL select statements needed to implement the data hierarchy. As explained in the introductions (“The Data Hierarchy” on page 16), this will entail developing a series of nested SQL select statements that will reflect the logical structure of the data. These queries will be incorporated as an integral part of each pattern set and the details for working with PatternStream query objects are explained in subsequent chapters (Chapter 6, “Constructing queries”). However, its important to understand that most of the logic, and hence most of the work, involved in developing the data hierarchy is not really specific to PatternStream. Essentially, it s a function of the database

implementation itself, the logic structure of the data represented in the basic layout, and any intermediate layers that simplify viewing the data in the specific ways necessary to generate the desired output. In many cases the database itself is developed and refined as the pattern set is built. The data source phase typically includes the following tasks:

- setup of data source and data connections, sometimes requiring network administration.
- analyzing the database to learn where the data is located (i.e., what tables and what columns) and how extract different items of information.
- working the select statements themselves.

The FrameMaker Template

On the other side, we have the FrameMaker template. PatternStream itself is not concerned about the details of how to format the output. It uses the formats that are defined in a FrameMaker template document and is only concerned with the names of these various styles and tags, and not with how they are actually defined. This template document then serves as the repository of all these formatting details. All pattern sets (with a few exceptions) work with at least one template document. This document and the format definitions it contains is built in conjunction with the other two components of the development cycle as part of this iteration process. The following are some points to keep in mind, especially for experiences FrameMaker users:

- In developing the template document, you will tend to use far more styles and tags than you normally would. This is because PatternStream does not generally use format overrides hence every variation usually needs its own named format.
- Because clean logic is important in getting the pattern set to generate a coherent document, you will use techniques in the output document generated by a pattern set that would be impractical if the document were constructed manually.
- This template document usually has NO body pages.

Table 2 lists the FrameMaker format objects you will need to be familiar with.

Table 2: Format objects defined in the template document.

Named Format	Description
table format	Contains attributes such as the number of columns, the column widths, the default cell shading and fill, the default cell rulings, the position of the title and the number of heading and footing row.
paragraph format	Contain paragraph style information such as font size, style, margins and indentations,
character tags	These also specify font and font size, but only for a text fragment.
master page	Background page, referred to by name that. Every body page in a FrameMaker document must be assigned a background (or master) page.
cross-reference	These objects serve as prototypes for how a cross-reference will generate text.
marker and marker types	You will need to how specific built in marker types, such as index and cross-reference are used in FrameMaker.
variable formats	You will need to know how and when to use FrameMaker variable formats.
colors	You can define colors in a FrameMaker document and refer to them by name.
defined rulings	These objects define the style for the borders of table cells.

The Pattern Set

The pattern set sits between the data source(s) and the FrameMaker output document(s). All the various PatternStream objects which make up or are used in the pset hierarchy cooperate to compose the output document whenever the pattern set is run. The pattern set contains:]

- knowledge for how to connect to the data source(s).
- all the SQL select statements needed to retrieve the data.
- the names of all the styles and tags used to generate the structural elements (paragraphs, table, etc.) that will makeup the output document.
- instructions for inserting text in these structural elements.
- the logic that pulls all these pieces together.

PatternStream is the application you will use to build each pattern set.

PatternStream Menu

Along with launching the PatternStream application, the PatternStream main menu provides access most application administrative functions, documentation and data conversion utilities. This main menu, as shown in Figure 6, is part of the FrameMaker main menu bar

Table 3: PatternStream main menu.

Menu Item	Description
Application	
Launch PatternStream	Launches the PatternStream application and starts the PatternStream control panel.
PS Control	Independently starts the PatternStream control panel.
Run Batch PSet	Runs the pset file set for batch execution.
Documentation	
About PatternStream	
Getting Started	Brings up the PatternStream Getting Started document.
User Manual	Brings up a PDF version of the PatternStream user manual.

Table 3: PatternStream main menu.

Menu Item	Description
Registration	
Premier	The premier license of PatternStream allows complete access to create and modify pset files.
Get Serial Number	Displays the PatternStream serial number.
Enter Key	Dialog used to input the activation key for the current license.
Runtime	The runtime license of PatternStream allow a user to run any pset file but not to modify it.
Get Serial Number	Displays the PatternStream serial number.
Enter Key	Dialog used to input the activation key for the current license.
Run PSet File	Allows user to choose a pset file and then run it.
Convert	
Version 0xx to 0yy	

Main PatternStream Dialog

When you launch the PatternStream application, the **Pattern Set Dialog** (Figure 7) is automatically displayed. This is a non-modal, tabbed dialog. Table 4 lists the tabs and the object classes each tab is concerned with.

Table 4: PatternStream main dialog tabs

Tab Name	Description
Pattern Set View	This tab displays the pattern set hierarchy that you construct. From this tab, you can manipulate the attributes of any object in the current pattern set. You can also switch from one pset file to another.

Table 4: PatternStream main dialog tabs

Tab Name	Description
Data Link	This tab provides access to the objects that handle data retrieved from external data sources and provide storage for those values. You can manipulate the following objects classes from this tab: • Variables • Query Objects • Connections
String Templates	Use this tab to create and manipulate string templates, which tell the PatternStream application how to build text from the values derived from external sources.
Pattern Builder	This tab is where you create the patterns, or logical structure, of the information you want to publish. Use this tab to create and manipulate patterns and targets.
Conditions/PLists	This tab provides access to conditions and parameter list objects.
Transforms	Functionality provided on this tab allows you to transform data into a more desirable or useful form. For example, use this tab to transform ANSI characters with numeric values of 126 or higher to their equivalent FrameMaker characters, so that they will appear correctly in the FrameMaker document created by the PatternStream application.
Extensions/Targets	Functionality on this tab lets you set up specialized program extensions, such as a log file. You can also use this tab to more easily edit and manipulate target objects without having to know their exact position in the pset hierarchy.

Besides these dialog tabs, there are also three drop-down menus accessible from the main dialog.

PSet Menu

The PSet menu is analogous to the File menu in other Windows application. Here you perform the usually file operations such opening and closing a pset file.

Table 5: PSet Menu

Menu Item	Description
New...	Create a new pset file.
Open...	Open an existing pset file
Save	Save the current pset file.
SaveAS	Save the current pset file to another file.
Close	Close the current pset file.
Import...	Import an object from another pset file that is also open in the current session.
Run	Run the current pset file.

Parent Menu

This menu is concerned functions that deal with aggregate objects. It is only enabled when a tab associated with an aggregate object is active.

Template Menu

PatternStream itself is not concerned about the details of how to format the output. It uses the formats, as defined in a FrameMaker template document, and is only concerned with the names of these formats, and not how the actual format is defined. For example, a paragraph format call “Firm Name”, can defined in the template file to have various attributes typical of a paragraph, such as:

- left and right margins
- first line and subsequent line indentations.
- font, font size and style.
- line spacing and alignment.

The PatternStream application is only concerned with the name of this paragraph format and not how it is defined.

This menu provides functions that assist manipulation of FrameMaker template documents used in a pattern set.

Table 6: Template Menu

Menu Item	Description
Open Template	Opens the template document for the current pattern set.
Extract Formats	Load all of the format names defined in the template document into the current pattern set.
Update Template	Allow you to modify the format definitions in the FrameMaker output document and load them back into the template document.
Template Report	Create a report defining how each format is being used in the current pattern set.

Pattern Set View Tab

The Pattern Set View tab allows you to deal the pattern set as whole, providing a view of the complete pset hierarchy. Additionally, you can access any object from this tab by selecting the object and using the object popup menu.

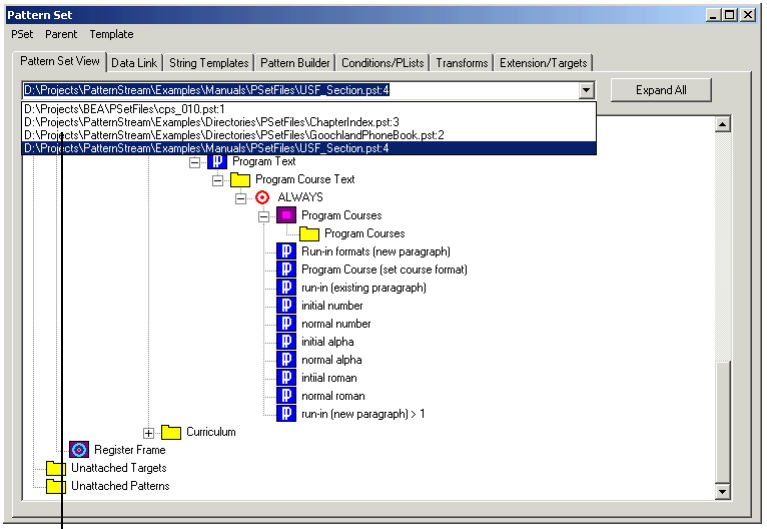


Figure 11: The Pattern Set View tab

Importing and Exporting

You can copy PatternStream objects from one pattern set to another. This process is referred to as exporting. When you export an object, PatternStream recursively places the object and all object it references into its internal paste buffer. To export an object follow these steps:

1. Find the object in the pattern view window and select it. You may have to expand sub-trees as you go.
2. Click the right mouse button and select Export from the object popup menu.

Once an object is in this paste buffer you can import it into another pattern set. To import an object that is in the PatternStream paste buffer:

a Select the pattern set from the pattern set drop-down list on the PatternView tab of the PatternStream main dialog. This will make it the current pattern set.

Select the PSet menu and choose Import. This brings up the import dialog.

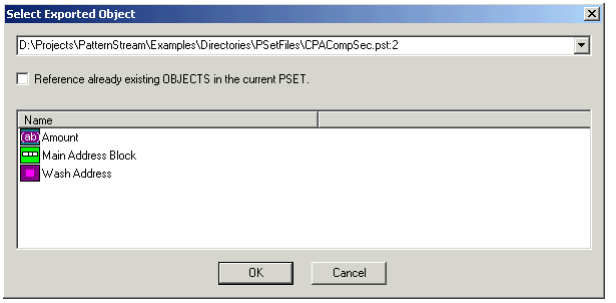


Figure 12: The PatternStream import dialog

b Using the drop-down list at the top of the dialog, select the pattern set that contains the object you wish to import. The objects that are in the paste buffer that belong to this pattern set are displayed in the object list.

c Only the actual objects exported from this pattern set are shown and not the objects that are referenced by them.

d Check this option if you wish PatternStream to reference objects with the same name in the target pattern set. If unchecked, PatternStream will create a new object with a different name.

e Click on the **OK** button. The current pattern set will now have a copy of the object selected and all of its associated objects.

Simple Object Window

Simple objects are accessed through a list box control that displays the object's icon and name, listed. Each object class has its own window. To access an

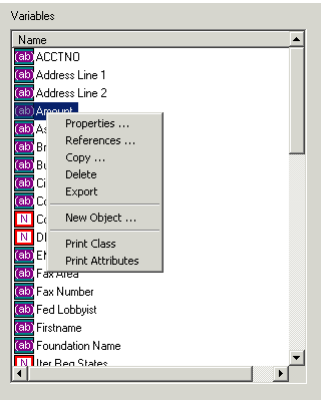


Figure 13: Simple Object Window.

object's attributes, select its icon, then click the right mouse button. Select the desired function from the object popup menu. describes the choices in this menu

Table 7: Object Popup Menu

Menu Item	Description
Properties...	Displays the selected object's properties dialog, allowing you to modify its attributes.
References...	Selecting this options displays a list of other objects in the current pset that reference the selected object.
Copy...	Used to create a copy an object.
Delete	Deletes the selected object. *Note:* An object cannot be deleted as long as another object is referencing it.
Export	Places the selected object in the export buffer. From here, you can paste a copy of this object into another pset file.
New Object...	Brings up the Create Object dialog.

Table 7: Object Popup Menu

Menu Item	Description
Print Object	Generates a description of the selected object in a FrameMaker file that can be printed or converted to PDF.
Print Class	Generates a description of each object in the current windows's class in a FrameMaker file that can be printed or converted to PDF.

Aggregate Object Tab

An aggregate object is one whose primary nature is to serve as a container for a collection of sub-objects. Most of the time, these sub-objects are subsidiary to the parent container and have no independent existence (the exception being pattern objects). There are three classes of aggregate objects in PatternStream:

- String Templates
  These objects are responsible for generating text content. They are composed of an indefinite number of sub-objects called segments of various types, each type performing a specific function.
- Transforms
  These objects are take the stored value of a PatternStream variable and change it to something else. They are composed of an indefinite number of sub-objects called directives. Each type of directory performs a specific operation such as string concatenation, addition, etc.
- Patterns
  These objects, along with the sub-targets they contain, are the most important objects in PatternStream. Unlike the other aggregate objects, the sub-targets of a pattern have an independent existence and can be moved from one pattern to another.

All different in detail, all aggregate objects tabs work the same way. Figure 14 shows an example of an aggregate object tab, in this case, the string templates tab of the PatternStream main dialog. On the left hand side of the dialog, there is a list of all the objects in the current pset belonging to the specific class, listed in alphabetical order. Selecting any object makes it the current object. On the right side of the dialog is the sub-object work area. Here is where you can add, remove and edit the sub-objects that will make up the current parent object. You

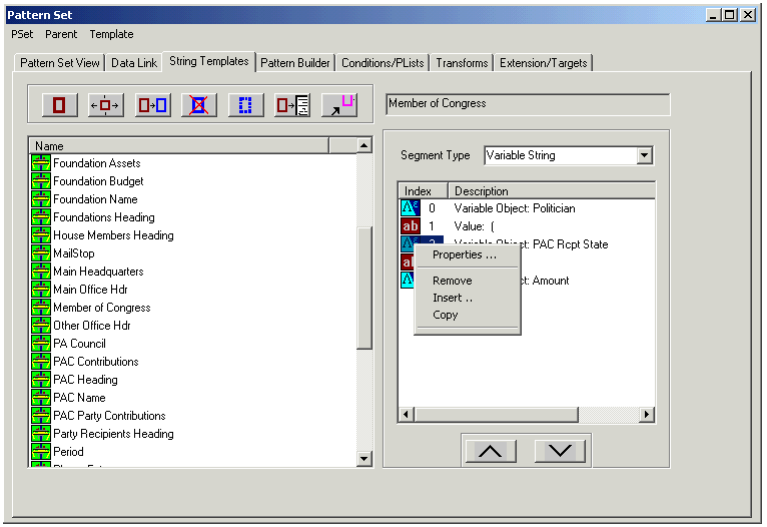


Figure 14: Aggregate Object Tab.

manipulate the sub-objects by first selecting one, then clicking on the right mouse button. Notice how when you select a different object (i.e., a string template in this case) the list of sub-objects changes to those of the newly selected parent object.

Aggregate objects are manipulated through the parent menu, which is enabled

Table 8: Parent Menu and Parent Controls

Control	Menu Item	Description
	**Object Info...**	Brings up the properties dialog for the current aggregate object.
	**References...**	Selecting this options displays a list of other objects in the current pset that reference the current object.
	**Copy...**	Used to create a copy of the current object. This copies not only the parent object properties but also all of the sub-objects and their properties.
	**Delete**	Deletes the current object. *Note:* An object cannot be deleted as long as another object is referencing it.
	**New Object...**	Brings up the Create Object dialog.
	**Print Object**	Generates a description of the current object in a FrameMaker file that can be printed or converted to PDF.
	**Export**	Places the selected object in the export buffer. From here, you can paste a copy of this object into another pset file.

only when an aggregate object tab is active. These function are also accessible using the tool bar in the upper left of the dialog. Table 8 describes the functions accessible through the parent menu and their corresponding button control.

Component Edit Dialog

A type of dialog that occurs frequently is the component edit dialog. Here, the object being modified has, among other attributes, a list of component items. In all cases, the components appear in a list box. Work can only occur on the currently selected component. Components are selected by clicking on the index number listed in the left most column of the component display. When a component is selected, its attributes are displayed the work area below the list, where you can modify them. Clicking the Modify button will replace the current components attributes with the values displayed in the work area. Clicking on the Delete button will, of course, delete the component.

Note: It is important not to forget to click on the Modify button. Even though the attributes displayed in the work area for the current component may have been modified, these changes will NOT go into effect until the Modify button is clicked.

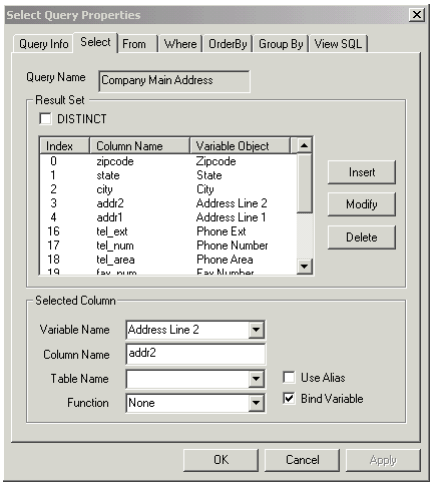


Figure 15: Component Edit Dialog

Error Messages and the FrameMaker Console

Whenever an error occurs, either during development or the running of a pattern set, a generic error message is displayed, such as in Figure 16. In all cases, there will be a more detailed message in the FrameMaker console.

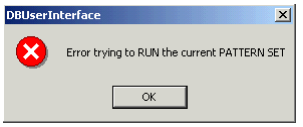


Figure 16: Generic PatternStream Error Message

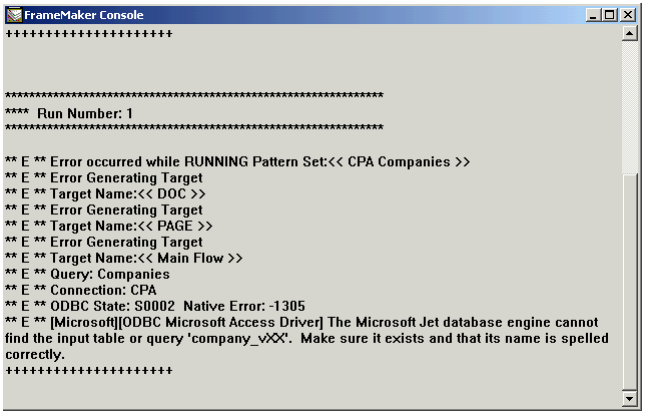


Figure 17: Detailed Error Message Appearing in the FrameMaker Console.

Chapter 3: Creating a new pattern set

A pattern set is an object that contains all of the instructions necessary to compose an output document from data residing in external sources, usually a relational database. These "instructions" are embodied in the hierarchical structure called the PSet hierarchy. To each pattern set there corresponds a file called a PSet file, which essentially is a pattern set saved to disk which can then be reloaded in a subsequent PatternStream session.

Preparation

Besides containing the PST hierarchy and all of the auxiliary objects needed to publish a document, the pattern set acts as an intermediate object that sits between data sources, imported files such as graphics and text insets, and output files, such as PDF, XML and FrameMaker book files. Because of this dependence on other files, it is important to organize a project before you even begin developing the pattern set itself.

Required Files

Three file types are required for every PatternStream project:

Template (.fm)	This is a normal FrameMaker file that will store all of the format definitions that the pattern set will use, such as paragraph styles, character tags, master pages, etc.
Data Source(.mdb, .DAT, etc.)	Every pattern set will need a data source of some kind, usually an ODBC Data Source (for large distributed DBMS systems such as Oracle or Sybase, thinking of the data source as a "file" can be a stretch but the concept still holds).
PSet File (.pst)	This is the file that will contain all of the various objects that PatternStream will use to generate the output document(s). This chapter contains instructions on how to create this file.

Additional Files

n most PatternStream projects additional files are also required to create a complete output document.I

Images (.tif, .bmp, .eps, etc.)	If the output will contain graphic images, such as pictures or charts, then these must exist somewhere (i.e., accessible from the file system). Usually, the database will contain pointers to these images (i.e., the filenames).
Insets (.fm, .doc, etc.)	Sometimes, the document generation process requires the inclusion of external text files such as pre-composed FrameMaker and Microsoft Word documents.

Creation of the Pattern Set

To create a new pattern set, follow these steps:

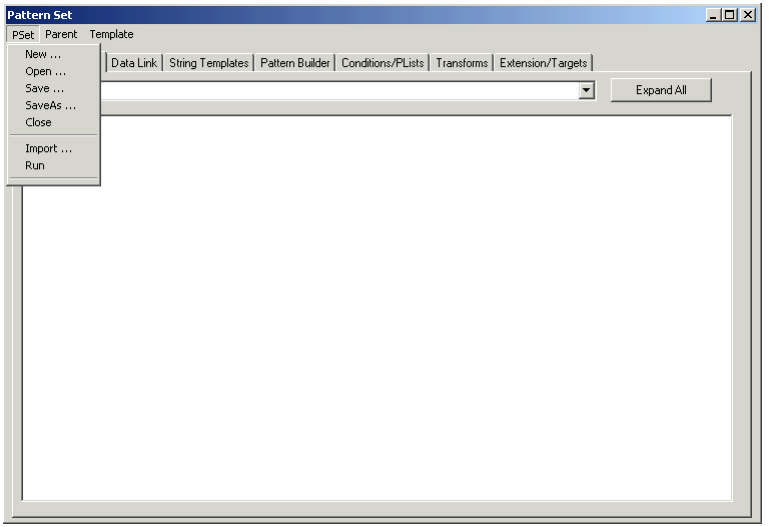


Figure 18: Select New from the PSet Menu

1 From the PSet menu on the PatternStream main dialog, select New. The New Pattern Set dialog box (Figure 19) is displayed.

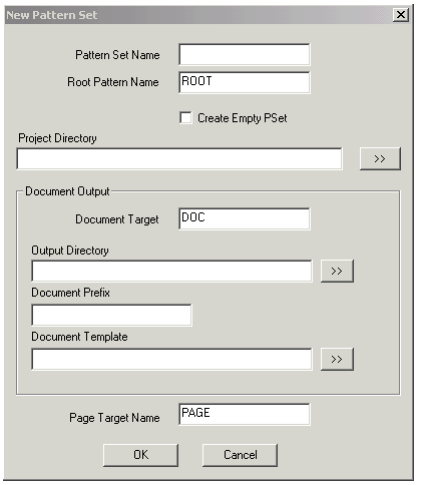


Figure 19: New Pattern Set dialog box

2 Fill in the fields in the dialog box and select **OK.** The fields in the New Pattern Set dialog box are as follows:

Pattern Set Name	The name of the pattern set you are creating.
Root Pattern Name	Every pattern set must begin with an initial pattern that starts the composition process. By default, the PatternStream application names the top-level pattern in the hierarchy ROOT. However, you may provide a different name.
Project Directory	All directory specifications in the new pattern set will relative to this absolute path name.
Document Target	This target will generate the output FrameMaker document. By default, this target associated with the ROOT pattern is named DOC but you may provide a different name.
Output Directory	The name of the directory in which you want your FrameMaker output document to be stored . You can type the path name of the directory or select the >> button, navigate through the Select Directory dialog box, then select the directory you want.

Document Prefix	The prefix you specify here is used for the output document.
Document Template	Specify the location of the FrameMaker template that you want to use to create your PatternStream-generated FrameMaker output document. You can type the full path name and file name of the template or select the >> button, select the template you want from the Open dialog box, then select the **Open** button.
Page Target Name	By default, the PatternStream application creates a PAGE pattern associated with the DOC target. It associates a PAGE target with the PAGE pattern. You can rename this PAGE target by typing a new name in this field.

3 The **Pattern Set View** tab is displayed showing the basic hierarchy of the pattern set you defined. It should be similar to the hierarchy shown in Figure 20.

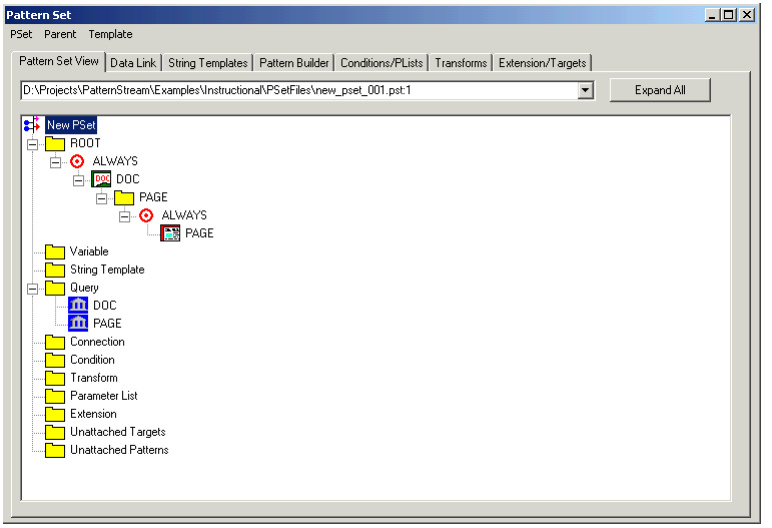


Figure 20: Basic hierarchy of newly defined Example pattern set

4 Select the **Save** from the PSet menu to save your newly defined pattern set.

Components of a newly created pattern set

Figure 20 illustrates the basic hierarchy of the minimal pattern set that is created. Starting at the top of the hierarchy, Table 9 describes the objects generated by the PatternStream application in a new pattern set.

Table 9: Components of a newly created pattern set in hierarchical order

Icon	Description
New PSet 	This icon represents the entire pattern set and gives you access the global pattern set attributes.
ROOT 	The top-level pattern in the hierarchy. ROOT is the default name for the top-level pattern. When the pattern set is run, execution of the patterns in the PSet Hierarchy starts here.
ALWAYS 	The red bull's-eye symbol indicates a named target list, which is the method by which target objects are inserted into the PSet Hierarchy. Target lists determine when and where a target is generated in the pattern set execution cycle. Targets in an ALWAYS target list of a pattern are generated every time the pattern's assigned query returns a result.
DOC 	The DOC Document target is part of the ROOT pattern's ALWAYS target list. This target generates a FrameMaker document using the template you specify when you create the pattern set. DOC is the default name provided for the DOCUMENT target. You can specify a different name when you create a pattern set.
PAGE 	The PAGE pattern is attached to the DOC Document target (as its subpattern). Like the ROOT pattern, this pattern will execute only once based on the attributes of its attached static query. PAGE is the default name provided for this subpattern. You can specify a different name by specifying a different PAGE target name when you create a pattern set.
ALWAYS 	Indicates that the PAGE Page target belongs to the PAGE pattern's ALWAYS target list.
PAGE 	The PAGE Page target is part of the PAGE pattern's ALWAYS target list. This target generates a series of connected body pages using the master page format of the FrameMaker template you specify.

Table 9: Components of a newly created pattern set in hierarchical order (continued)

Icon	Description
Variable 	Variables are placeholders for the data extracted by the queries you construct. For each value queried from the database, a corresponding variable must be created. Variables are stored in this folder. For details about variables, see Chapter 5, “Variables.”
String Template 	String templates create content (such as text, graphics, and markers) in your document and are attached to content targets (see “Kinds of targets” on page 128 for more information). String templates are stored in this folder. For details about string templates, see Chapter 15, “String templates.”
Query 	Queries are sets of instructions that, in most cases, retrieve information from a database. A query must be attached to every pattern. When you create a new pattern set, the PatternStream application automatically creates two STATIC queries (DOC and PAGE) to correspond with the two targets by the same name. For details about queries, see Chapter 6, “Constructing queries.”
DOC 	The default static query that is attached to the ROOT pattern. It forces the ROOT pattern to cycle only once.
PAGE 	The default static query that is attached to the PAGE pattern. It forces the PAGE pattern to cycle only once.
Connection 	For query objects that actually retrieve data from an external data source (usually an ODBC Data Source), the must a connection object that establishes communication with the DBMS where the data resides.
Condition 	Conditions are objects you attach to string template segments or targets that determine whether the object will generate output on whether they evaluate to TRUE or FASLE. Conditions are stored in this folder. For details about conditions, see Chapter 16, “Creating conditions.”

Table 9: Components of a newly created pattern set in hierarchical order (continued)

Icon	Description
Transform 	A transform is a set of directives that changes, or transforms, data extracted from a database. Transforms are stored in this folder. For more information about transforms, see Chapter 17, “Transforms.”
Parameter List 	Parameter lists are specialized lists of variables.
Extension 	Extensions are a class of miscellaneous PatternStream objects such as Lookup tables (arrays), PSetTrees, and various PatternStream plug-ins.
Unattached Targets 	This folder holds any targets that you detach from a pattern or another target.
Unattached Patterns 	This folder holds any patterns that you detach from a target.

Modifying properties of a pattern set

To modify the properties of a pattern set, follow these steps:

1. In the **Pattern Set View** tab (Figure 20), select the pattern set icon at the top of the pset hierarchy.

2. Click the right mouse button and select properties from the popup menu, or double click on the pattern set icon.

3 The Pattern Set Properties dialog is displayed (Figure 21).

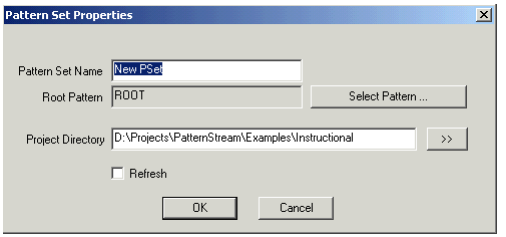


Figure 21: Pattern Set Properties dialog box

4 Modify the fields in the box as needed, then select **OK.** The fields are as follows:

Pattern Set Name	Type the new name for your pattern set here.
Select Pattern	Click on this button to change the root pattern.
Project Directory	Type in the name of a absolute directory. All directories specified in the pattern set will be resolved relative to the directory specified here.
Refresh	Check this box if you wish PatternStream to re-display the output document as it retrieves data.

Note: If the **Refresh** check box is selected, when the pattern set is run, the FrameMaker display updates every time a change is made. When **Refresh** is selected, you can view the output file generating. If **Refresh** is not selected, the output file generates faster.

Chapter 4: Connecting to an External Data Source

The PatternStream application connects to an external data source to extract the information you need for your document. You use PatternStream connect objects to establish and manage these transactions. A pattern set can have as many connect objects as is needed. There can be multiple connection objects pointing to the same data source or connects objects that point to different data sources. This means that you can extract and combine data from more than one database.

Types of connections

Table 10 describes briefly each type of PatternStream connection object.

Table 10: Types of PatternStream connections

Target name	Description	For details, see...
ODBC 	Used to connect to ODBC compliant databases to obtain database information via SQL.	"ODBC Connection" on page 51
Flat File 	Used to obtain information from flat files like HTML.	"Flat File Connection" on page 53
Custom 	Used to make custom connections to information sources via a DLL.	"Custom Connection" on page 55

Working with Connection Objects

Creating a database connection

To make a data source available to a pattern set, you create a database connection. To do so, follow these steps:

1 In the PatternStream application, select the **Data Link** tab. The bottom left section of the tab contains the Connections area (Figure 22).

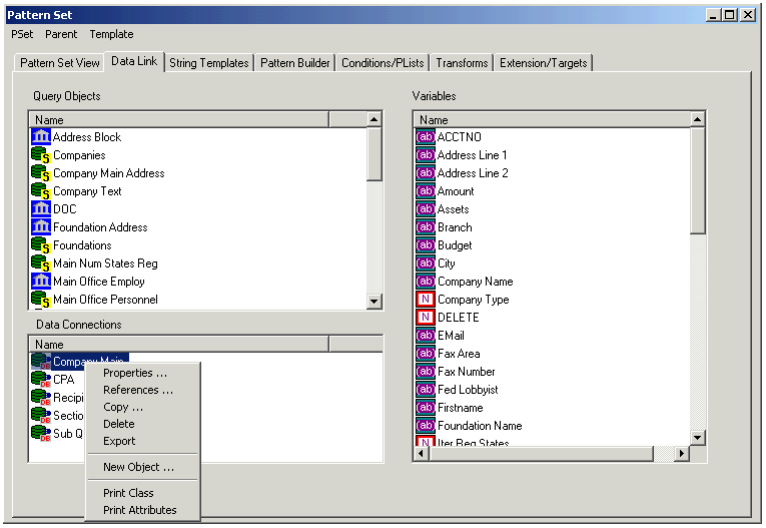


Figure 22: Connections area, Data Link tab

2 In the Connections area of the dialog box, click the right mouse button and select new object. The Create New Connection Object dialog box is displayed (Figure 23).

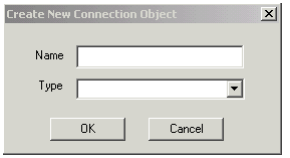


Figure 23: Create New Connection dialog box

3 In the Name field, type the database's ODBC data source name or a name descriptive of the specific pattern or project.

From the Type drop-down list, select the database type (see Table 11), then select **OK.**

ODBC Connection

Open the Create New Connection dialog, select **ODBC** as your database type and select **OK**, the DB Connection dialog box is displayed (Figure 24).

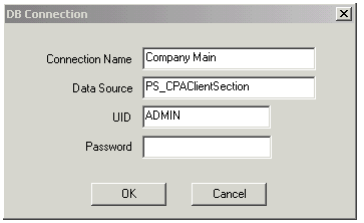


Figure 24: DB Connection dialog box

To define the properties of this connection, follow these steps:

Table 11: File Connection Parameters

Parameter	Description
Name	The name of the connection object. *Note:* The name must be unique within the class of connection objects.
ODBC Data Source Definition	
Data Source	The name of the data source as defined in the ODBC Data Source tab of the Windows Control Panel.
UID	The user id you wish to use to log into the database with.
Password	The password the given user id.

Defining an ODBC data source

Your PC's operating system recognizes a database through defined ODBC data sources. Before the PatternStream application can access your database using the

ODBC programmer's interface, you must define an ODBC data source for that database.

To define an ODBC data source on your workstation or server, follow these steps:

1 Select the Windows **Start** button, then **Settings**, and then **Control Panel.**

2 Double-click the ODBC icon 

. The ODBC Data Source Administrator dialog box is displayed. Select the **System DSN** tab (Figure 25).

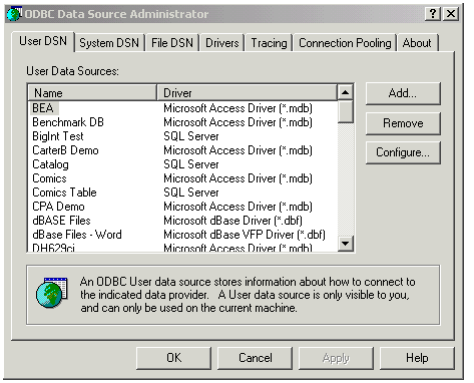


Figure 25: User DSN tab, ODBC Data Source Administrator dialog box

3 Select the **Add** button. The Create New Data Source dialog box is displayed (Figure 26).

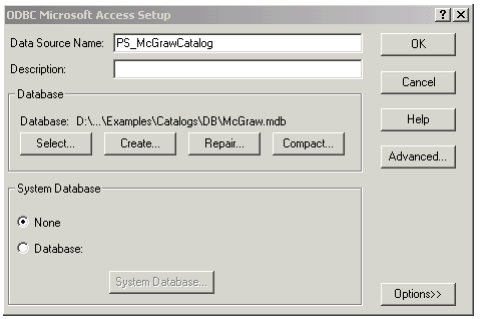


Figure 26: ODBC Setup dialog for MS Access Databases

4 From the list of database drivers installed on your PC, select the driver for the database you plan to use, then select the **Finish** button.

Note: If you are uncertain which driver to select, consult your database administrator.

5 In the ODBC Setup dialog box that is displayed for the driver you select, provide the necessary information for the driver, consulting your database administrator as necessary.

Note: In the Data Source Name field of this dialog box, type the name of the database you plan to use or a name descriptive of your project. The PatternStream application uses this name to query the database.

6 After you have provided all the information, select **OK**, and close the Control Panel window.

Now that your operating system can recognize the database, you must now make the data source available to your pattern set by creating a database connection.

Flat File Connection

Bring up the Create New Connection dialog, select **Flat Ascii File** as your database type and select **OK**, the File Connection dialog box is displayed (Figure 27).

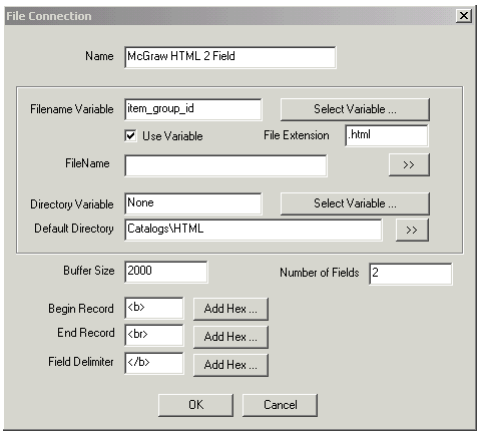


Figure 27: File Connection dialog box

Table 12 describes each parameter that determines how the connection object will interpret a given ASCII text file (or set of files):

Table 12: File Connection Parameters

Parameter	Description
Name	The name of the connection object. *Note:* The name must be unique within the class of connection objects.
File Definition	
Filename	The name of the file that contains the text data.
Filename Variable	An alternate way of specifying the filename of the source data that allows the connection object to point to a different ASCII file each time it opens a file for reading. Here, insert the name of the PatternStream variable that will contain the filename where the data resides. This value can change during the running of a pattern set.
Use Variable	Flag indicating whether a variable will be used to resolve the filename.
File Extension	The file extension of the ASCII text file (.txt, .log, .HTML, etc.)
Default Directory	The directory the contains the ASCII text file(s) that will be used as the data source.
Directory Variable	An alternate way of specifying the directory of the source data that allows the connection object to point to a different directory each time it opens a file for reading. Here, insert the name of the PatternStream variable that will contain the directory name where the current data resides. This value can change during the running of a pattern set.
Buffer Size	The maximum expected size of the records in the source data file.

Table 12: File Connection Parameters (continued)

Parameter	Description
Number of Fields	The number of fields in each record.
Begin Record End Record	The characters that determine where the data starts and ends in each record (for extracting only a sub-record rather than the entire record).
Delimiter	The character that delimits each record. Typically these files are tab or comma delimited.

Custom Connection

Open the Create New Connection dialog, select **Custom** as your database type and select **OK**, the File Connection dialog box is displayed (Figure 28).

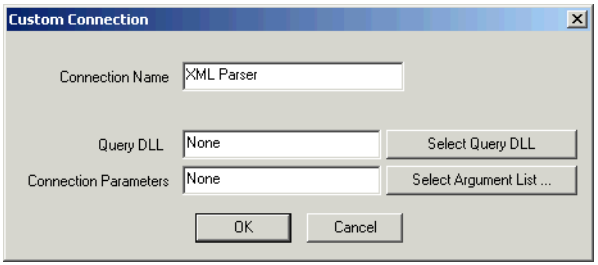


Figure 28: File Connection dialog box

Table 13 describes each parameter that determines how the connection object will interpret a given ASCII text file (or set of files):

Table 13: Custom Connection Parameters

Parameter	Description
Name	The name of the connection object. *Note:* The name must be unique within the class of connection objects.

Table 13: Custom Connection Parameters (continued)

Parameter	Description
Query DLL	Specifies the name of the DLL file to be used to obtain the information.
Connection Parameters	Specifies the name of the argument list to used for the custom connection.

Chapter 5: Variables

In the PatternStream application, *variables* are used for a wide variety of purposes, such as to store data extracted from a database, to act as iteration counters and to determine the true or false value of *conditions.* These variables are similar to the variables and identifiers used in other programming environments, such as C++ and Visual Basic.

Variable basics

The primary purpose of PatternStream variables is to provide easy access to data values extracted from some external data source. When you run a pattern set, a series of nested queries are executed to access and retrieve the data. A variable is associated with, or *bound* to, each database column you are accessing. The Database Management System (DBMS) writes the data to these variables, which in turn make the data available to the rest of the PatternStream system.

All PatternStream variables share certain characteristics.

- Variables have global scope in a pattern set. This means that any variable and its contents are accessible to all the other objects in the pattern set.
- Variables actually contain two values—the raw value and the current value. It is the raw value that gets bound to the external data source. Whenever another object needs to access the value of a variable, the raw value is first copied to the current value. Virtually all PatternStream objects use the current value of the variable object, not the raw value.
- You can't modify a variable's type, for example, change a String variable to a Integer variable. If you need to change a variable, you must first create a new variable. Then, wherever the old variable is used, replace it with the new one.
- Some applications where variables are required restrict the variable type that can be used. For example, only integer variables can be used as iteration counters.
- All variables, regardless of type, can be used to generate text output.
- If you need to modify the current value of a variable, you can do so by attaching a *transform* to the variable. For example, transforms let you change the characters of a string values (such as making letters uppercase) or perform mathematical calculations on numeric values.

There are three aspects to the stored value that must be kept in mind when using a PatternStream variable.

Raw Value

The actual value retrieved from the database. This value is not changed by any operations performed on the variable. This can be a single value, such as an integer or floating point, a string value or a set of numbers such as the triplet <year, month, day>

Current Value

The value contained in the variable after any operations (i.e., transforms) have been performed on the raw value.

Formatted String

The string representation of the current value. This aspect is controlled by the use of assigned format objects.

It is important to be aware, in any given application, which aspect is being used. Table 14 lists some examples of variable applications and the aspect of the variable value that is required.I

Table 14: Variable applications and their required aspect.

Application	Aspect required
Generate text content	Formatted string
Source variable of an Assignment target	Current or raw value
Condition evaluation	Current value
Pointer to a graphic file	Formatted string
Build a data-driven output filename	Formatted string
Argument for a numeric transform	Current value
Bound parameter value in an SQL select statement's where clause	Raw value
Text contained in a marker	Current value
Argument passed to a FrameScript script	Current value

Types of variables

PatternStream variables fall into four main groups.

Numeric

Variable types in this group store a number such as an integer or floating point.

String

These variable types store character data such as char(n), varchar, text blobs, etc.

Date

These variables store data and time information as set of numbers.

Special

This group contains variable types that don't fall in any of the above three classes.

Table 15 describes briefly each type of PatternStream variable.

Table 15: Types of PatternStream variables

Variable Type	Description	Valid Format Type
String		
String 	The value is represented by a character string. Used mostly to store the data type char(n) or varchar(n).	CStirng Character/Position Default
Text 	Used for store arbitrarily large blocks of text, such as memo (MS Access), clob (Oracle) and text (SQL Server) data types.	NONE
Numeric		
Big Integer 	Used to store 8-byte integers. For example, for state and county budgets, dollar values greater than 1 billion (1,000,000,000) cannot be stored accurately in a normal 4-byte integer	CString Financial Default

Table 15: Types of PatternStream variables

Variable Type	Description	Valid Format Type
Double 	Used to store 8-byte floating point numbers. This variable type is useful for storing budget numbers for Federal government agencies and for dealing with distances to galactic nebulae.	CString Financial Decimal Default
Integer 	Used to store 4-byte integers. A typical application is to store ID numbers from entities residing in a relational database.	CString Financial Default
Float 	Used to store 4-byte floating point numbers. A floating point number is defined as a decimal number composed of a mantissa with a value between one and minus one, and an integer that represents a power of ten.	CString Financial Decimal Default
Date		
Date 	Used to store the triplet <year, month, day> from data retrieved from a database that uses the datetime data type.	Date Default
Time 	Used to store the triplet <hour, minute, second> from data retrieved from a database that uses the datetime data type.	Time Default
Timestamp 	Used to store the six-tuple <year, month, day, hour, minute, second> from data retrieved from a database that uses the datetime data type.	Date Time Default
Special		
Reference 	This type of variable gets its raw value from another variable. Use when the raw value of a variable needs to be transformed for multiple purposes.	Depends on referenced variable type

Variables and Transforms

The value retrieved from a database sometimes must be modified before it can be used. To perform these modifications all variables can have a transform attached to them (see Chapter 17, “Transforms” for more details). This transform operates on the raw value to produce the current value each time another PatternStream object needs to access the value of the variable. Any object retrieving the value stored in the variable must use the transformed current value rather than the raw value. When a transform is attached to a variable, each directive that makes up the transform will be applied sequentially to the current value, starting first with the raw value. Figure 29 is an example of how the value of a string variable that contains the name of an entity (i.e., a firm) is transformed into a graphic filename.

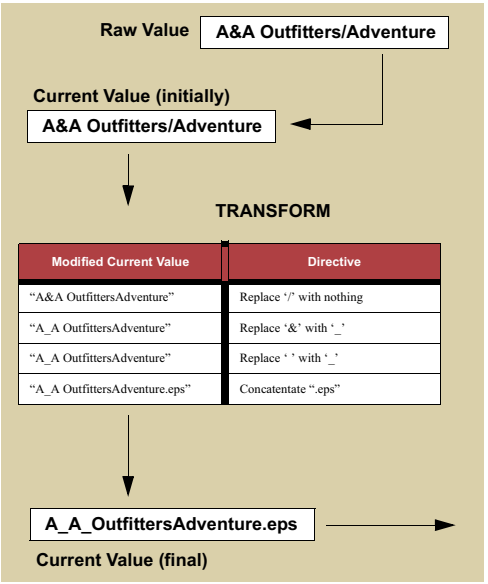


Figure 29: Attached transform modifies the current value.

Some other applications where you might attach a transform to a variable are:

- In many database applications, you can insert carriage returns in data strings. If the value is stored in the variable, these carriage returns must be removed before the strings can be formatted correctly. You remove the returns by attaching a transform object to the variable.
- Apply a transform that contains a modulo 2 directive to a variable that is acting as an iteration counter. This will force the current value of the variable to

alternately take on the values 0 and 1. You could then use this to alternate displaying a shaded and a non-shaded row in a table.

- To access values contained in a lookup table (see “Lookup target” on page 250).

Note: The attached transform object does not modify the raw value in any way; however, it is applied to the raw value to produce the current value whenever the variable is used.

Format objects

Each variable is assigned a format object to determine the string representation of the current value. Most format objects consist of just a control string, however, some contain multiple attribute settings. Figure 30 shows two examples of format settings for different variable types. The first illustrates a string variable using a character/position format type and format string that acts as a character template. The second example shows a data variable with a date format type assigned.

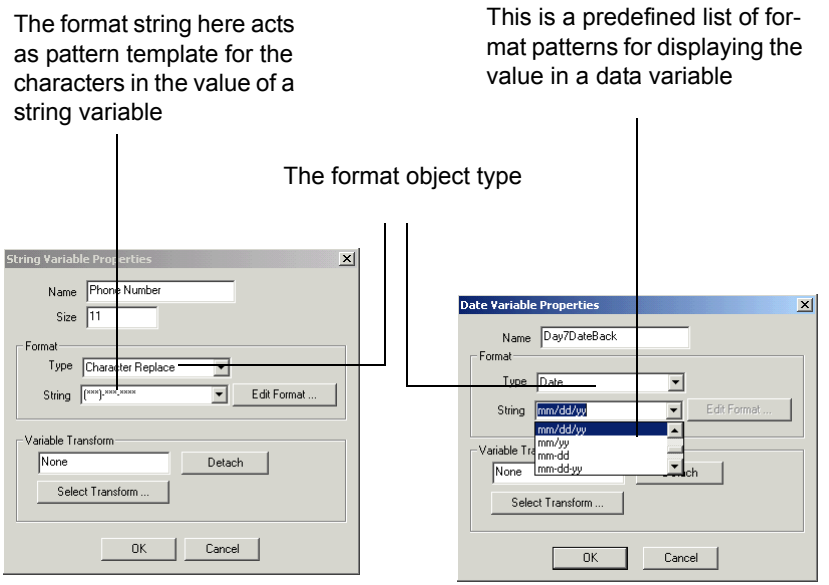


Figure 30: Format types and format strings

Notice how in this later case the choice of format string is restricted to the items in the drop-down list.

Default format

The Default format can be considered the Null format, because any variable assigned to this format generates no text output, regardless of its type or current value. There is no format string, nor are there any associated parameters.

CString format

The CString format is familiar to those who know the C programming language. As format strings with a specific syntax are used in the *printf* function call in a C program, format strings with the same syntax are used in the CString format type. Table 16 describes the CString format syntax for specific variables. For more information on the CString format, refer to the C compiler documentation or a good book on C programming.

Table 16: CString format syntax

Symbol	Variable type	Description
%s	String	Replace these two symbols with the current value of a String variable.
%d	Integer	Replace these two symbols with the current value of an Integer variable. The resulting display string has as many characters as there are digits in the number.
%nd	Integer	Replace these three symbols with the current value of an Integer variable. Limit the number of digits displayed to *n* and pad spaces to the left for numbers with fewer than *n* digits.

Table 16: CString format syntax (continued)

Symbol	Variable type	Description
%x.yf	Float Double	Replace this with the current value of a Float variable. Display the number using a total of x digits with y digits to the right of the decimal place. Uses trailing zeroes or rounding if necessary.

floating point value	format string	text
5.27892322	%7.2f	5.28
451.23897	%8.1f	541.2
871.000000	%5.0f	871
871.000000	%5.1f	871.0

%I64d	Big Integer	Replace this with the current value of a Big Integer variable (8 byte integers).

Financial format

The Financial format uses predefined format strings to display monetary data. Each format string works differently depending on whether the number is an

integer or a floating point. Financial format string options are displayed in Figure 31. Table 17 describes these options.

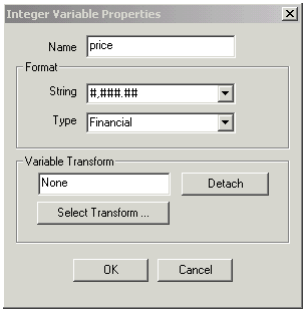


Figure 31: Financial format strings

Table 17: Financial format syntax

Symbol	Description
#,###.##	Displays dollars and cents, including commas. If the variable type is Integer, the value is interpreted as cents. Otherwise, the floating point number is displayed with only two decimal places.
(#,###.##)	Same as above, except negative numbers are displayed with parenthesis.
#,###	Displays dollars with commas. If the variable type is Integer, the value is interpreted as dollars. Otherwise, the floating point number is displayed without a decimal place. The value is rounded, not truncated.
(#,###)	Same as above, except negative numbers are displayed with parentheses.
(#,###.#)	Displays a single decimal place, including commas. If the variable is Integer, the value is interpreted as tenths rather than units. (This format string typically does not apply to integers).

Decimal format

The Decimal format object uses parameter settings rather than a format string to format a floating point value. You change these parameters in the Edit Decimal Format dialog box (Figure 32). Table 18 list the parameters appropriate for a Decimal format.

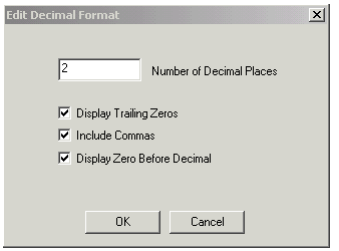


Figure 32: Edit Decimal Format dialog box

Table 18: Decimal format syntax

Parameter	Description
Number of Decimal Places	The number of digits to display to the right of the decimal point.
Display Trailing Zeros	Uncheck this option to avoid printing zeros at the end of the decimal point.
Include Commas	Check to insert commas to the left of the decimal point.
Display Zero Before Decimal	If the number is between 1.0 and -1.0, include a zero directly to the left of the decimal point.

Character/Position format

The Character/Position format object uses both a format string and a set of parameters and is valid only for String variables. This format serves as a template for replacing characters. For example, a database field might contain a phone number with no dashes or parentheses, such as 2125553856. The Character/Position format object can add parentheses and a dash in the appropriate places, such as in (212)555-3856. Along with other characters, each template string contains one or more placeholder characters. When a String variable needs to be displayed, each placeholder character is sequentially replaced by a character in the current value. The output string is meant to fit the pattern of the format string.

Example 33 illustrates how the Character/Position Format modifies the display of a string value.

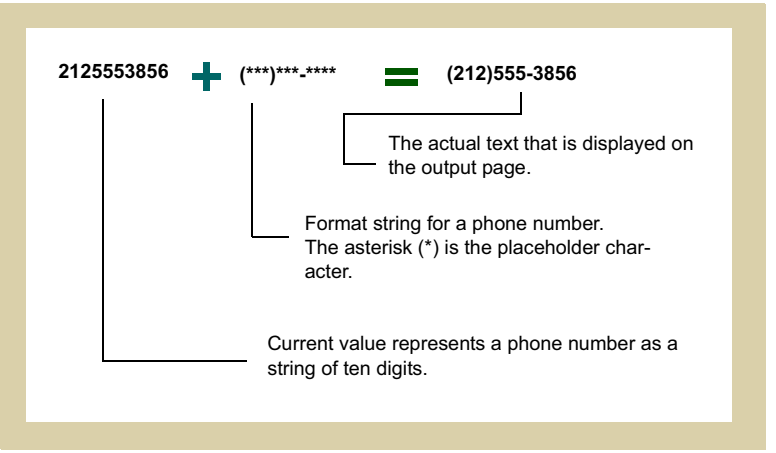


Figure 33: How character position format works

You modify the parameters in the Edit Character/Position Format dialog box (Figure 34).

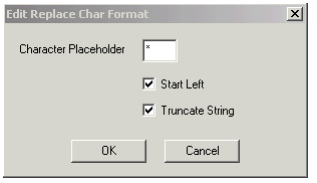


Figure 34: Edit Replace Char Format dialog box

Table 19 describes valid parameters for the Character/Position format.

Table 19: Replace Character format string

Parameter	Description
Character Placeholder	The character in the format string that will be interpreted as the placeholder. The default is an asterisk (*).

Table 19: Replace Character format string (continued)

Parameter	Description
Start Left (default)	Check to begin replacing placeholder characters starting with the first character in the string and moving left to right in the format string. Otherwise, it will start with the last character in the data string and replace the placeholder characters moving right to left in the format string.
Truncate String	Deletes the rest of the string if there are more placeholder characters in the string than characters in the current value. Any characters in the format string after the last used placeholder will not be displayed.

Date format

The Date format uses predefined format strings to display Date or Timestamp variables. The strings are displayed in Figure 35. Table 20 shows the output for each string.

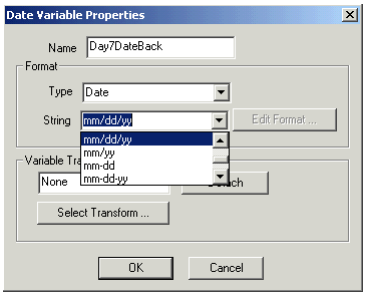


Figure 35: Date Variable Properties dialog box

Table 20: Date format string examples

Format string	Example
mmmm dd, yyyy	March 16, 1993
m/d/yy	3/16/93

Table 20: Date format string examples (continued)

Format string	Example
mm/dd/yy	03/16/93
mmm. dd	Mar 16
mmmm dd	March 16
mmm yy	Mar 93
dd-mmm-yy	03-Mar-93

Time format

The Time format uses predefined format strings to display Time or Timestamp variables. You select the format string in the Time Variable Properties dialog box (Figure 36). Table 21 shows examples of each string.

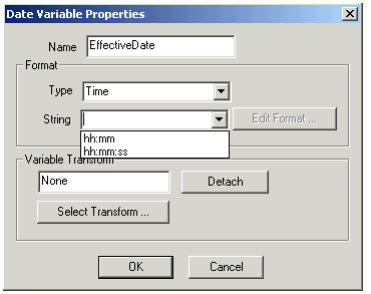


Figure 36: Time Variable Properties

Table 21: Time format strings

Format string	Example
hh:mm:ss	12:03:23
hh:mm	12:03

Working with PatternStream Variables

You access the properties of variable from the Data Link tab of the PatternStream main dialog (Figure 37).

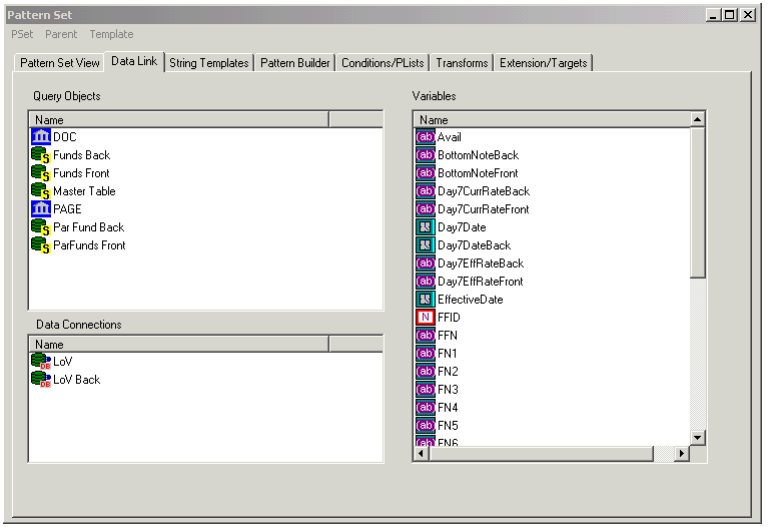


Figure 37: Data Link tab

Creating a variable

To create a new variable, follow these steps:

1 Select the **Data Link** tab In the Variables window, right-click and select **New Object.** The Create New Variable Object dialog box is displayed (Figure 38).

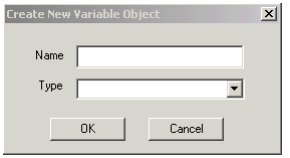


Figure 38: Create New Variable Object dialog box

2 Type the variable's name in the Name field. Variables must have unique names, although variables can have the same name as other objects in the PatternStream application, such as queries or targets.

3 Select a variable type (described in Table 15 on page 59) from the Type drop-down list, then select **OK.**

The variable properties dialog box for the selected variable is displayed. For detailed instructions about the settings in the dialog box for a particular variable, refer to the sections that follow.

Note: Select the variable's type carefully. Though you can later modify a variable's name or properties, you cannot change its type.

After you complete the dialog box for the specified query type, the variable and its icon are displayed in the Variables section of the **Data Link** tab. Variables are listed in alphabetical order.

Modifying variables properties

You modify a variable through its dialog box. You can access the properties dialog for a specific variable in two ways:

- From the Variables section of the **Data Link** tab, select the variable name, right-click, and select **Properties** from the pop-up menu
- From the **Pattern Set View** tab, select the variable name under the Variables folder (if the variables are not displayed, select the **+** beside the folder), right-click, and select **Properties** from the pop-up menu.

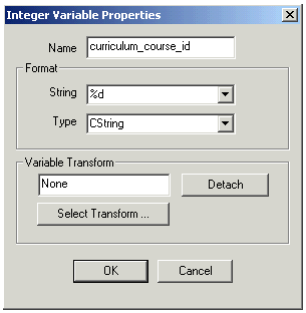
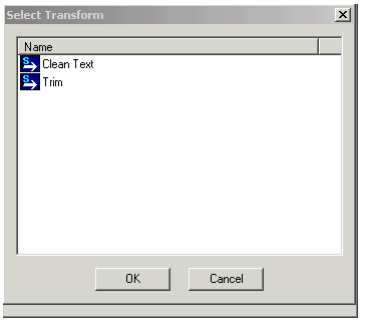


Figure 39: Typical variable properties dialog box

.

Table 22: Common variable attributes

Parameter	Description
Name	Edit the name field to assign or modify the name of a variable. When you change the name of a variable, all objects reference the variable will automatically use the new name.
Format	
Type	Select the type of format object from the drop-down list. The list of valid format types depends on the type of variable.
String	You can either type in the format string or where appropriate, select it from the drop-down list.
Edit Format	This button is enabled if the chosen format type has parameters associated with it. Click on this button to display the Format edit dialog.
Transforms	
Select Transform	Click on the Select Transform button. This displays the Select Transform dialog.  Select the desired transform, then click OK.
Detach	Click on the Detach button to detach a transform

Finding references

Many times it is necessary to find out which objects are using a particular variable. To find these references do one of the following:

- Select the **Data Links** tab of the PatternStream main dialog, select the particular variable, right-click and select **References** from the object popup menu.

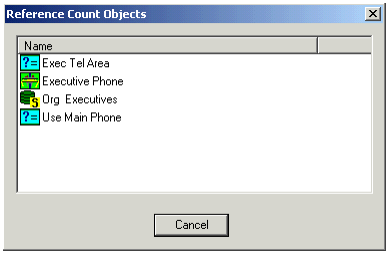


Figure 40: Variable Reference dialog

- Select the PatternView tab of the PatternStream main dialog. Below the pset hierarchy, there is a folder containing a list of all variable objects for the current pattern set in alphabetical order. Select the desired variable, click on the right mouse button, then select **References** from the popup menu.

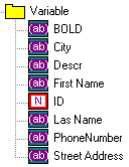


Figure 41: Variable list from the PatternView tab.

Copying a variable

If you are creating a variable of the same type as an existing one, sometimes it is easier just make a copy of it.

To copy a variable, follow these steps:

1. From the **Data Link** tab, select the variable you want to copy.
2. Right mouse-click, then select **Copy** from the object popup menu.

3 The Copy Variable Object dialog box is displayed, showing the name of the copied variable (Figure 42).

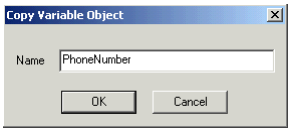


Figure 42: Copy Variable Object dialog box

4 Type the name you want for the new variable, then select **OK**. The variable properties dialog for the type of variable copied will be displayed.

5 Modify the attributes appropriately then close the dialog.

6 The new variable now appears in the variable list on the **Data Link** tab.

Deleting a variable

Because you can delete only those variables not currently used by other objects, you must remove the variable's reference from all objects before deleting the variable. See "Finding references" for details about determining if a variable is currently in use.

To delete a variable, follow these steps:

1 Remove references to the variable from all other objects in the PatternStream application. See Table 23 for details.

Table 23: Removing variable references from objects

Object	To remove the variable reference...	For details on editing the object, see...
Query	Unbind the variable from its associated column or parameter or just delete the query	Chapter 6, "Constructing queries"
Condition	Edit the conditional expression by selecting another variable or just delete the condition.	Chapter 16, "Creating conditions"
String template	Find the segment that contains the variable, then delete the segment or edit its properties and select another variable.	

Table 23: Removing variable references from objects

Object	To remove the variable reference...	For details on editing the object, see...
Parameter List	Find the parameter list that uses the variable, then either delete the list entry or assign another variable	
Target	Many targets allow you to use variables to assign attribute values. Find the target that references the variable and either assign another variable or use a static value for the attribute.	
Transform	Some directives determine the operation to perform from an assigned variable (e.g., string concatenation). Either delete the directive, assign another variable or use a static value (if appropriate).	

2 Select the variable you want to delete from the Variables section of the **Data Link** tab.

3 Right-click and select **Delete** from the pop-up menu. A confirmation dialog box is displayed.

4 To delete the variable, select **OK**.

String and Text Variables

String variables

When you create a String variable, the String Variable Properties dialog box is displayed (Figure 43). Besides all of the standard parameters variable have an

additional attribute that specifies how much memory to allocate for storing string values.t

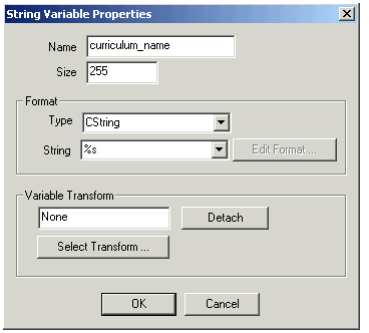


Figure 43: String Variable Properties dialog box

Table 24: String variable parameters

Parameter	Description
Size	Specify the maximum length of the character string (the number of characters and spaces) in the Size field. Try to estimate the string length as accurately as possible. Overestimating the length may fragment memory, while underestimating may result in truncated values. The default value is 50 and maximum value is 2024. *Note:* If you need a larger size, then use a text variable.

Text variables

Use a Text variable for large blocks of text. Typically, this variable type is required when retrieving data from a database field with a data type of Memo (MS Access), Text (SQL Server), or clob (Oracle).

When you create a Text variable, the Text Variable Properties dialog box is displayed (Figure 44).

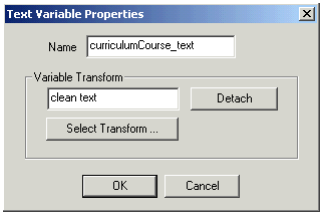


Figure 44: Text Variable Properties dialog box

Note: Text variables do not use format objects.

Reference variable

Reference variables are used when there are multiple derived values of an initial value that are required to track each other throughout the pattern set. The valid format types you can assign and the allowed transform types you can attach will depend on the type of variable that is referenced. When you create a Reference variable, the Reference Variable Properties dialog box is displayed (Figure 45).

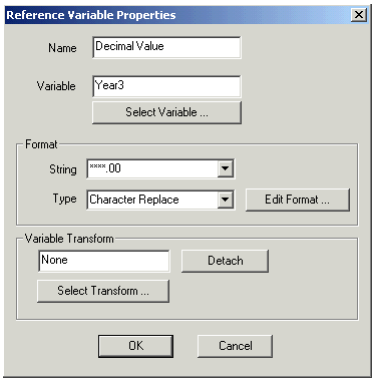


Figure 45: Reference Variable Properties dialog box

Table 25: String variable parameters

Parameter	Description
Variable	Enter the name of the variable that the reference variable will derive its value from
Select Variable	Click on the Select Variable button. This brings up the Select Variable dialog. Select the variable that the reference variable will derive its value from and click OK.

Chapter 6: Constructing queries

In the PatternStream application, the patterns you create are driven by *queries*, which are sets of instructions that retrieve data from some external data source, usually a database. Patterns and queries are closely tied—patterns specify how information is structured and formatted in your document, and queries specify what information to obtain and how to obtain it.

For example, in a *select query* an SQL statement is sent to the DBMS through its associated *connection* object. The DBMS returns a set of records as logical rows that satisfy the given query. This set of rows, called the result set, drives the execution of the *pattern* object to which the select query is attached. A logical device, called a cursor, points to each successive row in turn, starting with the first row. Each field, or column in the result set is bound to a PatternStream variable, and the value stored in this variable changes each time the cursor moves to the next row. Also, each time the cursor moves, the pattern, through the targets it contains, generates structural elements in the current document such as paragraphs and tables. The values stored in the variables are used to control execution flow, create text and import external files such as graphic and text insets.

All query objects use this model to drive the execution of the PSet file, whether they actually correspond to an SQL Select statement or retrieve data through a stored procedure or a flat file.

This chapter describes how to create and modify queries in the PatternStream application. For an overview of the available query types, see Table 26 on page 81.

Query construction basics

To create effective queries, you should be familiar with your database and have at least a basic understanding of the SQL programming language.

The following are some ideas to keep in mind when working with query objects:

- Write and test your SELECT queries in a dedicated SQL tool before typing them into the PatternStream application. Debugging a query from within the PatternStream application may be difficult because you may not be able to separate the effects of a query from the effects of formatting in a document, especially if your pattern set uses conditions.
- Confirm that there are valid connection objects to the external data source. The first step performed when a pattern set is run is to establish a valid transaction to the data sources pointed to by each connection object. Running a

pattern set is then a good way to validate these connections before getting started with the query objects, themselves.

- Try to anticipate the variables you will need and create them before entering the query properties dialog.
- If the SQL syntax for a certain select query is especially complex, and you work in an organization with an IT department you may need to ask your DBA to construct a view or stored procedure to simplify the queries you need to input into PatternStream.

Working with Query Objects

Query objects are accessed through the Data Link tab of the PatternStream main dialog (Figure 46). The query objects belonging to the current pattern set appear in the upper left window of the dialog in alphabetical order. To perform an operation on an existing query, select the icon of the desired query, then click on the right mouse button. Choose the operation you wish to perform from the object popup menu.

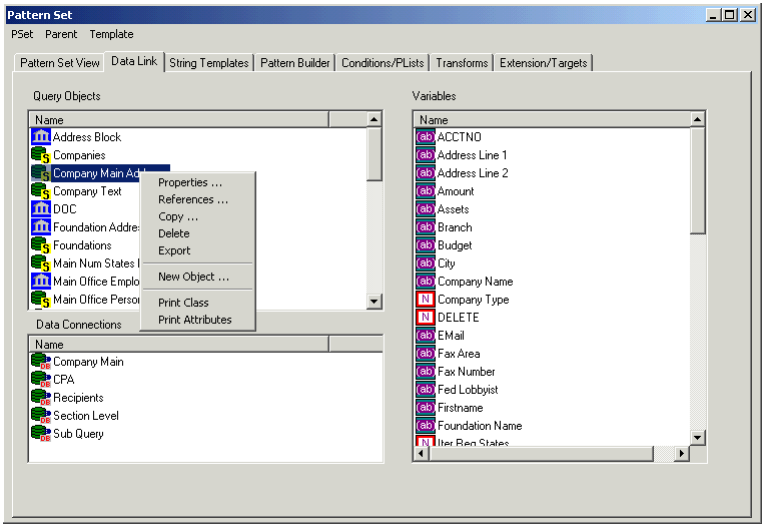


Figure 46: Data Link tab

Creating a new query

To create a new query object, follow these steps:

1 Click on the white area of the query window with the right mouse button and select **New Object** from the popup menu. The **Create New Query Object** dialog box is displayed (Figure 47).

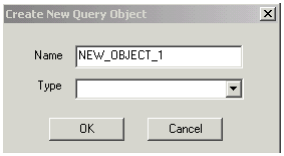


Figure 47: Create New Query Object dialog box

2 Type the query's name, select a query type (described in Table 26) from the drop-down list, then select **OK**.

Table 26: Query types

Type	When to use	Description	For details see...
FLAT FILE	With flat file databases, such as .txt files	To define a FLAT FILE query, you specify the characters that indicate the beginning and end of a record and the field delimiter, then you bind variables to the fields.	"The FLAT FILE query" on page 86
SELECT	With relational databases	Based on information you provide, the PatternStream application creates a SELECT statement in SQL language to specify what data to retrieve from the database.	"Defining a SELECT query" on page 88
STATIC	With patterns whose execution is not data-driven	The STATIC query is unique to the PatternStream application. Use a STATIC query with patterns whose execution is not data-driven. The query specifies a fixed number of times the pattern set cycles through the pattern.	"Defining a STATIC Query" on page 108

Table 26: Query types (continued)

Type	When to use	Description	For details see...
STORED PROCEDURE	With relational databases where the complex SQL statement or stored procedure code is already written	A STORED PROCEDURE query returns values or parameters from the database. Instead of defining the query in the PatternStream application, as you do with a SELECT statement, you link to the already-written query and specify input and output parameters.	"Defining a STORED PROCEDURE query" on page 109
CUSTOM	When it is necessary to access data from a data source not supported by PatternStream.	Provides entry points and a calling interface for routines exported from a dynamic link library that allow user written data access functions to fit into the Query object model.	

3 A query properties dialog box is displayed. For detailed instructions about defining the properties for each query type, refer to the sections that follow (see Table 26 for page references).

Note: In some areas of the query definition process, the PatternStream application assigns items in a list an index number, which helps the system track those items. The index numbers are arbitrary and do not indicate order.

After you complete the dialog box for the specified query type, the query and its icon are displayed in the Query Objects section of the **Data Link** tab. Queries are listed in alphabetical order.

Query info tab

All query objects, regardless of type, have common attributes which are accessible through the Query Info tab.(Table 27).

Table 27: Query Info tab options

Attribute Name	Description
Query Name	Type in the name of the query object. *Note:* This name must be unique for objects in the query class, regardless of the type of query

Table 27: Query Info tab options

Attribute Name	Description
Description	Enter a description of the query object here for documentation purposes. This description will be displayed when the query object's attributes are printed out.
Connection Object	All query objects (with the exception of static queries) must be associated with a connection object. When the actual query is sent to the DBMS, it through this connection object that this transaction is controlled. The following table lists what type of connection is valid for the different query types.
Iteration Variable	If you select an integer variable from the drop-down list, then the value of the variable will be initialized to zero when the query gets executed (i.e., is sent to the DBMS) and incrementally increased each time a row is fetched from the result set.

Query Object Type	*Connection Type*
Select	ODBC
Stored Procedure	ODBC
Flat File	Flat File
Static	NONE
Custom	Custom

Copying a query

If you are creating a query that is similar to an existing one, you can save time by making a copy of the original query and modifying its properties.

To copy a query, follow these steps:

1. From the **Data Link** tab, select the query you want to copy.
2. Right mouse-click, then select **Copy** from the object popup menu.

3 The Copy Query Object dialog box is displayed, showing the name of the copied query (Figure 48).

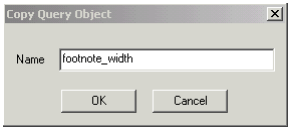


Figure 48: Copy Query Object dialog box

4 Type the name you want for the new query, then select **OK**. The query properties dialog for the type of query selected will be displayed.

5 Modified the attributes appropriately then close the dialog.

6 The new query now appears in the query list on the Data Link tab.

Finding references

Many times it is necessary to find out which pattern objects are using a particular query. To find these references do one of the following:

- Select the Data Link tab of the PatternStream main dialog, select the particular query, right-click and select **References** from the object popup menu.

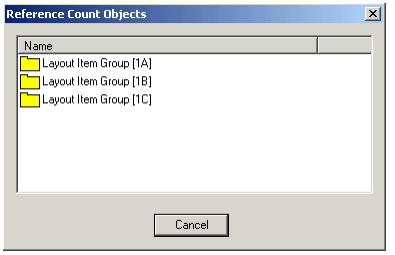


Figure 49: Query Reference dialog

- Select the PatternView tab of the PatternStream main dialog. Below the pset hierarchy, there is a folder containing a list of all query objects for the current

pattern set in alphabetical order. Select the desired query object, click on the right mouse button, then select **References** from the popup menu.

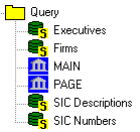


Figure 50: Query list from the PatternView tab.

Deleting queries

Before you can delete a *query*, you must detach it from all of the patterns that reference it (see "Finding references" and "Detaching a query").

To delete a query, follow these steps:

1. Select the particular query from the alphabetized list of query objects in either:
   - the query window in the upper left corner of the **Data Link** tab
   - the query folder on the **PatternView** tab.
2. Click the right mouse button and select **Delete** from the popup menu.

Assigning a Query to a Pattern

The only way to execute a query when a pattern set is run is to attach it to a pattern. The results of the query will then control the execution of pattern. The pattern will cycle for as many rows as the query returns. If the pattern itself is a sub-pattern to another pattern then it can be executed more than once, hence any query can be executed any number of times. You can also attach a query to more than one pattern.

Note: You must be careful not to have the same query nested with itself.

To assign a query to a pattern, you must first access the properties dialog for the desired pattern.

1. Select the Pattern Builder tab of the PatternStream main dialog.
2. Find the desired pattern in the pattern list on the left side of the dialog and select it. This makes it the current pattern.
3. Select Object Info from the parent menu. This brings up the Pattern Info dialog.

Attaching a query

To attach a query the current pattern, click on the Select Query button, then select the desired query from the Query Selection dialog.

Detaching a query

To detach a query from a pattern, click on the Detach button.

Note: Even though a query is not attached to any pattern, it can still generate database errors when the pattern set is run.

The FLAT FILE query

Use a FLAT FILE query when the data is contained in a flat file, such as a tab delimited ASCII text file. In reading data from a flat file, we first assume that the file is composed of a series of structurally identical records. Each record consists of a fixed number of fields separated by a delimiter, typically a tab or a comma. Figure 51 illustrates how a flat file structure is defined. We then bind a Pattern-Stream variable to each field. Each record is then read from the file one at a time, with each bound variable taking the value of its associated field for the current record. To specify the variable bindings for a flat file query, open the query dialog and select the Variable Binding Tab.

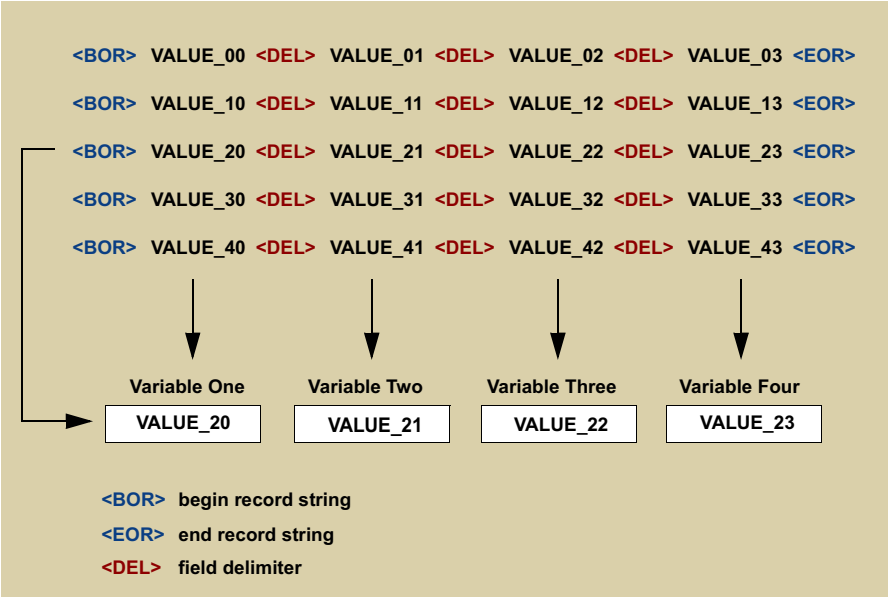


Figure 51: Logical breakdown of a typical flat file.

Variable Bindings tab

4 This tab allows you to bind PatternStream variables to the delimited fields of an ASCII text file. From the **Flat File Query Properties** dialog select the **Variable Bindings** tab (Figure 52)..

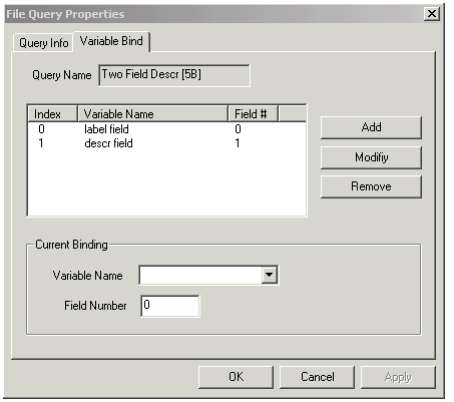


Figure 52: Variable Bindings tab

Table 28: Query Info tab options

Operation	Procedure
Add a variable binding	**1** Select the desired variable from the variable drop-down list. **2** Type in the field number that the variable is to be bound to. *Note:* The field number start at zero, from the left side of the record. **3** Click on the **Add** button.

Table 28: Query Info tab options

Operation	Procedure
Modify an existing binding	**1** Select the desired binding by clicking on the index number in the binding list. This then becomes the current binding and its attributes are displayed in the current binding area. **2** Modify the attributes in the current binding area (variable name and field number). **3** Click on the **Modify** button. The modified attributes will then appear in the binding list. *Note:* The changes will not take effect until the Modify Binding button is clicked, even though the changes were made in the current binding area.
Remove a variable binding	**1** Select the desired binding by clicking on the index number in the binding list. **2** Click on the Remove button. The selected binding is removed from the binding list. *Note:* Fields not bound to a variable are ignored.

Defining a SELECT query

Use a SELECT query to access a relational database. Based on the information you provide in the Select Query Properties dialog box (Figure 51 on page 86), the PatternStream application constructs an SQL statement (which you can view in the **View SQL** tab).

Most of the tabs in the **Select Query Properties** dialog box represent a clause in the syntax of a SQL statement. These SQL clauses and there corresponding dialog tabs are listed in Table 29.

Table 29: SQL syntax and their corresponding dialog tabs

Dialog Tab	SQL Statement Clause	For details see...
Select	Specify which columns in the database the query will access	"The SELECT clause—specifying columns" on page 92
From	Specify which tables in the database the query will access	"The FROM clause—specifying tables" on page 90

Table 29: SQL syntax and their corresponding dialog tabs

Dialog Tab	SQL Statement Clause	For details see...
Where	Specify restrictions on which records the query returns.	"The WHERE clause—restricting records" on page 96
Order By	Specify the order that the records will be returned from the database.	"The ORDER BY clause—setting sort order" on page 105
Group By	Specify the grouping of the result set records when aggregate functions are used (i.e., SUM(), AVG(), MAX(), etc.).	"The GROUP BY clause" on page 107

Note: To simplify the SELECT queries you create in the PatternStream application, create (or ask your DBA to create) views in the database. Views let you organize and manipulate your data beforehand so that your queries are simpler and easier to debug. Also, by using views, the pattern set becomes less dependent on the implementation details of a specific database.

The FROM clause—specifying tables

1 This section describes how to build the FROM clause of a SELECT statement. The FROM clause specifies a list of tables from which the data will come.

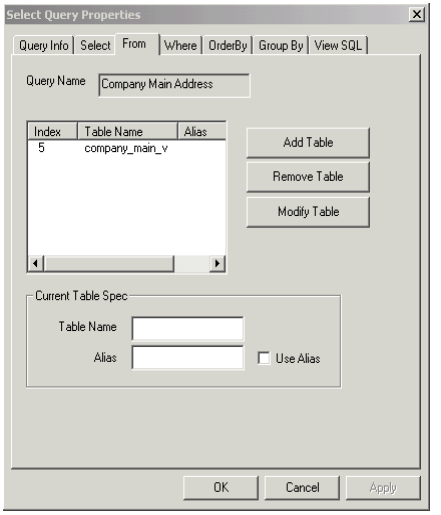


Figure 53: From tab, Select Query Properties dialog box

Note: This dialog is an example of a component edit dialog described in "Component Edit Dialog" on page 39.

Table 30: Query Info tab options

Operation	Procedure
Add a table name	**1** Type in the name of the table in the Table Name field in the current table spec area. **2** Optionally you can type in the name of an alias. This alias can simplify the reading of the SQL syntax by substituting a simpler name when more than one table is required (i.e., for inner joins). Make sure you also check the Use Alias box in order for the alias name to be used in building the SQL syntax. *Note:* If you change the alias name, it will be changed in all other instances where that alias is used in the current query. **3** Click on the Add button. A new table name is added to the list of tables.
Modify an existing table name	**1** Select the desired table name by clicking on the index number in the table name list. This then becomes the current table name and its attributes are displayed in the current table spec area. **2** Modify the attributes in the current table spec area (table name, alias name and alias flag). **3** Click on the **Modify** button. The modified attributes will then appear in the table name list. *Note:* The changes will not take effect until the Modify button is clicked, even though the changes were made in the current table spec area.
Remove a table name	**1** Select the desired table name by clicking on the index number in the table name list. **2** Click on the Remove button. The selected table name is removed list. *Note:* Any syntactic elements in the query that reference a given table name will generate a syntax error when a table name is removed.

The SELECT clause—specifying columns

This section describes how to build the SELECT clause of a SELECT statement. Besides indicating the columns to extract from, you also need to bind PatternStream variables to each of these columns. It is then through these variables that the data is made accessible to the other objects in the pattern set. This dialog is similar to the other component object dialogs.

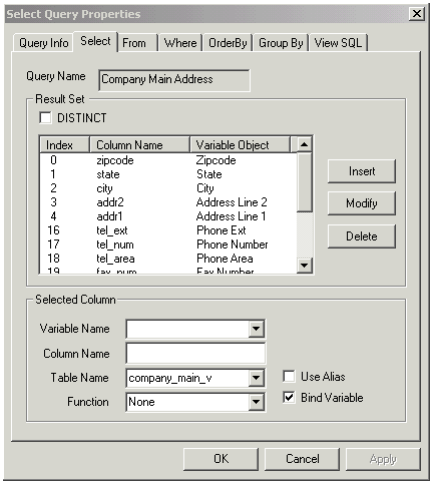


Figure 54: Select tab, Select Query Properties dialog box

The attributes that you can specify are listed in Table 31.

Table 31: Selected Column attributes

Attribute Name	Description
Variable Name	From the **Variable Name** drop-down list, select a variable to associate with this column. *Note:* Confirm that the variable's data type is consistent with the data type of the column to which it is associated. For example, an INTEGER variable assigned to a column containing character data will produce an error in most database systems.

Table 31: Selected Column attributes

Attribute Name	Description
Column Name	In the **Column Name** field, type the column's name as it appears in the database. *Note:* Technically, you are not required to just use a single column name. Rather, if required, you can type in any valid expression. See "Performing mathematical operations on columns" for more details.
Table Name	From the Table Name drop-down list, select the table that contains this column (if you specified an alias for the column in the **From** tab and want to refer to the alias here, select the **Use Alias** check box.

Table 31: Selected Column attributes

Attribute Name	Description
Function	To apply a set function or *aggregate* to a column, select one from the drop-down list. The table below describes the available aggregates. These aggregates perform a mathematical operation on a column's values for all records in a result set that group together based on the group by clause.

Aggregate Function	*What it does*
AVG()	Takes the average of all the values of a column in the same group.
SUM()	Sums all the values of a column in the same group
MAX()	Returns the maximum all the values of a column in the same group
MIN()	Returns the minimum all the values of a column in the same group
COUNT()	Returns the number of rows (usually used as COUNT(*)) in each grouping.

Note: In most database systems, when applying an aggregate function to a column, if one of the values being grouped is null, then the aggregate result is null also.

Performing mathematical operations on columns

You may need to perform a mathematical operation on a returned record or on records from two different columns to get the results you want. For example, the following SELECT statement combine the values for a company's first three months' sales and returns the sum:

```
SELECT january + february + march
FROM sales
```

To include a mathematical operation in the SELECT statement, type the equation into the Column Name field of the **Select** tab. (For example, if you want to see what your books' prices would be with sales tax added, you could type `books * 1.06` in the Column Name field.)

- Supported operations are addition (`+`), subtraction (`-`), multiplication (`*`), and division (`/`).
- By default, SQL performs `*` and `/` operations first, then operations of equal precedence from left to right. Use parentheses to control the order in which operations are performed. In this example, a dollar is discounted from each book's price, then sales tax is added:
  ```
  SELECT (book - 1) * 1.06
  ```

Note: The Select clause tab is an example of a component object dialog. The following table describes how to add, modify and remove the column specifications for a particular select query.

Table 32: Building the select clause.

Operation	Procedure
Add a column specification	**1** Edit the attributes in the selected column area of the dialog. **2** Click on the Add button. A column specification is added to the list of columns.

Table 32: Building the select clause.

Operation	Procedure
Modify an existing column specification	**1** Select the desired column specification by clicking on the index number in the column list. This then becomes the current selected column and its attributes are displayed in the selected column area. **2** Modify the attributes in the selected column area (Table 31). **3** Click on the **Modify** button. The modified attributes will then appear in the column list. *Note:* The changes will not take effect until the **Modify** button is clicked.
Remove a column specification.	**1** Select the desired column by clicking on the index number in the column list. **2** Click on the Remove button. The selected column is removed from the column list.

The WHERE clause—restricting records

To restrict which records are returned in the query, you must construct a *filter*, which is a logical expression (for example, chapter = 4) that the rows returned by the query must satisfy.

Table 33 describes the filter types available in the PatternStream application.

Table 33: Filter types

Type	Use for...	For details, see...
Simple Predicate	A simple expression. For example: `WHERE` chapter = 4)	"Modifying a Simple Predicate (or expression)" on page 100
Compound Predicate	A combination of two or more predicates connected by the logical operator `AND` or `OR`. For example: `WHERE (chapter is = 4 AND name IS NOT NULL)`	"Modifying a Compound Predicate (AND and OR)" on page 103

By combining simple expressions and compound predicates you can create as complex a where clause as you need. A good way to visualize this is to picture the filter as a logic expression tree. Figure 55 illustrates how this is done.

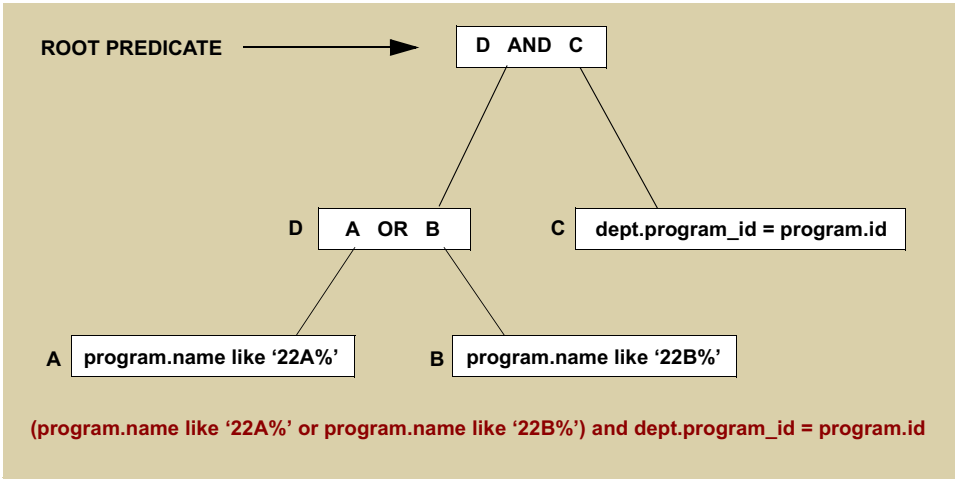


Figure 55: Logic tree structure of a compound predicate

.At the top of the Where tab, there is a list of all of the predicates belonging to a particular select query. Not all predicates are necessarily used in the complete filter. What predicates are used depends on how the logical expression tree

expands. This means that you can have alternate versions of the filter and switch from one to the other as you develop the pattern set logic.

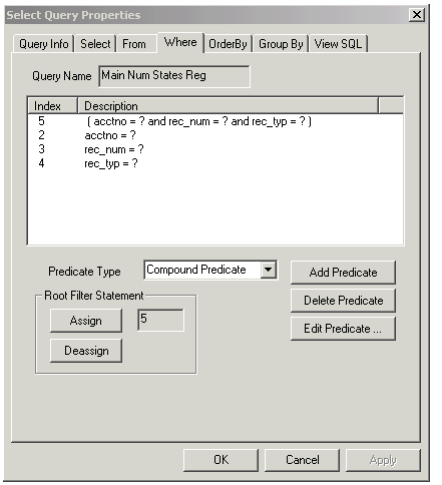


Figure 56: Where tab, Select Query Properties dialog box

.Table 34 describes how to create predicates and to generally put them together to create the final filter. Then each predicate type is discussed in detail.

Table 34: Working with the Where tab

Operation	Procedure
Add a new predicate	**1** Select the type of predicate from the Predicate Type drop-down list. Currently, there are only two types of predicate: • Simple predicate, which are just expressions. • Compound predicates, which are two or more predicates separated by AND or OR. **2** Click on the **Add Predicate** button. A new predicate is added to the list with the following default description: • NOOP if it is a simple predicate. • () if it is a compound predicate.

Table 34: Working with the Where tab

Operation	Procedure
Modify an existing predicate	**1** Select the desired predicate by clicking on the index number in the predicate list. **2** Click on the **Edit Predicate** button. The dialog for the type of predicate selected is displayed. **3** Modify the predicate attributes, then click OK to incorporate these changes.
Delete a predicate	**1** Select the desired predicate by clicking on the index number in the predicate list. **2** Click on the **Delete Predicate** button. The selected predicate is deleted and all references to that predicate are removed.
Assign the Root Predicate	**1** Select the desired predicate by clicking on the index number in the predicate list. **2** Click on the **Assign** in the root predicate area at the lower left of the dialog.
Deassign the Root Predicate	**1** Click on the **Deassign** button. *Note:* Deleting a predicate will automatically deassign it as the root predicate.

Modifying a Simple Predicate (or expression)

Modify a simple predicate through the **Edit Simple Predicate** dialog (Figure 57).

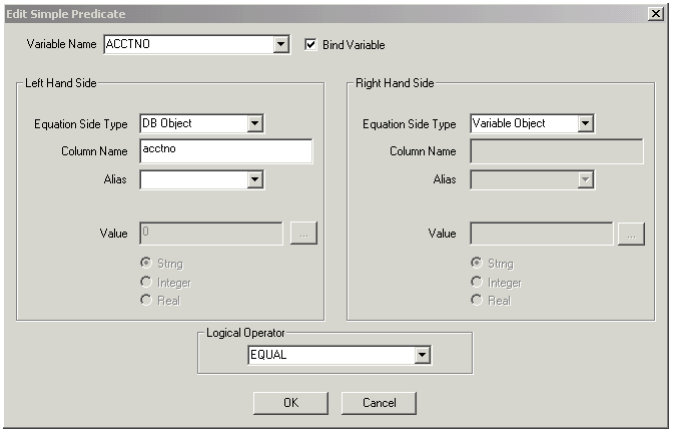


Figure 57: Edit Simple Predicate dialog box

1 Starting with the left side of the expression, select an object type from the Equation Side Type drop-down list, then specify the name or value for the left side of the expression. See Table 35 for a description of the selections.

Table 35: Valid object types for each side of the expression

Object type	Description	To specify its name or value...
DB Object	A column name	Type the column name in the Column Name field. Optionally, you can select an alias name from the alias drop-down list. *Note:* When joining two tables that have the same column name, then using the alias is NOT optional. You must qualify the column name with the table name (or alias).
Literal	A value, such as a number or a string	Select among String, Integer, and Real (any noninteger number), then type a value in the Value field. Do not enclose a literal value in single quotes; the PatternStream application will add the quotes when constructing the SQL statement. If the expression is long, you can open a dialog window that contains an edit field that scrolls by clicking on the edit button (...).
Variable Object	A variable name. You must first define the variable before assigning it to an expression. See Chapter 5, “Variables,”for details.	Select a variable name from the drop-down box near the top of the dialog box. You can use only one variable in a simple predicate. You will typically use a variable in an expression in sub-queries where the value of the variable was retrieved from a higher level query. For an example of this see Figure 3 on page 17 *Note:* The bind check box automatically checks and un-checks when the variable is selected.

Table 35: Valid object types for each side of the expression

Object type	Description	To specify its name or value...
Custom	A functional expression.	This type allows you insert complex functional expressions that will not be treated as a string. No adjustment to the syntax will be made (i.e.,. inserting single quotes).

Note: You can indicate wildcard characters in a string value. To specify one wildcard character, type an underscore (_); for more than one wildcard, type a percent sign (%). For example, for A_ (used in conjunction with the LIKE logical operator), the query will look only for two-letter values that begin with A. For A%, the query will look for any values beginning with A.

2 Select a logical operator from the Logical Operator drop-down list. See Table 36 for details.

Table 36: Logical operators for filters

Operator	SQL symbol	Description[a]
EQUAL	=	True if lhs equals the rhs.
NOT EQUAL	<>	True if lhs does NOT equal the rhs.
GREATER THAN	>	True if lhs is greater than the rhs.
GREATER THAN EQUAL TO	>=	True if lhs is greater than or equal to the rhs.
LESS THAN	<	True if lhs is less than the rhs.
LESS THAN EQUAL TO	<=	True if lhs is less than or equal to the rhs.
LIKE	LIKE	Used with the wildcard character. True if the lhs matches the pattern of the rhs.
NOT LIKE	NOT LIKE	Used with the wildcard character. True if the lhs does not match the pattern of the rhs.

Table 36: Logical operators for filters (continued)

Operator	SQL symbol	Description[a]
IS NOT NULL	N/A	True if the lhs is null. *Note:* There is a difference between a null value and a blank string. Sometimes a query does not work as expected because a blank string is thought to be a null value.
IS NULL	N/A	True if the lhs is not null.

a. In this table, `lhs` refers to the left side of the equation, and `rhs` stands for the right side.

3 Repeat step 1 for the right hand side of the expression.

Modifying a Compound Predicate (AND and OR)

You modify compound predicates through the **Edit Compound Predicate** dialog (Figure 58).

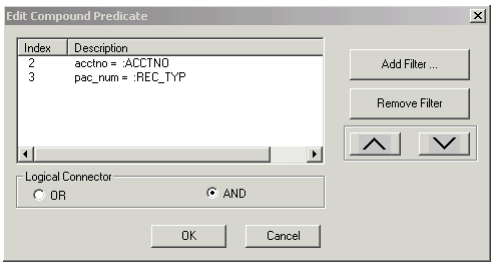
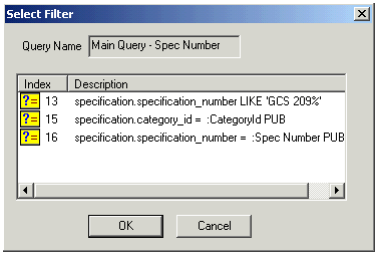


Figure 58: Compound Predicate dialog box

.

Table 37: Working with the Where tab

Operation	Procedure
Select a logical operator	A compound predicate is a collection of predicates separated by either the logical connector AND or OR. Click on the desired connector. If AND, then all of the predicates in the list must evaluate to TRUE for the compound predicate to be TRUE. For the OR statements, at least one of the predicates must be TRUE for the compound predicate to evaluate to TRUE.
Add a predicate to the list	**1** Click on the Add Predicate button. This brings up the Select Predicate dialog.  **2** Select the icon of the desired predicate. **3** Click **OK** to insert the selected predicate into the predicate list.
Remove a predicate from the list	**1** Select the icon of the desired predicate. **2** Click on the Remove button. **3** This removes the predicate from the list.
Change the predicate order	Sometimes, depending on the DBMS, the order of the expressions in a where clause makes a difference in performance. To change the order of the predicates in a compound predicate: **1** Select the predicate whose order you wish to change. **2** Click on the up arrow to move it close to the top and the down arrow to move it lower.

The ORDER BY clause—setting sort order

Use the ORDER BY clause to specify what order the rows in the result set will be returned by the database. Select the **Order By** tab (Figure 59).

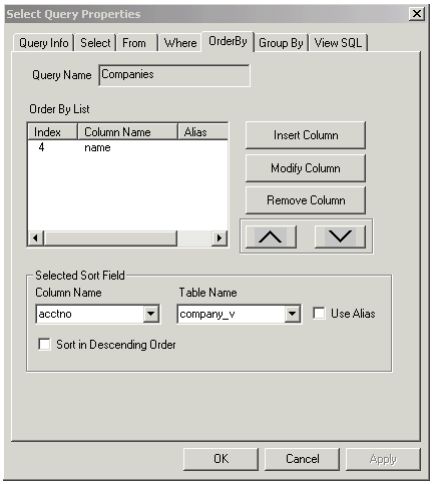


Figure 59: Order By tab, Select Query Properties dialog box

You specify sort order by listing a series of columns, actually positions in the result set.

Table 38: Working with the Order By tab

Operation	Procedure
Add a column.	**1** Select the column from the drop-down list. Only columns used in the actual select statement appear in the list, but you can type in column names that are valid but do not appear in the select clause. You can also type in a number, representing the position a column appears in the select clause. This is necessary when the column specification is an expression rather than just a simple column name. **2** Select the table name from the drop-down list if desired. **3** If the sort on this column is to be in descending order, then check the box **Sort in Descending Order**. **4** Click on the **Add Column** button.
Modify a column specification	**1** Select the desired column by clicking on the index number in the predicate list. **2** The attributes appear in the selected sort field area. **3** Modify the desired attributes. **4** Click on the Modify Column button to incorporate the changes. *Note:* The changes will not be incorporated unless the modify button is clicked.
Remove a column	**1** Select the desired column by clicking on the index number in the column sort list. **2** Click on the **Remove Column** button. The selected column is removed from the sort order list.
Change the precedence order of the columns	**1** Select the column whose order you wish to change. **2** Click on the up arrow to move it close to the top and the down arrow to move it lower.

The GROUP BY clause

Use the GROUP BY clause to specify how you want query results to be grouped during aggregation. Select the **Group By** tab (Figure 59).

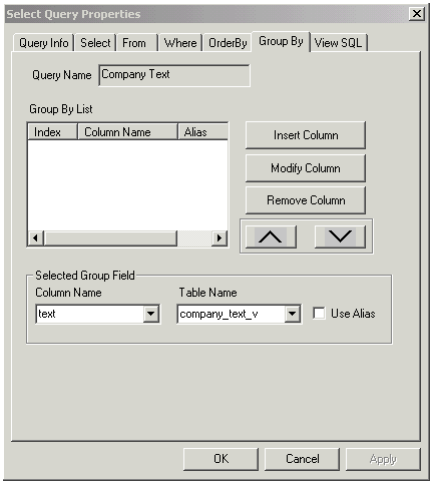


Figure 60: Order By tab, Select Query Properties dialog box

The Group By tab works identically to the Order By tab. See "The ORDER BY clause—setting sort order" on page 105 for details.

Viewing the Generated SQL Syntax

The PatternStream application creates a SQL SELECT statement based on the data you provide in the Select Query Properties dialog box. To view this code, select the **View SQL** tab (Figure 61).

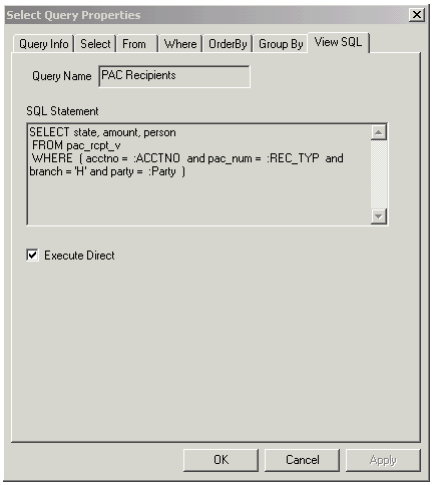


Figure 61: View SQL tab, Select Query Properties dialog box

The **Execute Direct** check box determines how the SELECT statement that the PatternStream application generates is handled in ODBC:

- If this box is checked, then the query will be executed new every time, as if the DBMS has never seen it before.
- If this box is unchecked, then an execution plan will be created at the beginning of the running of the pattern set that will be used every time the query is executed.

Defining a STATIC Query

A pattern cannot execute unless it is associated with a query, but some patterns are not data-driven. In such cases, create a static query. Unlike the other query types, a static query does not perform a query; instead, the static query signals its associated pattern to execute a set number of times. In essence, a static query serves the function of a DO or FOR loop in a more conventional language.

Cycles Tab(Figure 62).

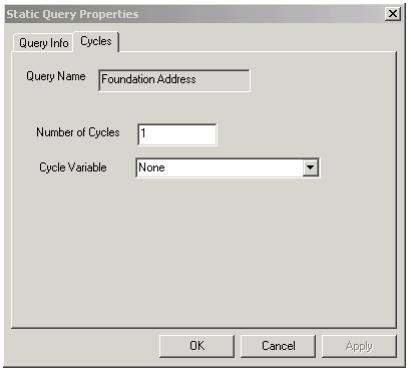


Figure 62: Query Info tab, Static Query Properties dialog box

4

Table 39: Filter list logical operators

Operator	Description
Number of Cycles	Specifies the number of times the pattern is to be performed. The default is one.
Cycle Variable	Select an integer variable from the drop-down list. The value of this variable each time the query is executed will determine how many times the associated pattern is executed.

Defining a STORED PROCEDURE query

A stored procedure is a program, written in a language that is specific to a particular DBMS, that retrieves results from a database like a select query, except the actual steps necessary to retrieve the result do not need to fit the normal relational database paradigm.

Before you define a STORED PROCEDURE query, you must create variables that will be bound to the query's return value or input or output parameters. The input parameters of a STORED PROCEDURE query correspond to the variables used in the WHERE clause of a SELECT query—the values of the variables in the output parameter list define what rows the STORED PROECEDURE query returns, just as the columns in a SELECT query.

As with a SELECT query, a STORED PROCEDURE query returns an arbitrary number of result rows. These rows are defined in the procedure itself, often as a SELECT statement.

Note: Unlike a SELECT statement, the order of the input and output parameters is significant.

Although the stored procedure itself can be very complex, working with a stored procedure query in PatternStream is quite simple.

Defining the stored procedure.

The stored procedure is declared using the **Define** tab (Figure 63).

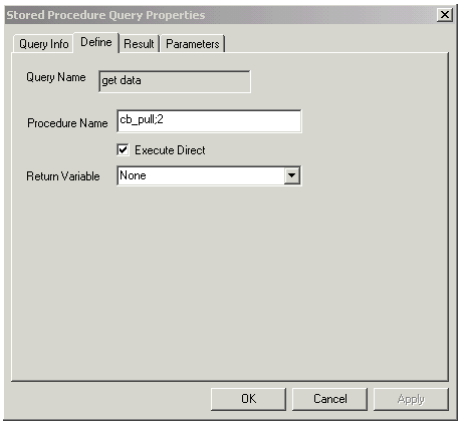


Figure 63: Define tab, Stored Procedure Query Properties dialog box

Declaring the Result set

You define the result set of a stored procedure using the **Result** tab (Figure 64).

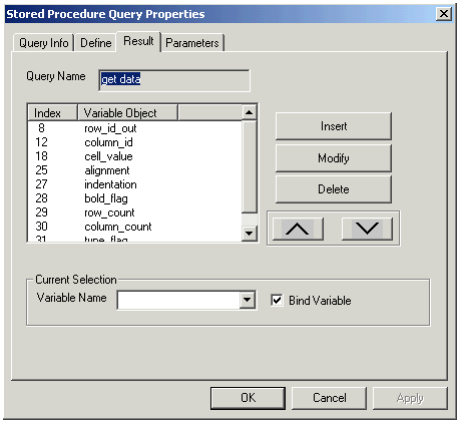


Figure 64: Result tab, Stored Procedure Query Properties dialog box

Declaring the Parameters

5 You define the parameters for a stored procedure using **Parameters** tab (Figure 65).

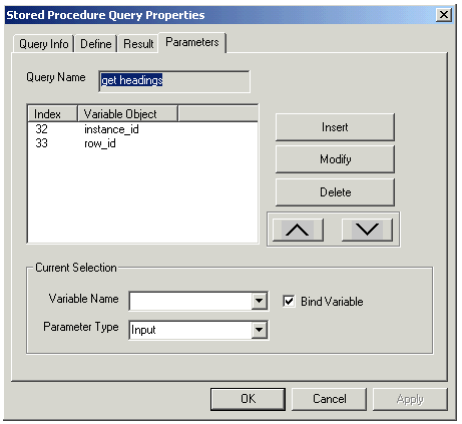


Figure 65: Parameters tab, Stored Procedure Query Properties dialog box

Chapter 7: Patterns

Patterns are objects that drive the execution of a pattern set through the action of their associated query object. A pattern represents a logical configuration of data that is repeatedly displayed. For example, in Figure 66, we have an example of a directory that displays a list of companies and some information associated with them. Notice how the configuration, or pattern, of information is repeated over and over again, even though the information itself changes from company to company. If we look further, we can see subpatterns within each company listing. For example, each executive is displayed as a name and a title, separated by a tab with leaders. This series of patterns and subpatterns define the data hierarchy of the output.

Some basic facts about patterns are:.

- Patterns cannot have another pattern as a parent. They can only be attached to the PSet Hierarchy as a subpattern to a target object.
- In order for a pattern to perform any action, it must have a query object attached. It is this query object that controls what a pattern will do when executed.
- A pattern can be executed any number of times but can appear in the PSet Hierarchy only once.
- A pattern is executed only of its parent target is generated.

Executing a Pattern

A pattern is executed when its parent target is invoked. If the parent target has more then one pattern attached to it, then these patterns will be executed in the order they appear in the parent target's pattern list. When a pattern is executed the following things happen:

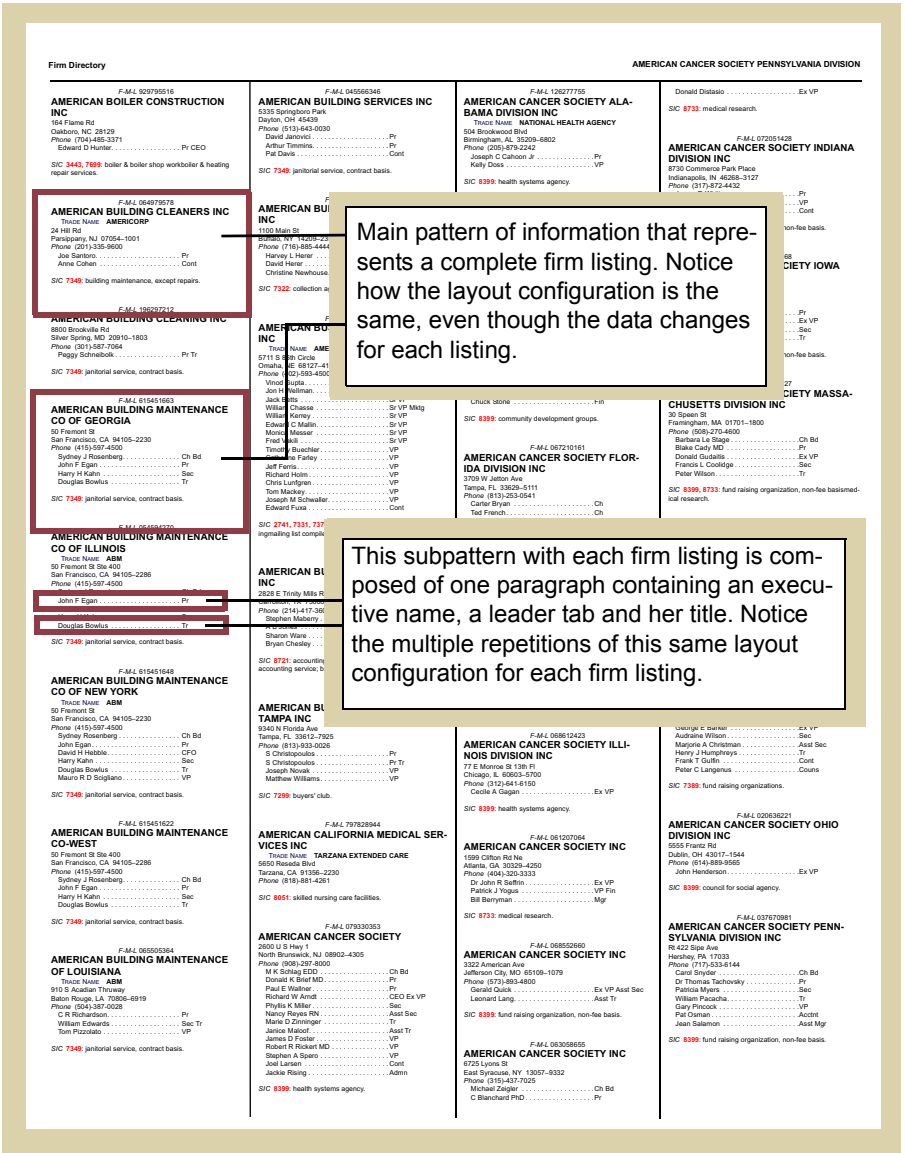


Figure 66: Examples of patterns and subpattern in a simple directory.

1 The attached query object initiates its execution phase. This will mean different things depending on what type of query object it is.

- For a select query, this would entail sending an SQL select statement to the appropriate DBMS.
- For a flat file query, this would entail opening an ASCII text file.
- For a static query, which is not connected to any external data source, its internal counter gets reset to 1.

2 The pattern then cycles for as many times as indicated by the result of the query object being executed. Again, this means different things depending on what type of query object is attached.

- For a select query, the number of cycles is equal to the number of rows returned by the SQL select statement.
- For a flat file query, it would be the number of records in the text file.
- For a static query, it is the number of cycles indicated by the query.

3 The actual action a pattern performs during each cycle is to invoke the target objects that are attached to it. There are actually four named targets lists associated with each pattern. During the first execution cycle the following sequence takes place:

3a All of the targets in the FIRST target list are invoked in the order they appear in the target list.

3b All of the targets in the ALWAYS target list are invoked, again in the order they appear in the target list.

4 For all subsequent execution cycles:

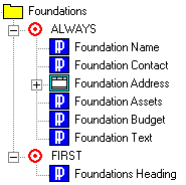
4a All of the targets in the BETWEEN target list are invoked.

4b All of the targets in the ALWAYS target list are invoked.

5 Finally, after the pattern has completed the last cycle, all of the targets in the LAST target list are invoked.

Table 40 contains examples of some subpatterns, comparing the pattern structure with the generated output.

Table 40: Examples of patterns and how they work.

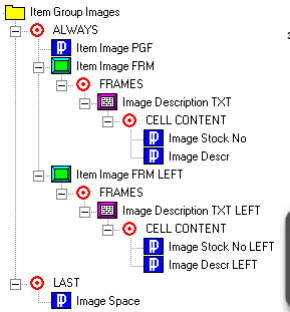
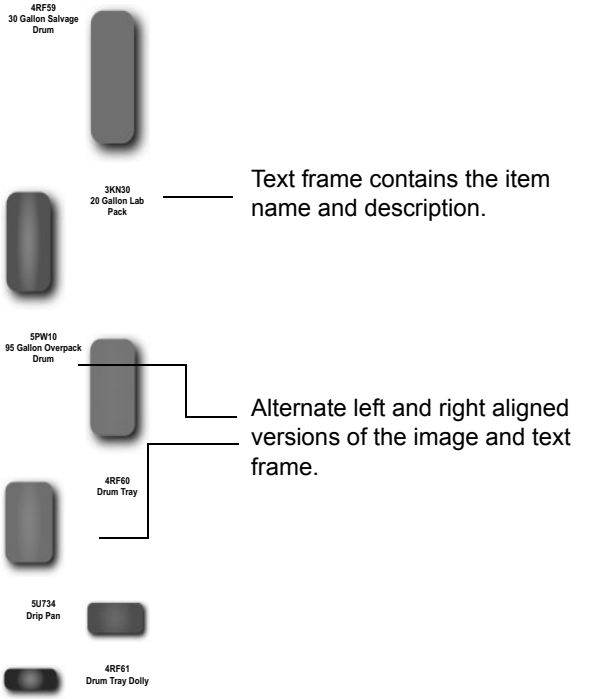
Pattern	Generated Output
	**Corporate Foundations and Giving Programs** First Chicago Corp. Foundation Contact: Diane M. Smith One First National Plaza Chicago, IL 60670 Tel: (312) 732 - 6948 Fax: (312) 732 - 2437 **Assets:** $10,210,998 (1996) **Annual Grant Total:** $750,000 - $1,000,000 *Preference: Chicago area organizations. Primary interests: Ìsocial services and welfare, economic and community development, Ìeducation, and the arts. Recent recipients: University of ÌChicago, Harvard University, United Way/Crusade of Mercy.* NBD Bank Charitable Trust Contact: Laura J. Trudeau 611 Woodward Ave. Detroit, MI 48226 Tel: (313) 225 - 3735 Fax: (311) 225 - 2109 **Assets:** $3,937,470 (1992) **Annual Grant Total:** over $5,000,000 *Preference: giving limited to primary business markets: ÌMichigan, Indiana and Illinois. Grants in 1994 totalled Ì$6,866,862 in the state of Michigan.*

The Foundations pattern will cycle for each result returned by the attached query. In this case, the query returns a record for each foundation belonging to a particular parent company. During each cycle, the following structural elements (and text content) is generated:

- A paragraph containing the foundation name
- A paragraph containing the contact name.
- A table that will hold address and phone information.
- A paragraph containing asset information.
- A paragraph containing the total amount given for the current year.
- A paragraph that contains a text description of the foundation.

Notice that at the beginning of the first cycle the heading for the foundation section is generated.

Table 40: Examples of patterns and how they work. (continued)

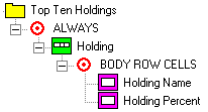
Pattern	Generated Output
	

The Item Group Images pattern will cycle for each result returned by the attached query. In this case, the query returns a result for each image belonging to a particular item group. During each cycle, the following structural elements (and text content) is generated:

- A paragraph that will contain the anchored frame.
- A anchored frame that will hold the imported image.
- A text frame that will contain the image caption.
- A paragraph containing an item name.
- A paragraph containing an item description.

Note that there is a right aligned and a left aligned version of the anchored frame and all of its subcomponents. These target sets are generated on alternate cycles.

Table 40: Examples of patterns and how they work. (continued)

Pattern	Generated Output
	Microsoft Corp 12.3 % Oracle 5.4 % Pfizer 4.3 % Compuware Corp 4.1 % American Home Products 3.9 % Rite Aid 3.8 % Sprint Corp 3.2 % Cisco Systems 3.1 % Philip Morris 2.8 % Mobil Corp 2.1 %

The Top Ten Holdings pattern will cycle for each result returned by the attached query. In this case, the query returns a result for each of a portfolio's top ten holdings, in decreasing order. During each cycle, the following structural elements (and text content) is generated:

- A table row.
- A table cell that will contain the name of the stock.
- A table cell that will contain the percent of the total equity for the current portfolio.

Note: The cell targets are used to control the format characteristics of each table cell such as the shading, rulings, etc.

Understanding the Pattern Builder tab

You use the **Pattern Builder** tab (Figure 67) to create and define both patterns and their attached targets.:

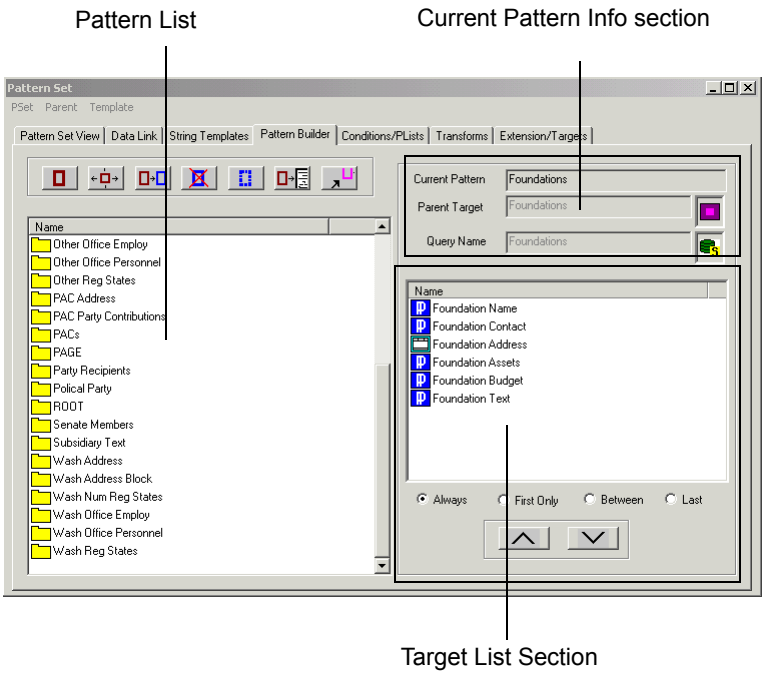


Figure 67: Pattern Builder tab

Pattern List

This window contains all of the existing patterns in the current pattern set, listed in alphabetical order. Select any pattern in the list to make it the current pattern.

Note: You can also make a given pattern the current pattern by selecting it in the pset hierarchy display on the Pattern View Tab of the main dialog, clicking the right mouse button and choosing properties from the popup menus, or double clicking on the pattern icon.

Current Pattern Info Section

Displays information about the current pattern—its name, parent target (the target to which it is attached in the pattern set hierarchy), and its attached query.

Target List Section

The target list window that displays the targets in each of the current pattern's standard named target lists. A specific target list is displayed by clicking on the appropriate radio button below the target window.

Parent Menu and Button Controls

The parent menu provides access to the various operations that can be performed on the current pattern. The buttons along the top of the dialog provide easy access to the functions in the parent menu (See Table 8 on page 38)

Creating a pattern

This section describes how to create and define a pattern and how to attach the pattern to its parent target.

To create a new pattern, follow these steps:

1 From the **Pattern Builder** tab (Figure 67), select the parent menu, then choose the create option. The Create New Pattern Object dialog box is displayed (Figure 68).

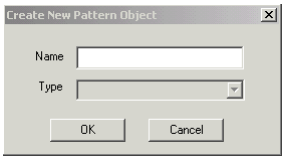


Figure 68: Create New Pattern Object dialog box

2 Type the pattern's name in the Name field and select **OK**. The new pattern now becomes the current pattern.

3 The Pattern Properties dialog box is then displayed (Figure 69).

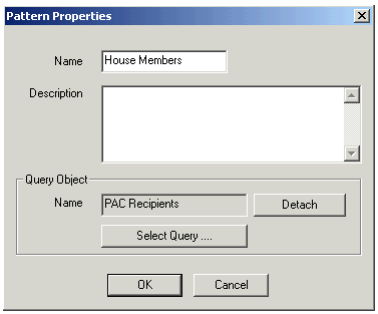


Figure 69: Pattern Properties dialog box

4 To attach a query to the pattern, click on the **Select Query** button. This brings up the Select Query dialog.

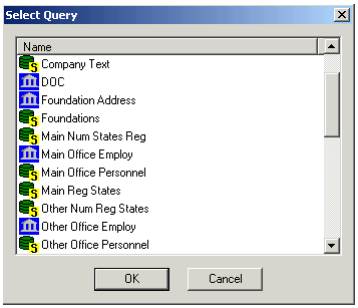


Figure 70: Select Query dialog box

5 Select a query from the list and click **OK**.

6 To detach the query from the pattern, click the **Detach** button. A particular query object can be used more then once, but it can't have the same query nested with itself.

7 Optionally, you can type a description of the pattern in the Description box.

8 Click **OK** to close the Pattern Info dialog and incorporate the changes into the current pattern set.

You can now attach the pattern to a target. Attaching a pattern to a target defines the pattern's location in the pattern set hierarchy.

Inserting a Pattern into the PSet Hierarchy

Creating patterns and targets is a recursive process: targets are created in patterns, and all patterns you create must be attached a targets. Patterns are attached and removed from the PSet Hierarchy through the Subpattern Tab of the target's properties dialog. You can access the properties dialog of a target object in the following ways:

- Finding the target in the Pattern Set View tab of the main dialog and double clicking on the target's icon.
- Finding the target in the Target List tab of its parent target and double clicking on the target's icon.
- Using the Extensions/Targets tab of the main dialog to find the target listed in alphabetical order. Again, to bring up the target's properties dialog double click on the target icon.

Note: You should never attach the current pattern to a parent target through the Pattern Builder tab. This will cause a circular reference that will make the pattern unusable until the attachment is broken.

Once you have the desired target opened, select the SubPatterns tab.

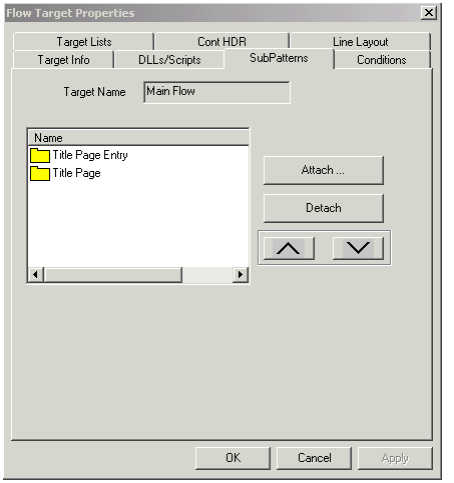


Figure 71: Subpattern tab of the target properties dialog.

Attaching a Pattern to a Target

Attaching a pattern to a parent target is done through the Subpattern tab of the parent target's properties dialog.

1 To attach the pattern, click on the Attach button. This brings up the **Select Pattern** dialog.

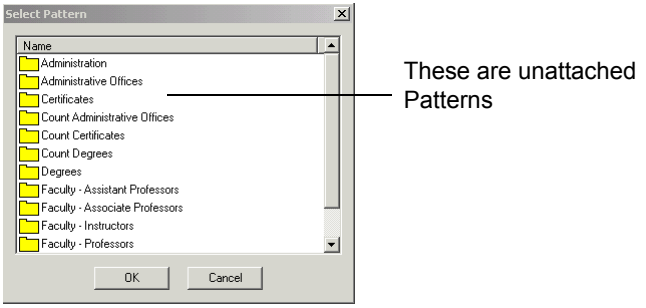


Figure 72: Select Pattern dialog box

Note: Only unattached attached patterns appear in the list of patterns.

1 Select the desired pattern. In this case, it would be the pattern just worked on.
2 Click **OK**.

Detaching a Pattern from a Target

1 First bring up the properties dialog of the target and select the Subpattern tab.
2 Select the subpattern you wish to remove.
3 Click the **Detach** button.
4 Do the same for each subpattern you wish to detach.

Assigning a Pattern to a PSetTree Object

There is actually another way to use pattern object. Instead of inserting them directly into the PSet Hierarchy, you can make a pattern the root of a separate subtree. This is done by attaching an unattached pattern to an object call a PSetTree. PSetTree objects are members of the extension class of PatternStream object. When a pattern is attached to one of these objects, it can be used multiple times in the PSet Hierarchy through the call target mechanism (see “Call target” on page 245). To attach a pattern to a PSetTree:

1 Select the Extensions/Targets tab of the main dialog.

2 Select the desired PSetTree object, click the right mouse button, and then select properties from the popup menu, or double click on the PSetTree icon.

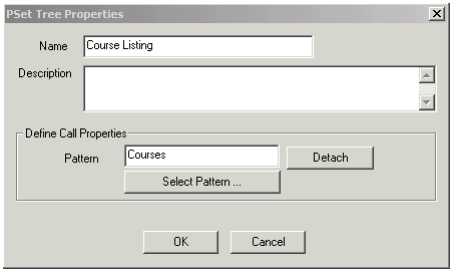


Figure 73: Attaching a Pattern to a PSetTree.

3 Click on the **Attach Pattern** button, and then select the desired unattached pattern.

4 Click OK to incorporate the changes.

5 To detach an already attached pattern, bring up the PSetTree dialog and click the **Detach** button.

Note: Like before, only unattached patterns will appear in the Select Pattern dialog. Patterns already associated with a PSetTree object are considered attached and will not appear in the UNATTACHED list.

Making a Pattern the Root Pattern

Any unattached pattern can be made the root pattern. To assign a pattern to the role of the root pattern:

1 Select the PatternView tab of the main dialog. In the upper left hand corner of the pattern view display is the icon for the pset, itself.

This icon represents the entire pset, and is used to access its global properties.

2 Click on the **Select Pattern** button. This brings up the Select Pattern dialog.

3 Select the unattached pattern from the displayed list that you wish to make the root pattern.

4 Click **OK.** The pattern set view now shows the selected pattern as being the root pattern.

Copying a pattern

If you need to create several similar patterns, you can create one, copy it, then modify the new pattern's properties as needed.

To copy a pattern, follow these steps:

1 From the pattern list box section of **Pattern Builder** tab, select the name or icon of the pattern you want to copy, then select the **Copy Pattern** button. The Copy Pattern dialog box is displayed (Figure 74).

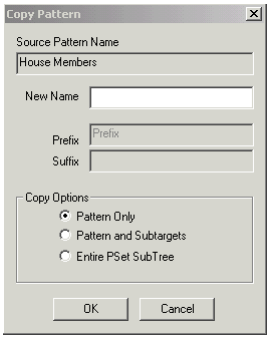


Figure 74: Copy Pattern dialog box

2 In the Prefix and Suffix fields, type a prefix or suffix (or both) you want attached to the copied pattern (and, if applicable, to its copied subtargets and subpatterns).

3 Select one of the three copy options:

- Pattern Only
- Pattern an immediate subtargets.
- Pattern and its entire subtree.

4 If Pattern Only was selected, then the dialog prompts for the new pattern's name, otherwise, you have the option of declaring a prefix and/or a suffix that will be used to build the names of all of the subtargets and subpatterns that will be copied.

5 Select **OK**. The new pattern is displayed in the pattern list box of the **Pattern Builder** tab and becomes the current pattern.

6 Define the properties of the new pattern (and, if applicable, its subtargets and subpatterns):
- Confirm that the attached query is appropriate for the copied pattern.
- Attach the new pattern to a parent target.
- Confirm the properties of the subtargets and subpatterns contained within the new pattern.

Deleting a pattern

You can delete only unattached patterns. Before you can delete a pattern, you must detach it from its parent target.

To delete a pattern, follow these steps:

1 Detach the pattern from its parent target. Refer to “Detaching a Pattern from a Target” on page 123 for details.

2 From the pattern list box section of the **Pattern Builder** tab, select the name or icon of the pattern you want to delete, then select the **Delete Pattern** button. The Delete Pattern Set Hierarchy Object dialog box is displayed.

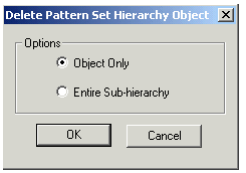


Figure 75: Delete Pattern Set Hierarchy Object dialog box

3 Select whether you want to delete the pattern only or its entire subhierarchy as well. If you delete the pattern only, the targets created in the pattern are not deleted; instead, they are relocated to the Unattached Targets section of the **Pattern Set View** tab. If you select the entire sub-hierarchy, then all of the sub-targets and subpatterns are recursively deleted.

4 Click **OK**.

Chapter 8: Target basics

What is a target?

A target is a type of PatternStream object that creates the output document's structural elements, such as a table, a paragraph, or an anchored frame. For example, a Paragraph target generates a new paragraph as if a user presses the **Enter** key to create a new paragraph and then applies a particular paragraph tag. Likewise, a Table target generates a new table at the current insertion point every time it is invoked as if a user manually selects **Insert Table** from the FrameMaker Table menu and then selects a particular table format.

A target usually doesn't supply the content for the output document; it just creates the structural element that will contain text or graphics. However, you can associate certain targets with *string templates,* another class of PatternStream object. String templates specify how to build text elements from the data extracted from the database. By grouping data values and constant strings with other text items (cross-references, index markers, and anchored graphic insets), these string templates create the content. Remember, targets generate the structural elements that can act as containers for text; string templates create the text content.

You insert targets into named target lists, which determine when or how often targets in the target list are invoked. These differences are discussed in "Kinds of targets" on page 128 and "Named Target Lists" on page 131.

The following rules apply to all targets:

- Unlike other objects, such as conditions and string templates, targets can only be used once in the PSet hierarchy. If you wish to use a target more than once, you must make a copy of the target and use a different name.
- Targets that are used in the PSet Hierarchy will always be associated with a parent object (either another target or a pattern) by being attached to a named target list.
- The PatternStream application invokes targets by their order in the named target list.
- Targets that are not attached to a named target list appear in the Unattached Targets list at the bottom of the **Pattern Set View** tab and also on the **Extensions/Targets** tab of the main dialog.
- Only targets in the unattached list can be deleted unless they are part of an entire sub-hierarchy that is being deleted.

Kinds of targets

Targets can be grouped by their function in the PSet hierarchy.

Structural targets

These targets provide the structural components of the document being created. Many commonly used targets belong to this category. There are two types of structural targets:

- *Text targets*
  Besides creating a structural element, text targets provide content for PatternStream-generated FrameMaker documents through their associated string templates. Paragraph targets are the most common example of a text target. String templates associated with text targets define how to build the text that goes into the particular element being created. String templates are described in detail in Chapter 15, "String templates."
- *Non-text targets*
  These targets generate FrameMaker elements that provide structure where insertion of text would be ambiguous or meaningless. For example, Document, Page, and Anchored Frame targets are associated with document, page, and anchored frame elements. To ultimately generate text connect within a non-text target, you must attach a text target to the appropriate named target list

associated with the non-text target

.

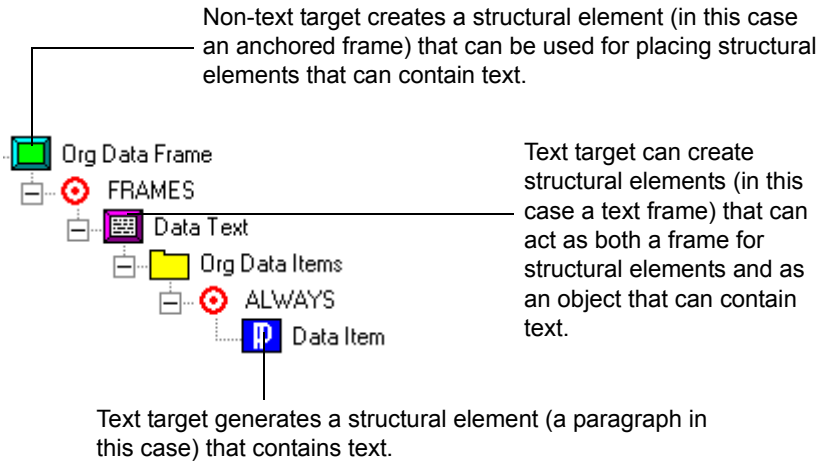


Figure 76: Relationship between text targets and non-text targets

Non-structural targets

The targets in this group do not directly relate to the FrameMaker formatting engine or any of the structural elements being created.

- *Data manipulation*
  These targets retrieve and manipulate data. For example, Lookup Table targets add entries to an associative array that can be used later in the document generation process.
- *Execution control*
  These targets provide constructs that work somewhat like the BEGIN/END, CASE and IF/ELSE statements in a standard programming language, such as C or Visual Basic. For example, the IF/ELSE target works by invoking the targets in the IF target list when an attached condition is TRUE, and invokes the targets in the ELSE target list when that same condition evaluates to FALSE.

Table 41 categorizes the structural and non-structural targets.

Table 41: Structural vs. non-structural targets

Structural		Non-structural	
Text	**Non-text**	**Data manipulation**	**Execution control**
Paragraph	Book	Empty	Blank
Table	Document	Assignment	Target Set
TextFrame	Page	Lookup Table	If/Else
Table Cell	Row		Case
VarCell	Anchored Frame		Call
Log	Unanchored Frame		PSet
Marker			
Variable Format			

In addition to being a structural or non-structural, targets are either simple or compound:

- *Simple targets*
  Simple targets only generate themselves and do not have associated named target lists. An example of a simple target is a Paragraph target, which generates a simple paragraph.
- *Compound targets*

Compound targets use named targets lists to generate complex structural elements. Although the parent and child targets typically create separate structural elements, complex targets are used to create a complex ensemble of these elements in unison during each invocation. For example, a table cell target can have a named target list called CELL CONTENTS. This target list can contain one or more paragraph targets (or any valid flow insertion target). When the table cell target is invoked the paragraph targets in the target list are invoked as well, creating the additional paragraphs in the current table cell. A target can be both a compound target and a text target. The attached string templates create text in

the first default paragraph of the current object. The sub-targets then create additional structural elements in the current object.

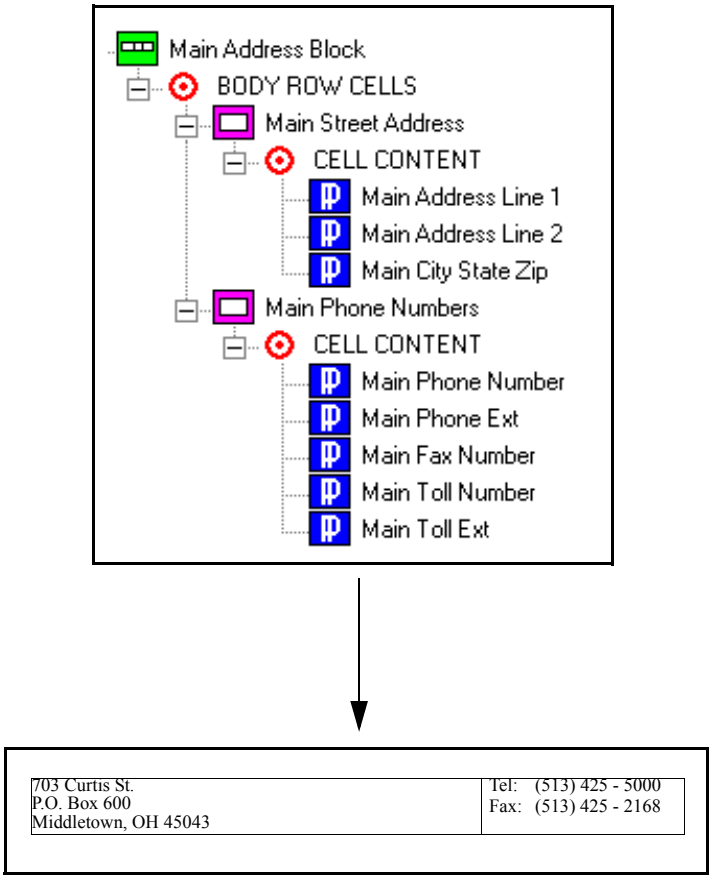


Figure 77: Text targets and compound targets.

Named Target Lists

You insert a target object into the PSet Hierarchy by attaching it to a named target list. This target list determines where or when the targets in the list will generate structural elements. For example, the target lists associated with a pattern object determine when the targets in the list are invoked:

- *Always*
  Targets in this list are generated during every loop of the pattern's execution

cycle.

- *First Only*
  Targets in this list are generated only during the first loop of the pattern's execution cycle, before the Always list is generated.
- *Between*
  Targets in this list are generated during every loop of the pattern's execution cycle except the first loop.
- *Last*
  Targets in this list are generated only after the completion of the last loop of the pattern's execution cycle.

Other target lists determine the logical position of the elements being created by the targets in the list. The Cell Contents list of the Table Cell target mentioned

above is a good example.When each cell target is invoked, it will create a series of paragraphs within the current cell.

.

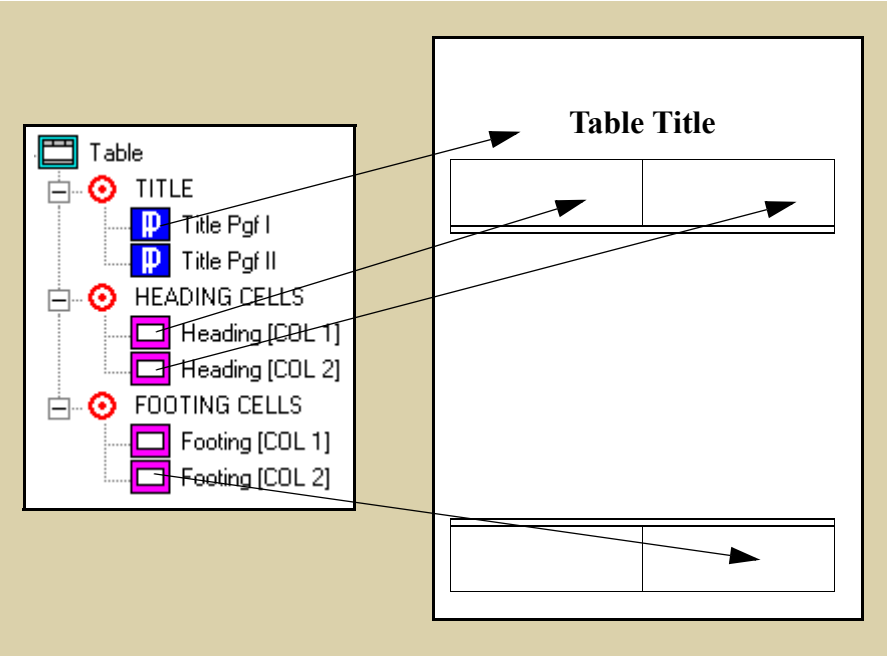


Figure 78: This illustrates the relationship between the three targets lists associated with Table targets and targets contained in them. The Heading and Footing target lists contain Table Cell targets that allow manipulation of the heading and footing cells of each table element generated by the Table target. The structure elements created by targets in the Title target list appear in the title of each table generated.

Targets not attached to a specific target named target list (i.e.,. do not have a parent) are referred to as “unattached” and reside in the UNATTACHED target list. Some important things to note about targets lists are:

- Targets can only reside in one target list at a time (this is a corollary of the rule that targets can only be used once in the PSet Hierarchy).
- You move a target around in the PSet Hierarchy by first removing it from one named target list and then inserting it into another.
- Targets in a given named target list are invoked in the order they appear in the list.

Available targets

This chapter divides targets by the function they perform in the PSet hierarchy. Targets that generate high-level output elements, such as documents and pages, are listed in Table 2 on page 136. Targets that create and manipulate a table structure are described in Table 4 on page 138.

If you don't know the function of a particular target, check the alphabetical list in Table 1:

Table 1: Available targets

Target	Icon	For details, see…
Anchored Frame		Table 3 on page 137
Assignment		Table 5 on page 138
Blank		Table 3 on page 137
Book		Table 2 on page 136
Call		Table 7 on page 140
Case		Table 7 on page 140
Document		Table 2 on page 136
Empty		Table 5 on page 138
Flow		Table 3 on page 137
If-Else		Table 7 on page 140
Log		Table 6 on page 139

Table 1: Available targets (continued)

Target	Icon	For details, see…
Lookup		Table 5 on page 138
Marker		Table 3 on page 137
Page		Table 2 on page 136
Paragraph		Table 3 on page 137
Pattern Set		Table 7 on page 140
Row		Table 4 on page 138
Table		Table 4 on page 138
Target Set		Table 7 on page 140
Table Cell		Table 4 on page 138
Textframe		Table 6 on page 139
Unanchored Frame		Table 6 on page 139
Variable Cell		Table 4 on page 138
Variable Format		Table 6 on page 139

Global targets

A global target generates the high level structural elements, such as a FrameMaker file, the body pages within a file, and a book file. Table 2 describes the targets used to create a global element.

Table 2: Global targets

Target name	Description	For details, see...
Document 	Generates a FrameMaker document from the FrameMaker template you specify.	"Document targets" on page 162
Page 	Generates a series of connected body pages according to the master pages in the template you specify.	"Page targets" on page 168
Book 	Generates a book file that will contain subsequently generated documents.	"Book Targets" on page 177
Variable Format 	Modifies the definition of user-defined FrameMaker variable formats.	"Variable Format Target" on page 182
Log 	Outputs data to an ASCII file.	"Log target" on page 182
Marker 	When used in non-create mode, flows text at the location of the marker with a given type and marker text.	"Marker targets" on page 180

Flow insertion targets

When you press the return key in a FrameMaker document, create markers or anchored frames, you are inserting structural element into a flow. Table 3 describes targets that create elements inserted into a flow.

Table 3: Flow insertion targets

Target name	Description	For details, see...
Paragraph 	Generates a paragraph in the format of the tag associated with the target in your template.	"Paragraph targets" on page 186
Flow 	Provides a specialized handle for flow insertion targets in the PSet hierarchy. Also used to implement a generalized continuation header for related groups of paragraphs.	"Flow targets" on page 191
Marker 	Inserts a marker of a specified type.	"Marker targets" on page 180
Anchored Frame 	Creates an anchored frame to which you can attach unanchored frames, text frames, and imported graphics.	"Anchored frame targets" on page 223
Table 	Generates a table according to the assigned table format.	"Table targets" on page 198

Table targets

You create tables using a target for each component—the entire table, the rows, and the cells. Table 4 lists the targets available for generating tables.

Table 4: Table targets

Target name	Description	For details, see...
Table 	Generates a table according to the associated table tag.	"Table targets" on page 198
Row 	Generates a one or more table rows when invoked.	"Row targets" on page 202
Table Cell 	Operates on the table cells of the associated row or table heading (footing). String templates associated with cells can be used to specify the content of the cell.	"Cell targets" on page 206
Variable Cell 	Used to generate tables with data driven structure. Unlike Table Cell targets, Variable Cells refer to the column they are associated with by name rather than by number.	"Cell targets" on page 206

Data manipulation targets

Some targets are not associated with a structural element, such as a table or anchored frame. Instead, they modify or manipulate the data retrieved from the database. Table 5 describes targets that manipulate data.

Table 5: Data manipulation targets

Target name	Description	For details, see...
Assignment 	Takes the current value of a source variable and applies it to a target variable (X := Y).	"Assignment target" on page 247
Empty 	Acquires data without generating content.	"Empty target" on page 249

Table 5: Data manipulation targets (continued)

Target name	Description	For details, see...
Lookup 	Lets you store data in an associative or indexed array.	"Lookup target" on page 250

Frame targets

Some targets are used infrequently, but they are handy for specific situations. Table 6 lists these targets, which are described in Chapter 12, "Frame targets."

Table 6: Frame targets

Target name	Description	For details, see...
Textframe 	Creates a text frame inside an anchored or unanchored frame.	"Text Frame Target" on page 231
Unanchored Frame 	Generates an unanchored frame in three different modes: 1. Identical frames at a given x, y location on every body page in a sequence of connected body pages. 2. Single frame in a given x, y location on the current page. 3. Single frame in a location determined by a page grid algorithm.	"Unanchored Frame target" on page 214

Execution control targets

Execution control targets let you manage the flow of execution as the pattern set is running. These targets are similar to the structured programming conventions

of standard programming languages such as C and FORTRAN. Table 7 describes these targets.

Table 7: Execution control targets

Target name	Description	For details, see...
Blank 	Provides a placeholder for attaching a subpattern directly below another pattern without any intermediate output.	"Blank target" on page 240
If-Else 	Provides an IF-ELSE control structure in the form of two target lists, IF and ELSE.	"If-Else target" on page 240
Case 	Lets you attach a string variable to a target list so that the target list is executed when the value of the variable string is the same as the name of one of its target lists.	"Case target" on page 241
Pattern Set 	Allows you to run another pattern set from the current one. You can pass parameter values to this pattern set from the calling pattern set.	"Pattern Set target" on page 242
Call 	Part of the mechanism for placing a single pattern multiple places in the pattern set hierarchy.	"Call target" on page 245
Target Set 	Provides a way to organize target lists in a single target. A condition attached to the Target Set target applies to all target lists in the SET named target list.	"Target Set target" on page 240

Creating New Targets

New targets are created from the Create New Target dialog (Figure 79). This dialog can be accessed:

- From the Pattern Builder Tab of the Main Dialog, right click with the mouse in the white area of the target list window on the right-hand side of the dialog and select create target from the popup menu.
- From the Target List Tab of a compound target's properties dialog, right click with the mouse in the white area of the target list window and select create target from the popup menu.

- From the Extensions/Targets Tab of the Main Dialog, right click with the mouse in the white area of the target list window on the right-hand side of the dialog and select new object from the popup menu.

Note: It makes no difference whether the attached or unattached radio button is selected. All targets created using the target window on the Extensions/Targets Tab will be "unattached".

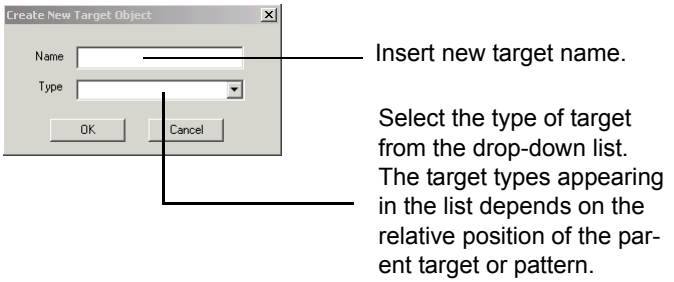


Figure 79: Create New Target Object dialog box

When the Create New Target dialog displays:

1 Type the name of the new target. As with all PatternStream classes, you cannot have two targets with the same name.

2 Select the type of target from the drop-down list. The types displayed in the list depend on the location of the new target in the PSet Hierarchy. This location will be in the Named Target List that was displayed in the target window when the create target command was invoked (i.e., the right mouse button was clicked).

Note: New targets destined for the UNATTACHED target list will have no type restrictions. (i.e.,. all target types will appear in the drop-down list).

3 Select OK. The target properties dialog for the type of target chosen will be displayed.

Managing Existing Targets

Managing targets is done from a dialog that contains a target window either through the commands on the targets popup menu or the target movement controls below the window. By selecting a target, then clicking the right mouse button, the target menu displays with the following options:

- *Properties*
  Opens the properties dialog of the selected target object.
- *References*
  Displays the parent target or pattern object for the selected target.
- *Copy*
  Displays the Copy Target dialog.
- *Remove*
  Removes the selected target from the current Named Target List.
- *Export*
  Exports the selected target to the object paste buffer.
- *Create Target*
  Displays the Create New Targets dialog.
- *Print List*
  Creates a printable FrameMaker file displaying the attributes of all of the target objects in the current Named Target List.
- *Print Attributes*
  Creates a printable FrameMaker file displaying the attributes of the selected target object.

Working with Targets

The attributes of any target object are modified through its target properties dialog, which can be accessed in any of the following ways:

- Pattern Builder tab of the Main Dialog.
- Through the TargetLists tab of a compound target's properties dialog.
- Using the target window on the Extensions/Targets tab of the Main Dialog.
- The PSet Hierarch displayed on the PatternView tab of the Main Dialog.

Using the Pattern Builder Tab

In the PatternStream application, you can create targets using the lower right half of the **Pattern Builder** tab (Figure 80).

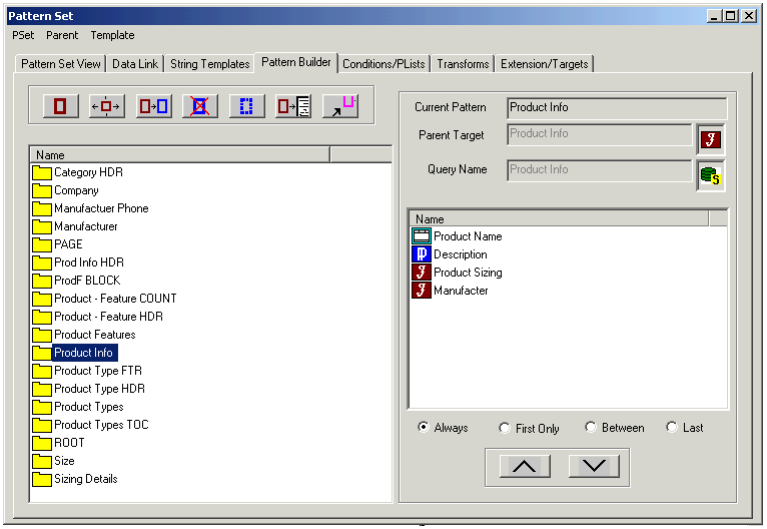


Figure 80: Pattern Builder tab

Before you can work with a target within the context of a pattern object, you must first select the pattern that either contains or will contain the desired target. You select a pattern to be the current pattern by either

- Selecting the desired pattern from the alphabetized list of pattern objects on the left side of the Pattern Builder Tab.
- Double-clicking on the desired pattern from the PSet Hierarchy in the PatternView Tab. In this case, the Main Dialog automatically changes to the Pattern Builder Tab.

Use the radio buttons in the lower right of the dialog to select which target list to work with. The Named Targets Lists associated with a pattern object are shown in the right hand side window of the Pattern Builder tab.

Note: You can create targets in any pattern, whether the pattern is attached or unattached. When you create a target in a pattern, the PatternStream application automatically restricts your target options to those appropriate for the pattern's location in the PSet hierarchy. This prevents you from making errors in the structure of the document you are creating. If you create a target in an unattached pattern, you're more likely to insert an invalid target, because your target options are not restricted.

Using the Target List Tab

Another way of accessing the commands in the target popup menu is through the Target List Tab of a compound target's properties dialog. A target attached directly to another target is called a sub-target and must be in one of the existing Named Target Lists of the parent target. Assuming the properties dialog of the parent target is already opened and the Target Lists Tab has been selected, choose the target list from the target list drop-down. The targets attached to the selected list will be displayed in the target window.

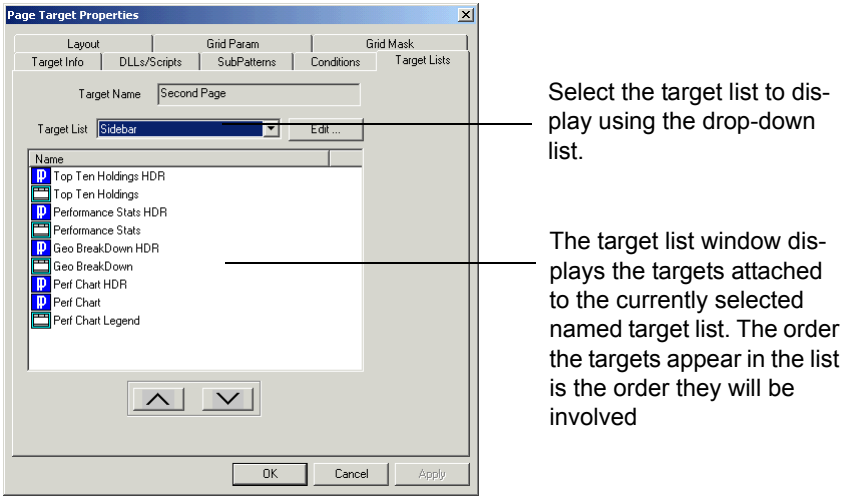


Figure 81: Subtargets in a named target list

Using the Extension/Targets Tab.

You can access the target commands in the target popup menu without having to know the desired targets position in the PSet Hierarchy through the Extension/ Targets Tab of the Main dialog.

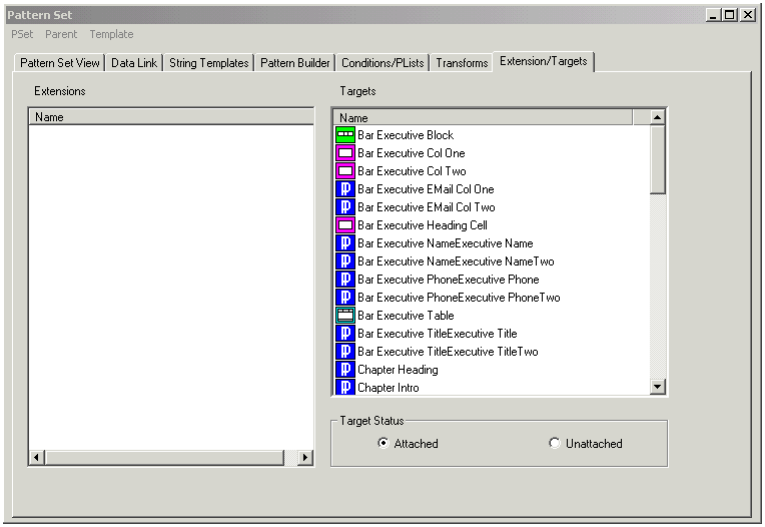


Figure 82: Accessing target properties through the Extensions/Targets tab.

- Clicking on the Attached radio button will display in the target window on the right side of the dialog all of the targets attached to some target list in alphabetical order.
- Clicking on the Unattached button will display all of the unattached targets. This is the same list that appears in the Unattached folder of the Pattern Set View.

Using the Pattern Set View

Like all PatternStream classes, you can access item manipulation commands through the object popup menu of the Pattern Set View Tab of the Main dialog.

1. Find the desired target by scrolling through the PSet Hierarchy, possibly expanding collapsed sub-trees as you go.
2. Select the target, then click on the right mouse button. This displays the object popup menu.

Modifying Target Properties

You edit the attributes of any target through it's Target Properties dialog. The tabs on this dialog box vary with the type of target you create. However, a minimum of four tabs are always present.:

The Target Info tab

The **Target Info** tab is the first tab on the target properties dialog and is used to input a name an a description.

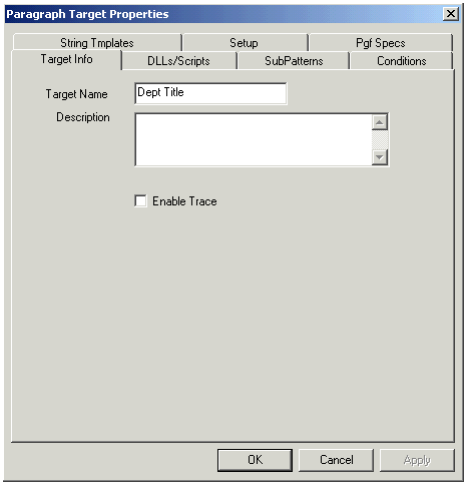


Figure 83: Target Properties dialog box

Table 8: Index heading attributes

Attribute Name	Description
Target Name	You can change the name of the target by typing in a new name here. *Note:* When changing the name of a target, you should click on the **Apply** button before proceeding to the other tabs.
Description	You can provide a brief description of the target in the Description field. This description shows up when printing target properties and is for documentation purposes only.

Table 8: Index heading attributes

Attribute Name	Description
Trace	When this option is checked, a message will be sent to the FrameMaker console whenever the target generates output. (i.e., it is invoked and its attached conditions are all TRUE).

The DLLs/Scripts tab

The **DLLs/Script** tab lets you call an external function compiled in a separate DLL or execute a FrameScript script each time the target is invoked. These external DLLs must be written as FDK clients in order to operate on the current FrameMaker document. Along with other parameters, the object ID of the structure element, which is created each time the target is invoked, is passed to the external function. For more information on customizing the PatternStream application using external DLLs, see Chapter 19, “Creating extensions.”

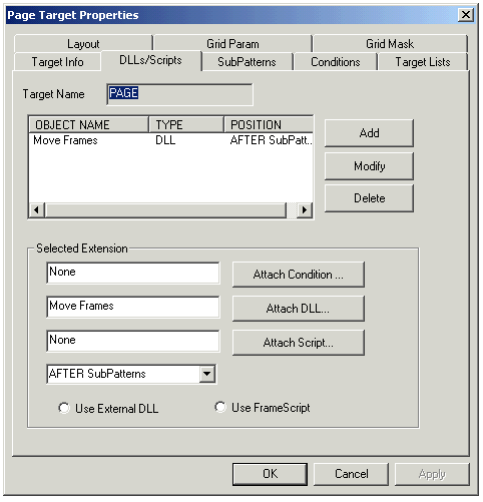


Figure 84: DLLs/Scripts tab

The Subpatterns tab

.The **SubPatterns** tab displays the names of subpatterns attached to the current

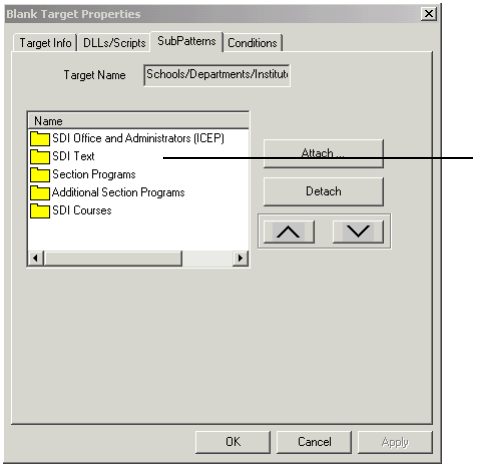


Figure 85: SubPatterns tab

target. The subpatterns that appear in the list will be executed in the order they appear in this list.

Table 9: Attaching and Detaching Subpatterns

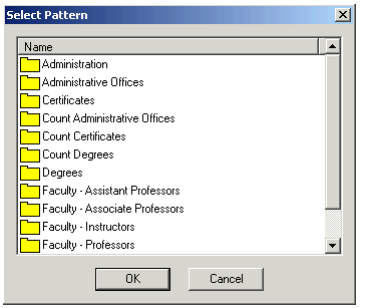
Operation	Procedure
Attaching a sub-pattern	**1** Click on the **Attach** button. The Select Pattern dialog is displayed.  **2** Select the pattern you wish to attach. **3** Click the **OK** button to attach the pattern.

Table 9: Attaching and Detaching Subpatterns

Operation	Procedure
Detaching a sub-pattern	**1** Selecting the sub-pattern you wish to detach from the list of subpatterns for the current target. **2** Click on the **Detach** button. The sub-pattern is removed from the list. **3** Click on the Apply button to incorporate the change. This moves the selected pattern to the unattached list. *Note:* If you haven't clicked on the Apply button, the change still hasn't been incorporated into the current pattern set and the "detached" sub-pattern will not appear on the unattached list.
Changing the order of execution	**1** Selecting the sub-pattern you wish to move from the list of subpatterns for the current target. **2** Click on the up or down arrow buttons to move the select sub-pattern forward or backwards in the sub-pattern list.

The Conditions tab

The **Conditions** tab (Figure 86 on page 150) is the final tab common to all targets. A condition is an object that at any given time evaluates to either *true* or *false*. If a condition equals *false*, the target it is attached to does not generate any output when invoked. If a target has more than one condition attached, all of the

conditions must be *true* in order for the target to generate. Targets with no conditions always generate when invoked.

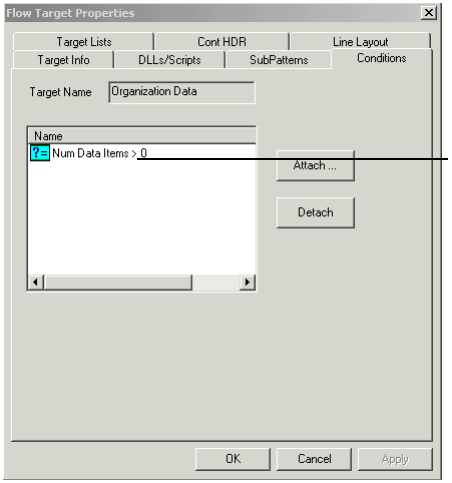
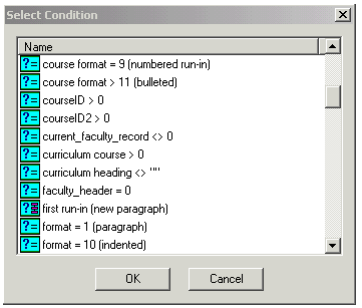


Figure 86: Conditions tab

Note: When the attached condition evaluates to False and the structural element is not generated, the entire sub-tree below this target, including all of its sub-targets are also skipped.

Table 10: Attaching and Detaching Conditions

Operation	Procedure
Attaching a condition	**1** Click on the **Attach** button. The Select Condition dialog is displayed.  **2** Select the condition you wish to attach. **3** Click the **OK** button to attach the condition.
Detaching a condition	**1** Selecting the condition you wish to detach from the list of conditions for the current target. **2** Click on the **Detach** button. The condition is removed from the list. **3** Click on the Apply button to incorporate the change or click OK to close the target properties dialog and incorporate the

Other tabs in the Target Properties dialog box vary with the target; however, certain tabs are always displayed for some of the main target groups:

The String Templates tab

All text targets, by definition, have a **String Templates** tab, so you can attach a string template that will be used to create content for the associated target. For details on creating string templates, see Chapter 15, “String templates.”
Figure 87 shows the **String Templates** tab for a for a typical text target (paragraph).. The List displays the string templates currently attached to the selected

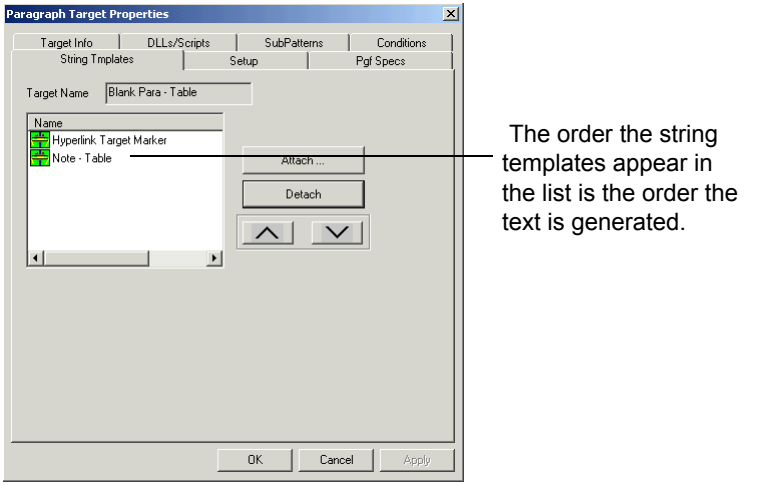


Figure 87: String Template tab

target. For targets with multiple string templates, they will be invoked in the order the they appear in the list.

Table 11: Attaching and Detaching Subpatterns

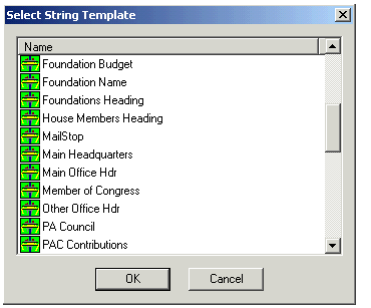
Operation	Procedure
Attaching a string template	**1** Click on the **Attach** button. The Select String Template dialog is displayed.  **2** Select the string template you wish to attach. **3** Click the **OK** button to attach the string template.

Table 11: Attaching and Detaching Subpatterns

Operation	Procedure
Detaching a string template	**1** Selecting the string template you wish to detach from the list of string templates for the current target. **2** Click on the **Detach** button. The string template is removed from the list. **3** Click on the Apply button to incorporate the change.
Changing the order of execution	**1** Selecting the string template you wish to move from the list of string templates for the current target. **2** Click on the up or down arrow buttons to move the select string template forward or backwards in the list.

Note: String templates can be used multiple times in the PSet hierarchy.

The Target List tab

All compound targets have a **Target Lists** tab where you can attach sub-targets.

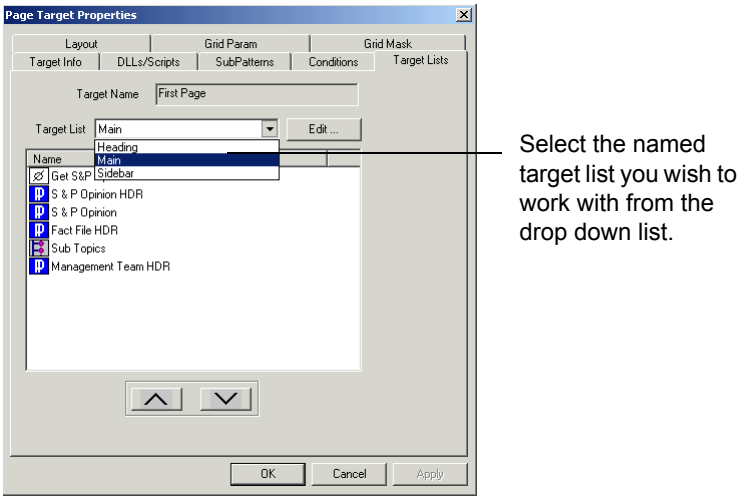
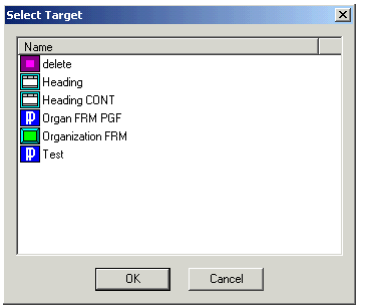


Figure 88: Target Lists tab

The Target List drop-down list displays the named target lists that exist for the current target. When you select a target list, the targets attached to that list appear in the target window.

Table 12: Attaching and Detaching Subpatterns

Operation	Procedure
Inserting a sub-target	**1** Click the right mouse button and select Insert from the target popup menu. This will display the **Select Target** dialog.  **2** Select the target you wish to attach. **3** Click the **OK** button to attach the target. The target now is displayed in the current target list.
Removing a sub-target	**1** Click on the sub-target you wish to remove from the current named target list. **2** The sub-target is immediately removed from the current named target list.
Changing the order of execution	**1** Selecting the sub-target you wish to move from the list of sub-targets for the current target. **2** Click on the up or down arrow buttons to move the selected sub-target forward or backwards in the target list.

For more information on target lists, see “Adding and Organizing Target Lists” on page 156.

Copying targets

You can save time creating targets by copying a target and using it elsewhere in the hierarchy. This saves time because the copied target can include copies of the subpatterns and sub-targets that are attached to the source target. To copy a tar-

get, first select the desired target, then click on the right mouse button. From the target popup menu, select Copy. The Copy Target dialog box is displayed (Figure 89), and the name of the target you selected is displayed in the Source Target Name field.

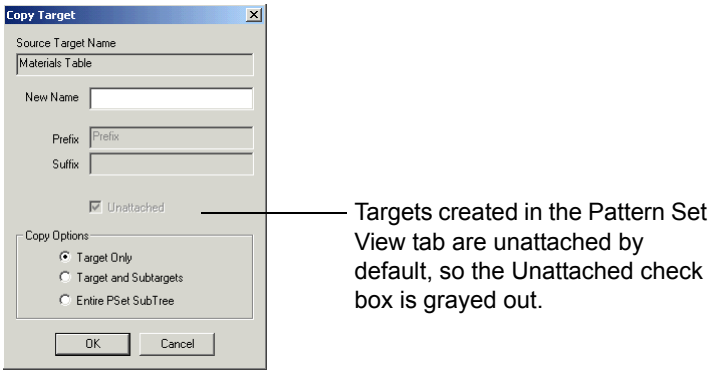


Figure 89: Copying a target

If you plan to attach the target to a target list other than the current one, select the **Unattached** check box. Then you can add the target to the PSet by selecting it from the Unattached Targets list.

3 Check the **Copy Options** check box that applies to the copied target. The options are:

- **Target Object Only**

If the source target is associated with subtargets or subpatterns, but you don't want to copy these attachments, check this option. Type the name of the new target in the New Name field.

- **Targets and Subtargets**

If you want to copy a source target's associated subtargets but not the subpatterns, check this option. To distinguish the newly copied target from the source target, you must include either a prefix and/or suffix. This prefix or suffix is added to the names of all the subtargets created.

- **Entire PSet Subtree.**

If you also want to copy the entire subtree below the target, check this option. As in the previous case, the prefix and/of suffix will be added to names of the copied targets and patterns.

4 Select **OK.** The copied target is displayed in the PSet hierarchy or the Unattached Targets folder (if you created an unattached target).

Note: If you attach targets and subpatterns to an unattached parent target, they're still considered to be attached. In the **Pattern Set View** tab, they display as part of the subtree attached to the unattached parent target.

Removing targets from a pattern

When you remove a target from a pattern or target list, the target is not deleted from the PatternStream application. Instead, the target is moved to the UNATTACHED target list.

To remove a target from a target list, select the target you want to remove, right-click, then select **Remove** from the pop-up menu. The target is no longer displayed in the Target List or in the hierarchy of the **Pattern Set View** tab.

Adding and Organizing Target Lists

In the **Target Lists** tab of the Target Properties, you manage target lists associated with a target. All target lists for the selected parent target are displayed in the drop-down list, and you can add more targets lists to a target.

All compound targets have at least one valid target list. Each target list typically generates elements in different parts of the document hierarchy and will therefore have different valid target lists. Target lists that are not in the valid target list for a particular target are ignored with two exceptions:

- Case targets: Valid target lists correspond to the value of the Case variable, so you can have target lists with user-defined names.
- Page targets: Valid target lists correspond to the named flows on the master page; therefore, for any name not in the list, the PatternStream application attempts to insert structural elements into a named flow that does not exist. This generates an error.

Note: Before you can attach a sub-target to a compound target, you must first add the appropriate target list first.

Adding a target list to a target

To add a target list to a target, follow these steps:

1. Select the compound target to which you want to attach the target list
2. Right-click and select **Properties** from the pop-up menu, or double click on the target icon. The Target Properties dialog box for the selected target is displayed (Figure 83 on page 146).

3 Select the **Target Lists** tab. The name of the current target list is displayed in the Target List drop-down list. Subtargets attached to the target list are displayed in the Name list box (Figure 90).

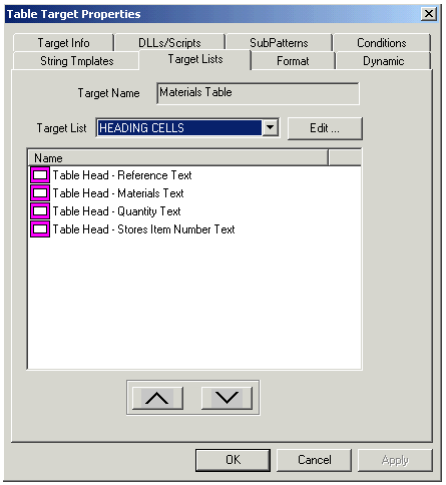


Figure 90: Flow A subtarget in the A target lists

4 Select **Edit** next to the Target List drop-down list. The Edit Target Lists dialog box is displayed (Figure 91).

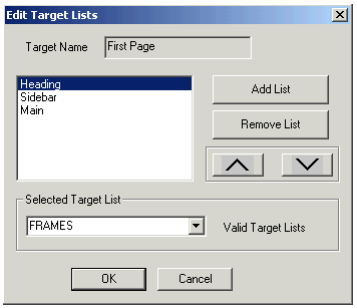


Figure 91: Edit Target Lists dialog box

Select the **Valid Target Lists** drop-down list to select the name of the list you want to add, or type a new name in the Selected Target Lists drop-down list box.

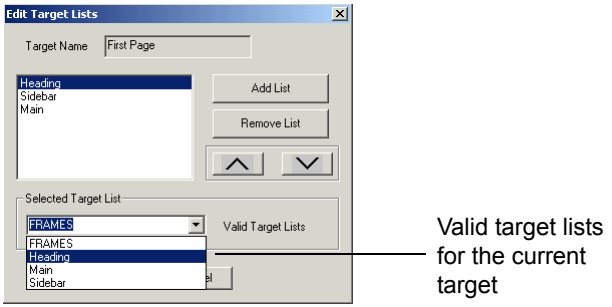


Figure 92: Valid target lists

5 Select **Add List,** then select **OK.**

6 Select **OK** to return to the target properties dialog.

You have now created an empty target list.

Rearranging target lists

The PatternStream application processes target lists in the order they appear in a PSet. In some cases this may be important, so you might need to rearrange target lists after you add them to a target.

To re-order the named target lists of a compound target, follow these steps:

1 Select the **Target Lists** tab of the compound target.

2 Select the **Edit** button. The Edit Target Lists dialog box is displayed (Figure 91 on page 157).

3 To move a target list up or down, select the target list, then select the up or down arrow.

4 Select **OK** to close the Target Properties dialog box.

The target lists will be displayed in the PSet view in the new order.

Removing a target list

You can remove a target list when you no longer need to generate specific content, and the targets that were in the deleted target list are moved to the UNATTACHED target list.

Note: Because the original targets are detached when you delete a target list, you'll need to reinsert the targets if you add the target list back to the parent target.

To remove a target list, follow these steps:

1. Select **Edit.** The Edit Target Lists dialog box is displayed (Figure 91 on page 157).
2. Select the list you want to delete, then select **Remove List.**
3. Select **OK,** then select **OK** again to close the Target Properties.

Chapter 9: Global Targets

Before you can create any content, you must first create a container for that content. The PatternStream application does this with three structural targets—the Document, Page, and Book targets. These targets are fundamental to all PSet files, because they create the high-level structures (books, documents and pages) into which content is generated.

This chapter describes how to configure the Document, Page, and Book targets. Table 13 provides a quick overview.

Table 13: Global targets

Target name	Description	For details, see...
Document 	Generates a FrameMaker document from the FrameMaker template you specify.	"Document targets" on page 162
Page 	Generates a series of connected body pages according to the master pages in the template you specify.	"Page targets" on page 168
Book 	Generates a book file that will contain subsequently generated documents.	"Book Targets" on page 177
Variable Format 	Modifies the definition of user-defined FrameMaker variable formats.	"Variable Format Target" on page 182
Log 	Outputs data to an ASCII file.	"Log target" on page 182
Marker 	When used in non-create mode, flows text at the location of the marker with a given type and marker text.	"Marker targets" on page 180

Document targets

The Document target generates output based on the FrameMaker template you specify when you create a pattern set. All pattern sets contain a Document target by default. This target serves as the container that holds the content of your document. See “Creation of the Pattern Set” on page 42 for more information on creating a pattern set.

The Document target makes a copy of the template and renames the copy based on selected options. The document name can be user-defined, data-driven, or generated automatically. You start with a blank document most of the time; therefore, the template should have *no* body pages. The exception is when you want to generate content into an existing document.

Typically, a PSet file has only one Document target, usually in the Root pattern; however, you can have more than one Document target in a PSet file. This might be necessary when you have introductory material that has a different layout structure than the rest of the document. Each structure is defined by a separate Document target. Conversely, some PSet files do not require a Document target at all

You can also invoke a particular Document target more than one time. For example, you might have a Document target that generates a chapter given a specific chapter number. You can then place the target in a pattern that loops for each chapter.

The Output tab

In the **Output** tab, you define the name of the generated document and specify if you want the output saved in PDF, HTML, and/or XML format (by default, the document is always saved as a Frame file). There’s also an option to close the document after it is generated. Some of the fields are identical to the ones you complete when creating a new PSet. If you completed those fields, the information is automatically displayed in the **Output** tab (Figure 93). Table 14 lists **Output** tab options.

Table 14: Output tab options

Option	Description
Document Naming Options	

Table 14: Output tab options (continued)

Option	Description
Program Generated	The PatternStream application will generate a document name with the prefix you indicate and a suffix of consecutive numbers. For example, the suffix for the first generated document might be 0001, the second document 0002, and so on.
User Defined	You specify the name of the document.
Data Driven	You assign a variable that contains the name of the document.
Output Specifications	
Close Document On Finish	If checked, the FrameMaker document is closed after it is generated. This is a good idea when you're generating more than 500 documents in Windows NT, which can run out of window handles if the documents are not closed.
PDF check box	Check if you want the document saved in PDF format. Specify the output directory for this file.
HTML check box	Check if you want the document saved in HTML format. Specify the output directory for this file.
XML check box	Check if you want the document saved in XML format. Specify the output directory for this file.
Directory/Browse button	For each box checked, type the output directory in the corresponding Directory field, or select the **Browse** button to select the location.

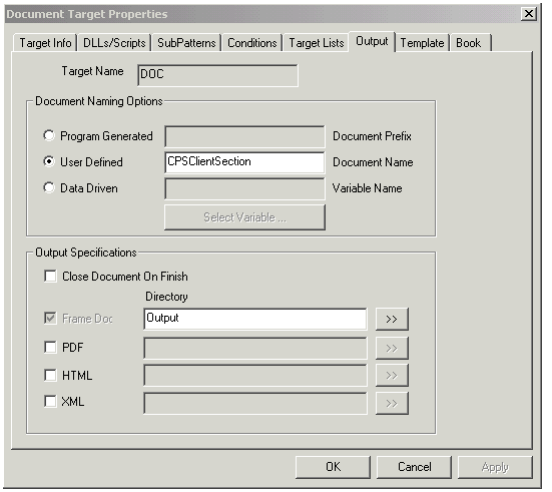


Figure 93: Output tab, Document target

The Template tab

When the PatternStream application generates a document, that document is formatted by a FrameMaker template. You specify the template in the Document target's **Template** tab (Figure 94). Table 16 describes the template options.

Table 15: Template options

Option	Description
Name	Specifies the location of the FrameMaker template that will format the PatternStream-generated FrameMaker document. If you specified one when you created your pattern set, it is displayed here.
Variable	Name of the variable containing the template name, if the template name is stored in the database. Select **Select Variable** to choose the variable.

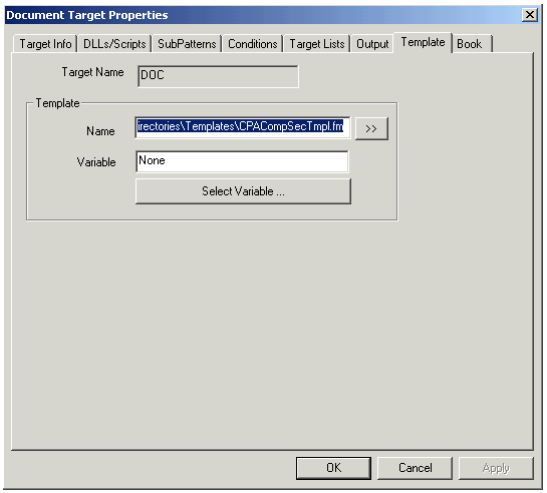


Figure 94: Template tab, Document target

The Book tab

Most PSET files generate only one output document. However, a document can be invoked an indefinite number of times during a given run. There can also be more than one document target in a given pset. As documents are generated, you can add them as components to a specified book file (see "Book Targets" on page 177). The attributes on the Document target's **Book** tab (Figure 95) are used to determine how the documents generated by a document target will be included in the current book, such as the position of the document, the page and paragraph numbering format, the side on which each file starts, etc. Table 16 describes the book options.

Table 16: Book options

Option	Description
Book Insertion options	
Include in Current Book	Documents generated by the current target will be inserted in the current book.
Component Name	The name of the book component to use when positioning the document in the book.

Table 16: Book options (continued)

Option	Description
Relative Position	The position in the book to insert the document, relative to the specified component. Valid options are: • After Component: The document just generated is placed after the component specified in the Component Name field. • Before Component: The document just generated is placed before the component specified in the Component Name field. • Current Component: Whenever a Document target creates a file (or component) in a book, that file becomes the current file. This option places the new file after the current file. The new file then becomes the current file. • End of Book: Places the currently generated document at the end of the book. • Beginning of Book: Places the currently generated document at the beginning of the current book.
Component Variable	If the component name will be data driven, the name of the variable containing the component. If a variable is specified, then the current value of the variable takes the place of the Component Name attribute.
Positional Conditioning	Only applies when you've selected the Current Component option under Relative Position. If the condition equals True, then the document is inserted after the component specified in Component Name. Otherwise, the document is placed after the current component. This feature is handy when generating a book that has a table of contents. The first chapter is placed after the table of contents and all subsequent chapters are placed after the previous chapter. In this case, the condition would be set to evaluate to True only on the first chapter.

Table 16: Book options (continued)

Option	Description
First Page Number	The first page number of the document.
Page Numbering Method	Method for numbering the pages in each chapter: • Continue: Page numbers continue from one file to the next. • From File: Reads the page numbers in each file. • Restart: Start over page numbers in each file.
Page Side	Start the file on a specific page: • Next Available: Start file on next page. • Start from File: Detect which page the file already begins on. • Next Left: Start the file on the next left page. • Next Right: Start the file on the next right page.
Paragraph Numbering Method	The method FrameMaker will use to autonumber the paragraphs in the document: • Continue: Continue autonumbering paragraphs from one file to another. • From File: Display page numbering method used in the file. • Restart: Start over autonumbering in each file.

Some important things to consider how the book placement attributes work are:

- When the component associated with the current output document is added to the book file, that component automatically becomes the current component.
- The component condition only applies when Current Component is selected as the position option. When the condition is True, the current component is set to the value of Component Name (or the value of Component Variable if a variable is specified).

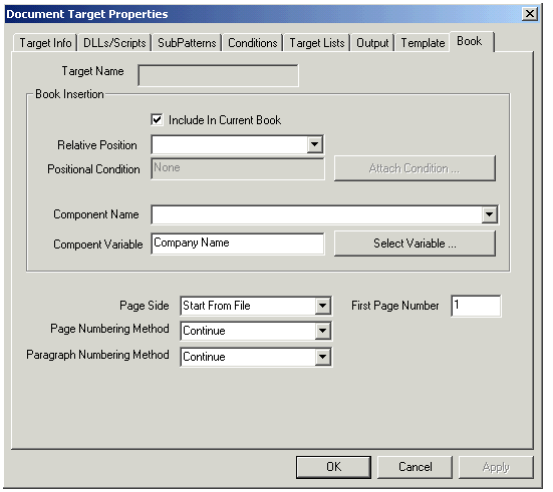


Figure 95: Book tab, Document target

- The position options End of Book and Beginning of Book are absolute positions which ignore the value of Component Name. However, the added component still becomes the current component.

Page targets

The Page target generates a series of connected body pages according to the master pages in the template you specify. You can select the master page to apply to each type of body page or store the master page names in a set of variables. In the Page target, you can also place unanchored frames directly onto the page by either specifying the X and Y coordinates or using a page grid. Figure 96 shows a series of connected body pages generated by the single invocation of a Page target. The flow begins on the first page and ends on the last page.

When the Page target in a PSet is invoked, it first creates a single body page. Next, the subtargets and subpatterns below the Page target insert text and other structural elements onto the body page, starting at point A. If the first page gets filled up, another body page is created automatically, and content begins to flow onto the new page (just as if you were typing the content by hand). When the second page is filled, a third page is created, and so on. When all of the subpatterns and subtargets are finished, the page target applies proper right and left background pages.

The illustration in Figure 96 demonstrates only one invocation of the Page target. If the Page target is in a pattern that loops more than once, the Page target

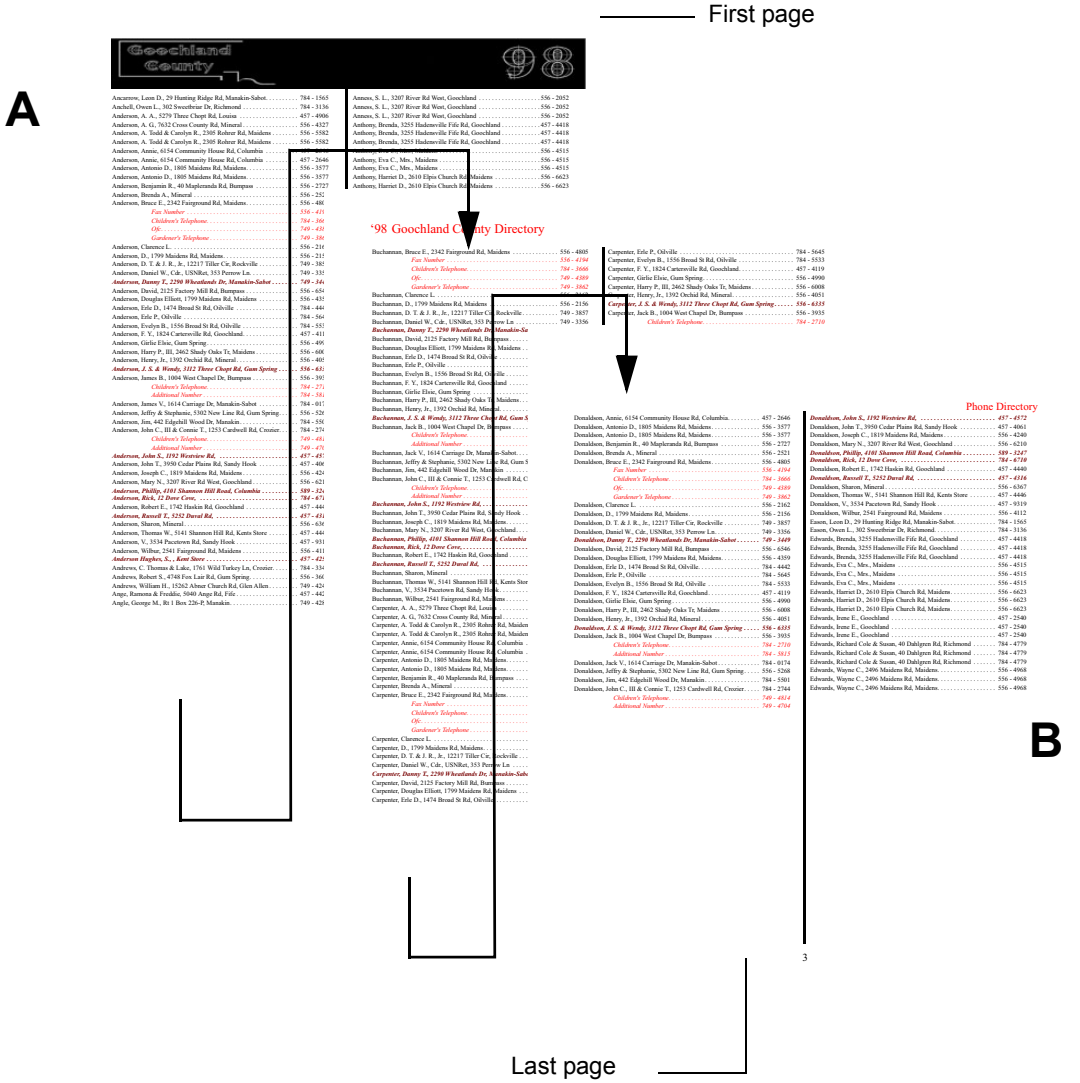


Figure 96: Connected body pages generated by a single invocation of a page target

will be invoked more than once, each time generating another set of connected body pages.

Note: The number of generated pages depends on the amount of content created by the patterns and subtargets below the Page target in the hierarchy. Typically, each invocation generates a different number of pages.

Target lists for Page targets

Page targets derive their set of valid targets lists from the master pages assigned in the **Layout** tab (see "The Layout tab" on page 171). On a master page of a FrameMaker document, you can create two types of text frames. The first type becomes part of the background of each body page which uses this master page. These text frames are not editable from the body pages. The other type is associated with a named flow. When a new body page is created using this master page, an corresponding editable text frame is created for each named flow on the master page. The position and size of each text frame on the body page is the same as it was for the corresponding frame on the master page.

For each named flow on the master page (right interior) specified in the layout tab of the page target properties dialog, there will be a corresponding named target list. Structural elements created by target objects in one of these named target lists will be inserted into the associated named flow on the body page that was created.

Figure 97 shows an example of a document that uses multiple named flows. The document is a database-derived questionnaire, which will be folded so that the address shows through the window of an envelope. To create the page, two named flows are defined on the master page. Flow A contains text that can flow through multiple body pages. Flow B holds the address, which must be placed precisely on the page to display through the clear envelope window.

The Frames target list has a special function. The contents of Unanchored Frame targets or TextFrame targets placed in the Frames target list will appear on every page in the current series of connected pages. The targets in this list create the effect of a variable background. See "Unanchored Frame target" on page 214 and "Text Frame Target" on page 231 for details.

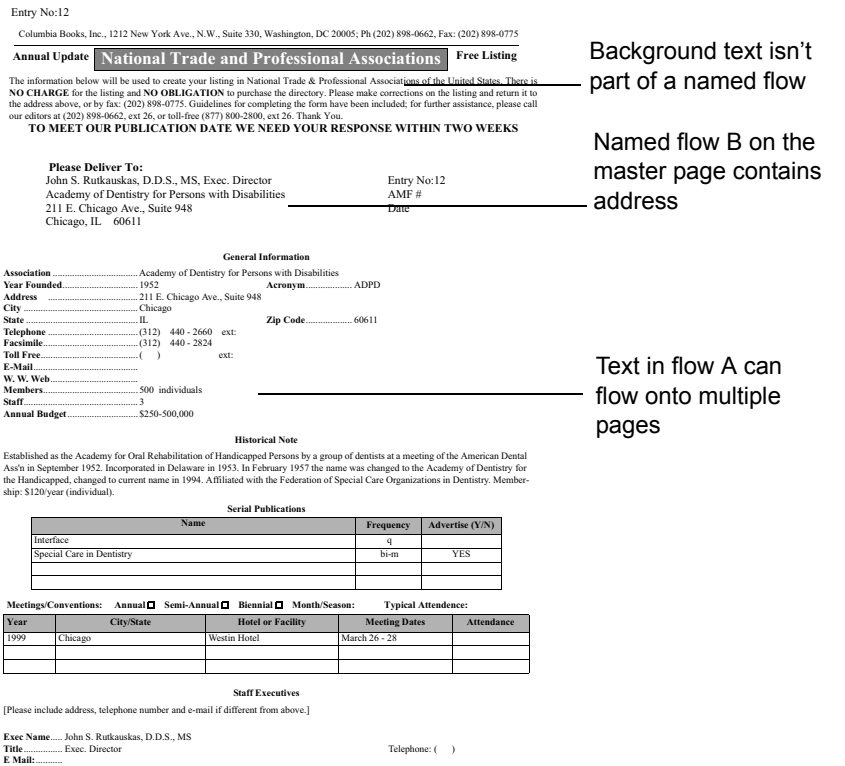


Figure 97: Multiple flows on a page

The Layout tab

You select which master pages are applied to output in the **Layout** tab (Figure 98). The master page names that appear in the drop-down lists for Interior, First, and Last (Right and Left) pages are from the FrameMaker template you specify in the parent document target.

Before the names of the master pages can be displayed in the drop-down lists, you must import them from the template file. Select **Extract Formats** from the **Template** menu on the **PatternStream** main dialog.

Instead of applying master pages from the FrameMaker template, you can associate PatternStream variables with master pages.

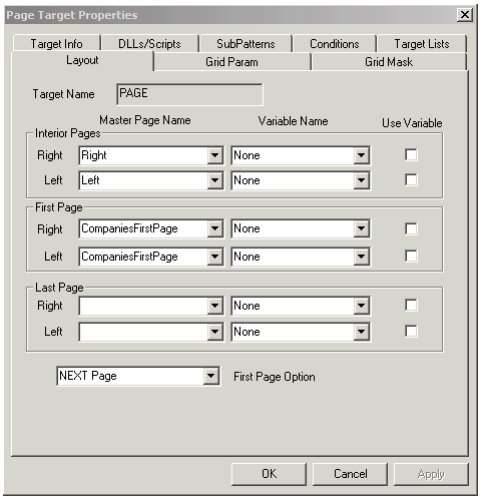


Figure 98: Layout tab, Page target

The Grid Param tab

You specify a position for unanchored frames on the **Grid Param** tab (Figure 99). These settings work in conjunction with Uframe targets. Unanchored frames generated by the Uframe target in this mode do not have a specific x, y location. Instead, the x, y locations are determined when the PatternStream application searches for unfilled portions of an imaginary grid. When the frame is positioned on the page, the grid cells that the frame covers are flagged as filled and cannot be used by another frame. The frame dimensions are assumed to be whole number multiples of the grid cell dimensions. The PatternStream application can search previous pages for partially filled grids or create new blank pages if the current page grid is full. Table 17 describes the settings on the **Grid Param** tab.

Table 17: Grid parameter options

Option	Description
Location	
Right Page X	The distance from the left edge of the upper right hand corner of the grid, on a right-hand page.

Table 17: Grid parameter options

Option	Description
Right Page Y	The distance from the page top of the upper right-hand corner of the grid, on a right-hand page.
Left Page X	The distance from the left edge of the upper right-hand corner of the grid, on a left-hand page.
Left Page Y	The distance from the page top of the upper right-hand corner of the grid, on a left-hand page.
Dimensions	
Cell Size Width	The width of each grid cell.
Cell Size Height	The height of each grid cell.
Grid Size Width	The width of the grid in number of cells.
Grid Size Height	The height of the grid in number of cells.
Fill Option	The name of the fill algorithm used to place frames onto a page: • Bottom UP: Starts with the bottom row of the grid. • Fill Current Page: Place frame only on the current page or subsequent pages, do not search previous page for a partially filled grid. • INNER to OUTER: Fill by column from the inner columns to the outer columns (first left column for a right page, last right column for a left). • OUTER to INNER: Fill by column from the outer columns to the inner columns (first left column for a right page, last right column for a left).

Table 17: Grid parameter options

Option	Description
Threshold	The fraction of the grid that needs to be filled before the grid on a page is complete. The PatternStream application will not add additional frames to a complete grid.
Grid Algorithm	The grid algorithm that points to an exported function in a declared DLL. The DLL contains a user-defined algorithm for filling a grid.
Units	The unit of measure for the specified dimensions.

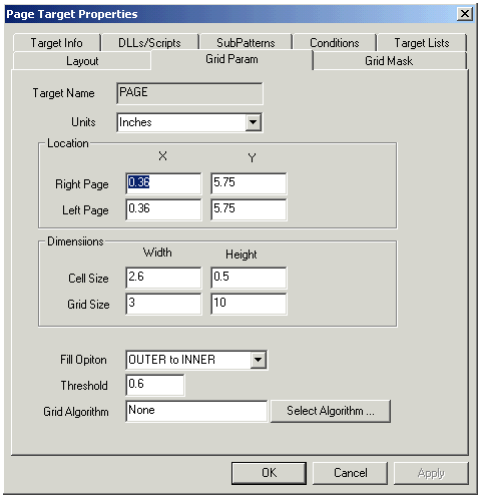


Figure 99: Grid Param tab, Page target

The Grid Mask tab

On the **Grid Mask** tab, you can cover, or mask, regions of the grid to prevent an unanchored frame from covering them (Figure 100). Each mask represent a rectangle of grid cells. Masks are applied depending on the type of page the parent grid is on. To create irregular shapes, you can place multiple masks on a grid. The mask value is used by custom grid algorithms and is not specifically used by the PatternStream application. However, the value must be positive in order for

the PatternStream application to recognize cells covered by a mask as being used. Table 18 describes the grid mask options.

Table 18: Grid mask options

Option	Description
Add	Adds the parameters specified in the Selected Mask pane.
Modify	Modifies the selected mask with what the values in the Selected Mask pane. (To select a mask, select the x value.)
Delete	Removes the selected mask from the parent page target. (To select a mask, select the x value.)
Selected Mask	
X Location	The number of grid cells from the left side of the page grid.
Y Location	The number of grid cells from the top of the page grid.
Width	The mask width in grid cells.
Height	The mask height in grid cells.
Mask Value	
Set Cell	Cell covered by the grid count toward threshold determination.
Ignore Cell	Cell is ignored when computing grid fill percent.
Set Value	Arbitrary value that can be passed to an externally defined grid algorithm.

Table 18: Grid mask options (continued)

Option	Description
Page Type	The type of page to apply this grid to. Notice that if you are using different master pages for the first and last page, you will need to set grid specifications for each page type. • First Left: First left page of a double-sided document. • First Right: First right page of a double-sided document. • Left Interior: The side closer to the binding (not counting first or last page). • Right Interior: The side farther from the binding (not counting first or last page). • Last Left: Last left page of a double-sided document. • Last Right: Last right page of a double-sided document.

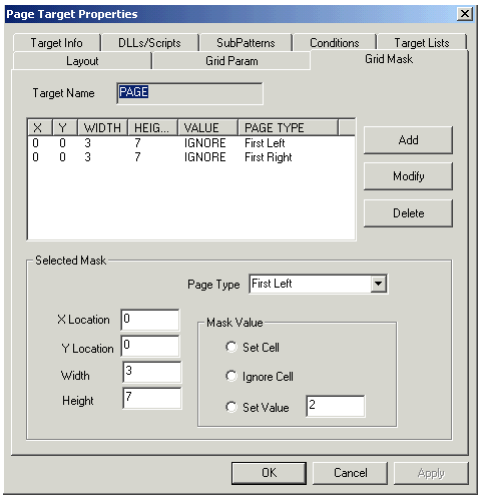


Figure 100: Grid Mask tab, Page target

Book Targets

When a PSet generates multiple documents, the PatternStream application can place these documents in a FrameMaker book file using a Book target. The PatternStream application uses an existing book, either by making a copy of the specified book (uses an existing book file as a template) or modifying an existing book, directly. The latter mode works best when the book components will be run separately from different PSet files at different times.

The following facts are important when considering how the book target will behave when components are added directly to an existing book:

- The Template Name and Template Variable controls are disabled, indicated that the values will be ignored.
- The output book file must already exist, or an error is generated.
- Only components that don't already exist in the output book file are created.
- Template functions, such as Extract Formats, will refer to the output book file rather than anything specified in Template Name.

Below is an example of a simple PSet Hierarchy that illustrates how to use book targets. Components are added to the output book file as the document targets below the book target generate output documents. For a discussion on how components are positioned in book, See "The Book tab" on page 165.

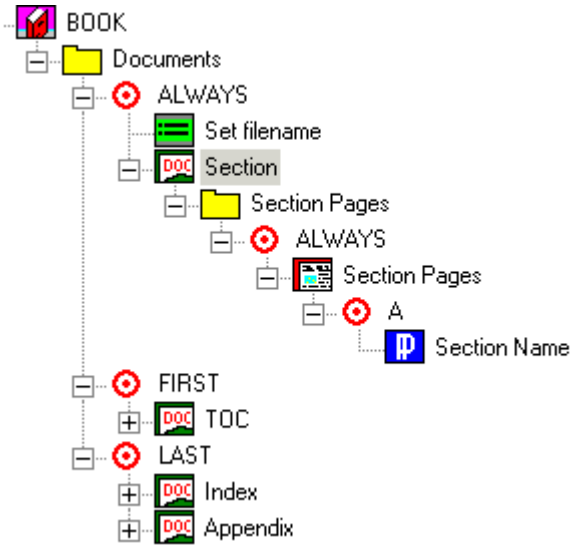


Figure 101: PSet Hierarchy that uses book targets and multiple document targets.

The Param tab

On the **Param** tab, you configure which book file and template file the PatternStream application will use to create and format the book file (Figure 102). Table 19 describes the **Param** tab.

Table 19: Parameter options

Option	Description
Template	
Use Template	The PatternStream application will make a copy of the name of the file specified and insert components into the copy.
Template Name	Browse to select the file to use as a template.
Template Variable	If your database contains the template name, select the template name variable.
Book Name	The name of the output book file.
Book Variable	The name of the variable containing the book name (if the name is data driven).
Directory	Indicate the directory containing the book file.

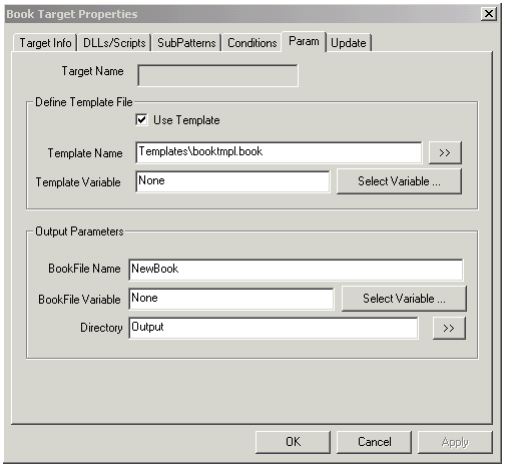


Figure 102: Param tab, Book target

The Update tab

On the Book target's **Update** tab, you indicate which components are updated in the final book (Figure 103). Table 20 describes these options.

Table 20: Update options

Option	Description
Page and Paragraph Numbering	Update the page numbers and autonumbered paragraphs, such as chapter numbers.
Generated Files	Update the generated files, such as the table of contents or index.
Cross References	Update the cross-reference links.
Reset Text Insets	Update the text inset links.
OLE Links	Update the linked objects, such as a PowerPoint slide.

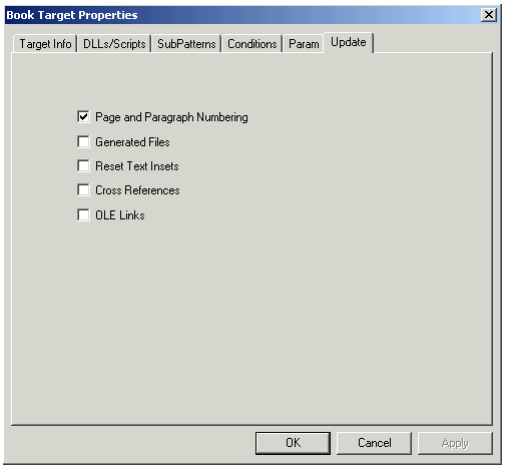


Figure 103: Update tab, Book target

Marker targets

Marker targets use FrameMaker markers to generate content at a specific location. The PatternStream application can either create this marker or search the FrameMaker document for an existing marker.

The Marker tab

Table 21 explains the marker options on the Marker tab.

Table 21: Marker target options

Option	Description
Marker Type	Select the marker type from the drop-down list. This list comes from the template specified in the project's Document target. If **Create Marker** is selected, then a marker of the selected type is created at the current insertion point. Otherwise, the PatternStream application searches the document for markers of the selected type.

Table 21: Marker target options (continued)

Option	Description
Marker Text	If **Create Marker** is not checked, the PatternStream application searches the document for the selected marker type and the designated marker text. If the marker type and text are not found, then an error is generated. *Note:* The PatternStream application ignores this parameter when creating a new marker. The text for the marker is derived from the attached string templates. See <<xref to String template>>.
Create Marker	Creates a marker of the selected type (if selected). The text of the marker is determined by attached string template objects. If not selected, a marker is not created. The PatternStream application searches the generated document for a marker with the given type and text. You attach the Attached Targets target list to the Marker target, then the targets in the list will be inserted in the document immediately after the selected marker. This option is used in conjunction with the Global named target list of a Document target. *Note:* You can only insert valid Flow Insertion targets in the Attached Targets list.
Text Variable	Lets you select the variable that will contain the marker text. The Marker target uses the current contents of the variable to search for a marker with the given type and text. Like static marker text, the text variable is ignored when the **Create Marker** check box is selected. For details on variables, see Chapter 5, "Variables." The Marker targets in the Globals named target list of a Document target (non-create mode) are not invoked until all subpatterns have been executed and the document is finished. Therefore, the Marker target can search for markers that were inserted previously in the current run.

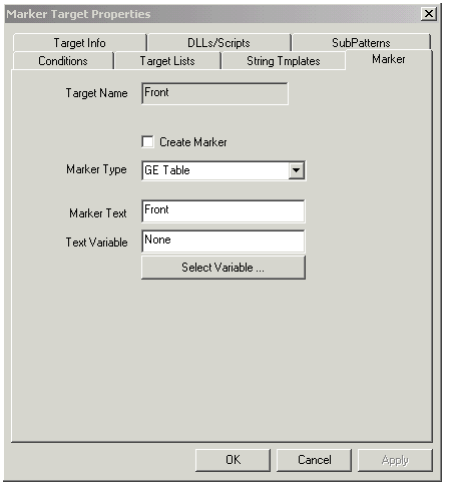


Figure 104: Marker tab, Marker target

Log target

The Log (LOG) target is a simple target that works with a Log File extension object to write data to an ASCII file. The extension specifies the .LOG file into which the Log target writes data. The data written to the log is specified by the string templates attached to the target.

The LogFile tab

Use the **LogFile** tab, shown in Figure 105, to select the extension that specifies the log file's name and location. Select the **Attach Log File** button and select an extension from the Select Extension dialog box.

Variable Format Target

Variable Format (VARFORMAT) targets are simple targets that modify an existing FrameMaker variable's definition. The string templates you attach to this target insert text into the variable format. These targets can only be used in the Global target list of the Document target.

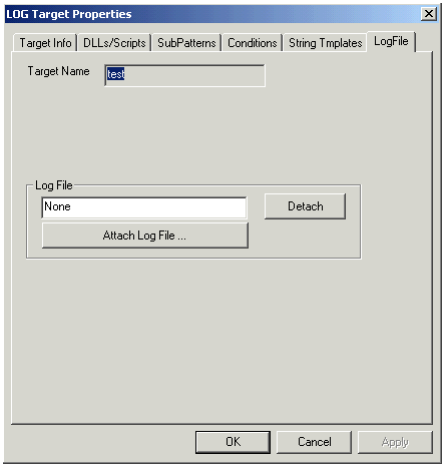


Figure 105: LogFile tab, Log target

The Format tab

Use the **Format** tab, shown in Figure 106, to select the variable whose definition you want to modify. The list of available variables comes from the FrameMaker template attached to the parent Document target.

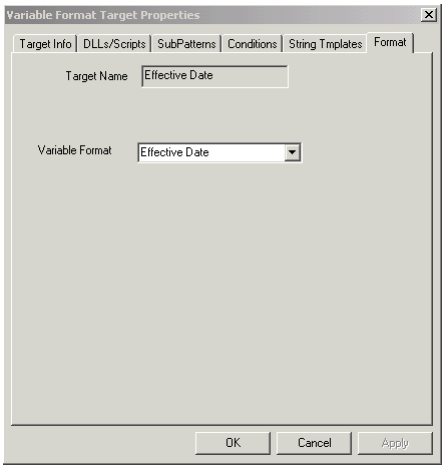


Figure 106: Format tab, Variable Format target

Chapter 10: Flow Insertion Targets

When you create text or structural elements such as paragraphs, table anchors, anchored frames, and markers, you insert these items into a text flow. The following are examples of text flows:

- A standard FrameMaker document that has a series of body pages and one main flow (typically called "A") on the master page. The PatternStream application inserts paragraphs, tables, and anchored frames into this main flow using the appropriate target object.
- A series of body pages with a master page that has multiple named flows. In this case, each named flow has a collection of targets that insert elements and text for that flow. There is a named target list for each of the named flows on the master page.
- Theoretically, table cells are like automatically resized text frames, each with its own unnamed flow. The targets attached to a table cell insert structural elements and text into each instance of the cell, as if it were a main flow. In FrameMaker, you can't insert tables within tables, so there are exceptions to generating nested tables in PatternStream. To generate nested tables, you must create an Anchored Frame target, insert a Text Frame target, then insert a Table target.
- You can attach a Text Frame target to either an anchored or an unanchored frame target. The subtargets attached to the text frame target will insert structural elements and text into the unnamed flow associated with each instance of the text frame.

Using Flow Insertion targets

By attaching subpatterns to Flow Insertion targets, you can create useful structures within your PSet hierarchy. For example, suppose your document consists of a series of tables. You might create a Flow Insertion target to which you attach each table subpattern. Subpatterns attached to Flow targets must contain content targets, such as Table and Paragraph targets, that generate output into a text flow.

You can only use flow insertion targets where it makes sense to insert items into a flow. For example, you cannot attach a Flow Insertion target directly to a Document target, because at the document level, you cannot clearly specify where content should go. Space for the content is specified at the page level. Likewise, you cannot place a Flow Insertion target directly below a Row target. Typically, table rows contain more than one cell and there is no way to distinguish one

table cell from another. They have no identifying features. Therefore, you must insert table cell sub-targets in a Row target to indicate where text will flow.

Note: One of the Flow Insertion targets is called *Flow*, so it's easy to confuse the two terms. Just remember that the capitalized Flow target is a type of Flow Insertion target.

Table 22 provides an overview of how Flow Insertion targets work.

Table 22: Flow insertion targets

Target name	Description	For details, see...
Paragraph 	Generates a paragraph in the format of the tag associated with the target in your template.	"Paragraph targets" on page 186
Flow 	Provides a specialized handle for flow insertion targets in the PSet hierarchy. Also used to implement a generalized continuation header for related groups of paragraphs.	"Flow targets" on page 191
Marker 	Inserts a marker of a specified type.	"Marker targets" on page 180
Anchored Frame 	Creates an anchored frame to which you can attach unanchored frames, text frames, and imported graphics.	"Anchored frame targets" on page 223
Table 	Generates a table according to the assigned table format.	"Table targets" on page 198

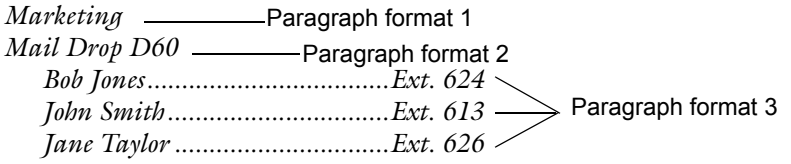
Paragraph targets

Paragraph targets create paragraph structural elements in your document. In the target's properties, you select the paragraph format to apply to the newly created paragraph. The FrameMaker template associated with the current PSet determines which paragraph formats can be applied to the target. Each time the target is invoked, the PatternStream application creates an instance of the paragraph with the given format. You attach string templates to the target to define the text

to be inserted into each new paragraph. Example 1 shows how Paragraph targets generate three types of formatted paragraphs.

Example 1: Paragraph formats

This example shows a pattern from a company directory.



Because the pattern uses three different FrameMaker paragraph formats, you would need to define at least three different Paragraph targets for this pattern.

In Example 1, the paragraphs containing the name and phone extension (format 3) are probably examples of one Paragraph target being invoked multiple times. This creates instances of a paragraph with a specific format.

The Setup tab

On the Paragraph target's **Setup** tab, you specify a limited set of paragraph overrides that will be applied conditionally to the paragraphs being generated by the parent target. (Figure 107). Table 23 describes the contents of the **Setup** tab.

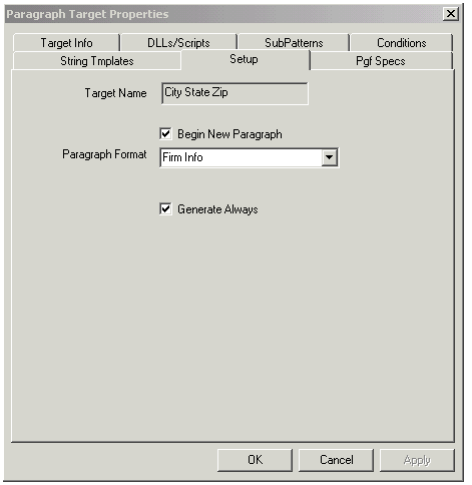


Figure 107: Setup tab, Paragraph target

Table 23: Contents of the Setup tab

Option	Description
Begin New Paragraph	Select to create a new paragraph each time the Paragraph target is invoked. This is the default setting; however, the new paragraph isn't necessary in all PSets. If this option isn't selected, the Paragraph target will insert text into the current paragraph. The current paragraph's tag is applied to the new content. In this case, you don't need to select a paragraph tag from the Paragraph Format drop-down list because the setting will be ignored.

Table 23: Contents of the Setup tab

Option	Description
Paragraph Format	Select a paragraph tag from the drop-down list to format the Paragraph target output. It is important to keep in mind that, to the PatternStream application, the paragraph format is just a name. The actual characteristics of a paragraph format are defined in the FrameMaker template. You can change the paragraph format attributes in the FrameMaker template without affecting the logical structure of the PSet file. If the Paragraph Format drop-down list is blank, you need to update the template information. Select the **PatternStream** menu in FrameMaker, select **Application,** then select **Update Template Info.**
Generate Always	If all main variables in the attached string templates have null values and this option is not selected, the Paragraph Target will not generate output. This is an alternative to attaching a condition that evaluates to "true" when the variables associated with the attached string templates don't have null values.

The Pgf Specs Tab

On the Paragraph target's **Pgf Specs** tab, you specify a limited set of paragraph overrides that will be applied conditionally to the paragraphs generated by the parent target (Figure 108). Table 24 describes the options.

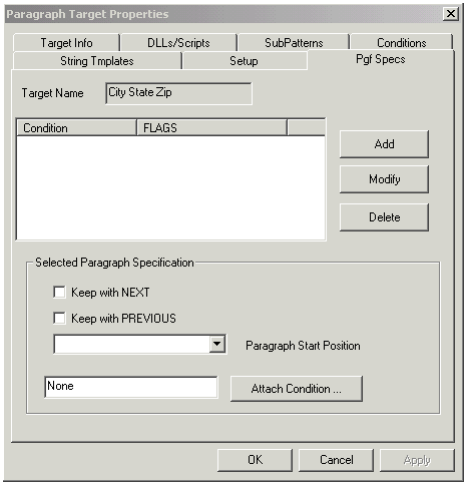


Figure 108: Pgf Specs tab, Paragraph tab

Table 24: Pgf Specs options

Option	Description
Add button	Adds a new paragraph specification to the Paragraph target.
Modify button	Applies the modifications to the currently selected Paragraph specification.
Delete button	Deletes the currently selected Paragraph specification.
Selected Paragraph Specification	
Keep with NEXT	Paragraph stays with the next paragraph when breaking across columns or pages.
Keep with PREVIOUS	Paragraph stays with the previous paragraph when breaking across columns or pages.

Table 24: Pgf Specs options (continued)

Option	Description
Paragraph Start Position	Position of the paragraph on the current page. Valid options are: • Anywhere • Top of Column • Top of Page • Top of LEFT Page • Top of RIGHT Page
Attach Condition	Used to select the condition object that will determine when to apply a particular paragraph ossification. If the attached condition evaluates to "true," the attribute values defined in the **Pgf Spec** tab are applied to the current paragraph.

Flow targets

Flow targets create no structural output themselves. They provide a convenient handle for attaching subpatterns that contain either Flow Insertion targets or targets that generate no structural elements (called *valid* Flow targets in this section). For example, suppose your document consists of a series of tables in the main flow. To generate these tables, you might place the Table target in a pattern controlled by an appropriate database query. As described in <<xref to Page target section>>, content intended for a named flow must start in the target list associated with that named flow, perhaps a target list with the same name as the flow. But you can't attach a subpattern to a target list, because subpatterns only attach to targets themselves. Flow targets solve the problem by acting as anchors between the target list and the subpattern. You insert the Flow target into the appropriate target list and then attach the subpattern to the target list.

Figure 109 illustrates how a Flow target serves as an anchor for a pattern.

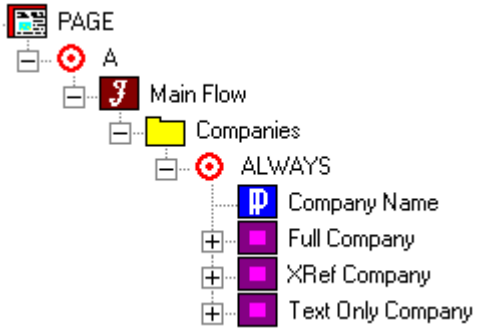


Figure 109: Flow target as an attachment point for a pattern.

One of the main consequences of the fact that the PSet Hierarchy represents the merging of the Logical Data Hierarchy with the Document Structure Hierarchy is that Patterns CANNOT be attached directly to a target list (in this case the A Target list) but must be attached to a target in that target list. This example illustrates the use of targets that function as a "handle" (for example, targets that generate no output) for pattern objects (Primary Numbers in this case) to effectively attach themselves to target lists.

When Flow targets serve as anchors in a target list, they resemble Blank targets with one very important difference. Targets and subpatterns below a Blank target derive their valid target list from the context of the parent Blank target; however, below a Flow target, valid target lists are derived from the Flow target itself, regardless of its context.

Flow targets can also create data-generated page or column continuation headers. If the contents generated at the beginning of a Flow target is on a different page or column than the contents generated at the end of the Flow target, the targets attached to the Continuation target list is generated.

The Cont HDR tab

The parameters on this tab control where and when the content generated by targets in the Continuation target list will display on the output page. Table 25 describes options on the **Cont HDR** tab

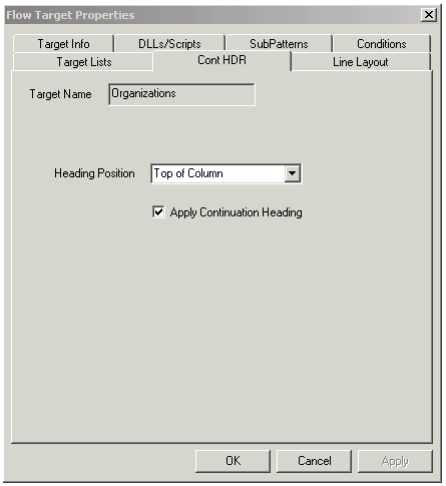


Figure 110: Cont HDR tab, Flow target

Table 25: Cont HDR options

Option	Description
Heading Position	This determines whether the structural elements generated by targets in the Continuation target list will be placed at the top of a page, frame, or column.
Apply Continuation Heading	Select to generate the Continuation target list, if appropriate. If not selected, the PatternStream application will ignore the contents of the Continuation list regardless of page or column breaking.

Chapter 11: Creating tables

This chapter describes how to create table structures with the PatternStream application. You create a table using a Table target and its associated subpatterns and subtargets. Figure 111 shows a typical pset subtree for a two column table. The major components of this example are labeled and described in Table 26. Here we have only labeled the major components that are found in almost every table and have ignored the ones that are implementation specific.

Table 26: Major elements of a typical table implementation in PatternStream.

Label	Description
A	This is the table target that when invoked, will create t a table using the assigned table format. The table format is defined in the usually way in the FrameMaker template file, and determines the following: • number of columns • column widths • the number of heading rows • the number of footing rows • the position of the table title.
B	These are table cell targets that deal with the heading cells. Notice that they are subtargets to the parent table target and are attached to the HEADING CELLS named target list. Besides the formatting of the heading cells, these table cell targets also use string templates to create the text for the table headings.
C	This subpattern attached to the table target will execute the query: **SELECT** courseID, credits, text **FROM** curriculumCourse **WHERE** (curriculumCourse.cID = :curriculum_id and curriculuCourse.seq_no=curriculum_seq_no **ORDER BY** seq_no2 and cycle for every row the query returns, invoking the target(s) contained in the pattern during each cycle.

Table 26: Major elements of a typical table implementation in PatternStream.

Label	Description
D	This is the main row target that generates a table row each time it is invoked. This means that the number of rows in the final table depends on the results of the query.
E	These are the table cell targets that format the cells belonging to each row that is generated by the parent row target. In this case they will use string templates to create the text that goes in each cell.

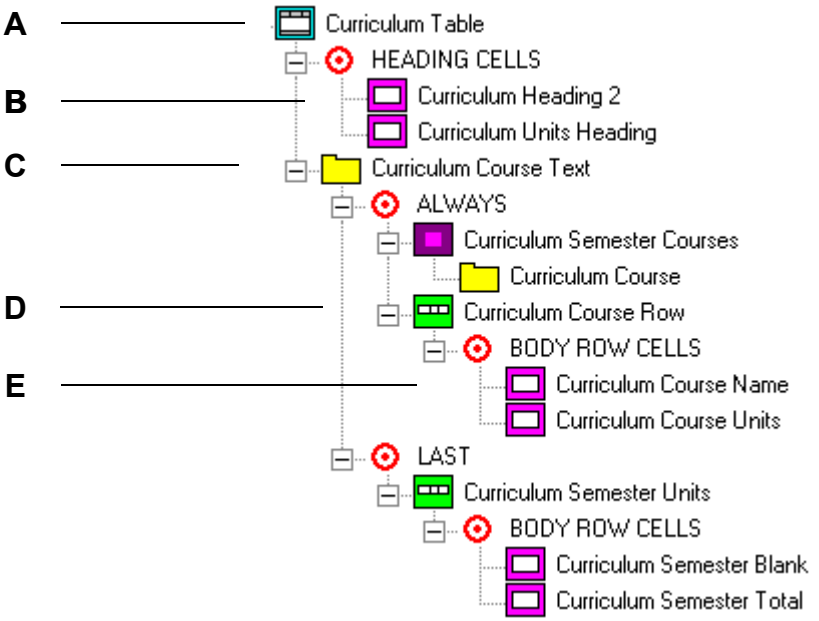


Figure 111: Typical table structure

Table 27 lists some common FrameMaker table features and describes where in the table construction process you define those features..

Table 27: FrameMaker table features

Table Feature	Define here...
Table format	**Format** tab of the Table Target Properties dialog box. Lets you select a table format defined in the FrameMaker template.
Table title	Create a string template for the title and attach it to the Table target, or for complex table titles, use the TITLE named target list.
Table heading or footing rows	From the **Target Lists** tab of the Table Target Properties dialog box, select either the Heading Cell or Footing Cell target list, then create Table Cell (or Variable Cell) targets in that list.
Custom cell ruling and shading	**Format** tab of the Table Cell (or Variable Cell) Target Properties dialog box.
Cell contents	The most common way of inserting text into a cell instance is creating string templates that define the cells' contents then attaching them to cell targets. You can also attach string templates to Paragraph targets, which are in turn attached to cell targets.

The targets described in this chapter are listed in Table 28.

Table 28: Table targets

Target name	Description	For details, see...
Table 	Generates a table according to the assigned table format.	"Table targets" on page 198
Row 	Generates a one or more table rows when invoked.	"Row targets" on page 202
Table Cell 	Operates on the tables cells of the associated row or table heading (footing). String templates associated with cells can be used to specify the content of the cell.	"Cell targets" on page 206

Table 28: Table targets (continued)

Target name	Description	For details, see...
Variable Cell 	Used to generate tables with data driven structure. Unlike Table Cell targets, Variable Cells refer to the column they are associated with by name rather than by number.	"Cell targets" on page 206

Table targets

A Table target is associated with a table format defined in the FrameMaker template. This table format determines the number of heading and footing rows, as well as whether the table has a title. For static tables, the table format usually determines how many columns the table will have and also the column widths.

The table target is both a compound target and a text target. When you attach a string template to a table target, the generated text is inserted into the table title.

The Format tab

Use the **Format** tab, shown in Figure 112, to specify table and paragraph formats for the table. Table 29 describes the contents of the **Format** tab.

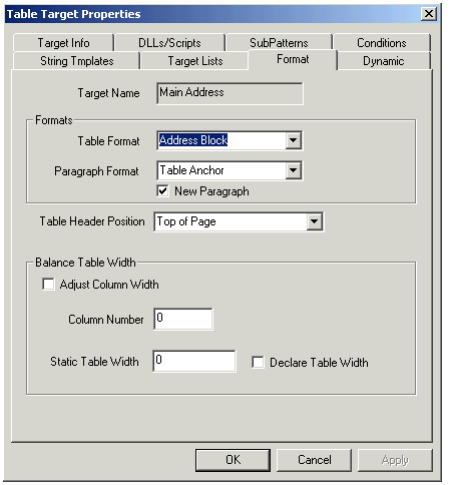


Figure 112: Format tab, Table target

Table 29: Contents of the Format tab

Option	Description
Formats	
Table Format	Select a table format from the drop-down list.
New Paragraph	Check if you want to create a separate paragraph for the table anchor. This generally gives more options as to how the table will flow onto the page, but is not necessary.
Paragraph Format	If a new paragraph is created along with the table, select a paragraph format from the drop-down list.
Adjust Column Width	Check to make the width of the table the same as the containing text frame or column (static tables only). You use this option when the width of a column is allowed to change dynamically and you wish the total table width to remain constant.
Column number	If you checked **Adjust Column Width**, specify the number of the column to adjust. This is the column that will resize to keep the overall table width the same as the enclosing text frame (or sub-column).

The Dynamic tab

Use the **Dynamic** tab, shown in Figure 113, to specify how the number of columns can change dynamically There are two ways to have the number of

columns be data-driven. Each method will use a different type of cell target. Table 30 describes the contents of the **Dynamic** tab.

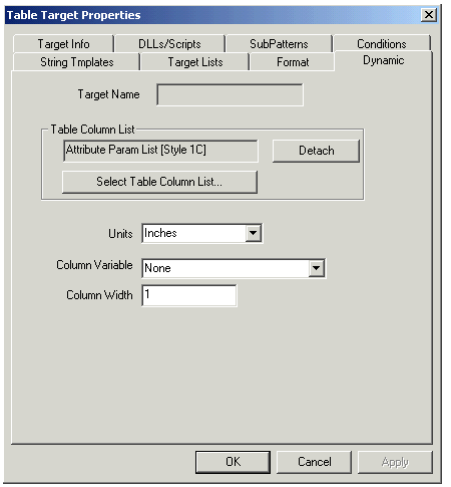


Figure 113: Dynamic tab, Table target

Table 30: Contents of the Dynamic tab

Option	Description
Method which uses variable cell targets	
Table Column List	Shows the name of the Table Column parameter list that defines the possible column names for a data-driven table. *Note:* Once you attach a Table Column List to a Table target, then you are committed to using Variable Table Cells instead of Static Table Cells.
Select Table Column List	Select to display the Select Parameter List dialog box. Select the Table Column List parameter list you want to attach to this table.
Detach	Select to detach the current Table Column List from this table.

Table 30: Contents of the Dynamic tab (continued)

Option	Description
Method which uses table cell targets	
Column Variable	This variable determines how many extra columns will be added to the basic table each time the table target is invoked.
Column width	The width of the columns that will be added to the table. *Note:* Currently, when you add columns this way all the additional columns will be the same width.
Units	Select **Inches, Picas,** or **Points** to specify the unit of measure for the value entered in Column Width.

Target lists for the Table target

There are four valid target lists available for attaching subtargets to the Table target (Table 31).

Table 31: Target lists specific to the Table target

Table list	Purpose
Heading Cells	The subtargets attached to this list specify the content of the table's heading cells.
Continuation	Use this list if your table has heading cells and you want the headers to contain different content after a column or page break. For example, the initial header reads "Western Region managers" but you want the repeated header to read "Managers" only.
Footing Cells	The subtargets attached to this list specify the content of the table's footing cells.

Table 31: Target lists specific to the Table target (continued)

Table list	Purpose
Title	Attach targets to this target list to create table titles with complex structure. Target types used must be flow insertion targets.

Row targets

Row targets are used to create the body rows of a table. Row targets can create more than one row at time. They can also be set to generate every *n*th iteration, allowing a row to contain more than one detail item. Some guidelines about Row targets are as follows.

- The cells of the rows generated are *not* manipulated by the row target itself. The cells take on the default properties of the table format that defined the original table. You must attach Table Cell or Variable Cell targets to the Row target (in the BODY ROW CELLS named target list) to manipulate the format and contents of the table cells that belong to each row generated.
- Row targets are pure structural (non-text) targets. You cannot attach String Templates to row targets.
- You can create row targets only in subpatterns that have a Table target as a parent or in the CONTINUATION target list of a Table target.

The row target is a compound target. The BODY ROW CELLS target list is the only available target list for attaching subtargets to a row target.

The Layout tab

Use the **Layout** tab, shown in Figure 114, to specify how many rows are created per cycle and when a new row is created.

Table 32 describes the contents of the **Layout** tab.

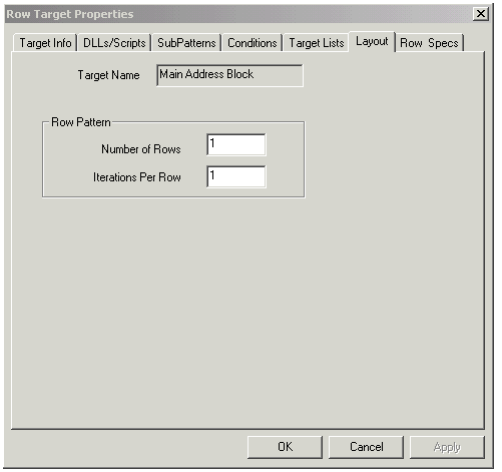


Figure 114: Layout tab, Row target

Table 32: Contents of the Layout tab

Option	Description
Number of Rows	Specify the number rows created during each cycle.
Iterations Per Row	Specify the number of cycles that must execute in the parent pattern before a new row(s) is created.

The Row Specs tab

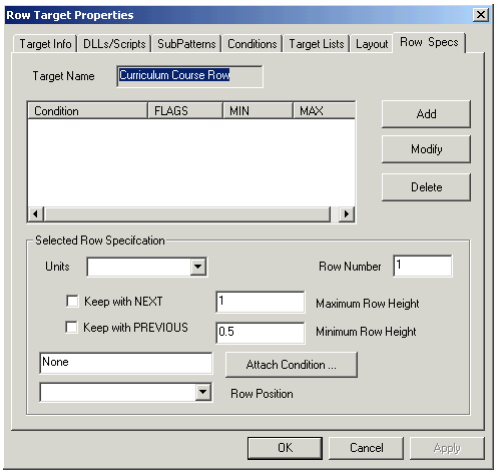


Figure 115: Row Specs tab, Row target

Use the **Row Specs** tab, shown in Figure 115, to modify the properties of rows being generated conditionally.

Table 33 describes the contents of the **Row Specs** tab.

Table 33: Contents of the Row Specs tab

Option	Description
Add button	Once you have specified parameters under Selected Row Specification, select this button to add the specification to the list.
Modify button	To modify the parameters of an existing row specification, select the condition name in the list, make changes under Selected Row Specification, then select this button.
Delete button	To delete a row specification, select the condition name then select this button.
Selected Row Specification	
Units	Specify the units for the Maximum Row Height and Minimum Row Height settings. Choose from inches, picas, or points.
Row Number	Specify the row number that the row specification applies to. This number is relative within each block of table rows created during each iteration.
Keep with NEXT	If this selection is checked, the current row will not be separated from the next row by a page or column break.
Keep with PREVIOUS	If this selection is checked, the current row will not be separated from the previous row by a page or column break.
Maximum Row Height	Specify the row's maximum height.
Minimum Row Height	Specify the row's minimum height.

Table 33: Contents of the Row Specs tab (continued)

Option	Description
Attach Condition	Select this button to display the Select Condition dialog box. Select the condition that determines if the row attributes defined by the Row Specification will be applied to the row generated by the current invocation of the row target.
Row Position	Specify where this row begins on a page. Choose from Anywhere, Top of Column, Top of Left page, Top of Page, and Top of Right Page.

Cell targets

Cell targets are the containers for a table's content, and the cell target's attached string templates provide that content. Cell targets are always subtargets of either row or table targets. Unlike most targets, cell targets do not generate a structural element. The table cells associated with the target are created when the parent table target creates an instance of the table, in the case of heading and footing rows, or the parent Row target creates a body row. There are two types of cell targets:

- Table Cell (TBLCELL) targets
  If the structure of the tables generated by a Table target is static, use Table Cell targets. Columns in Table Cell targets are referred to by number.
- Variable Cell (VARCELL) targets
  If the structure of the tables generated is data-driven, use Variable Cell targets. Columns in Variable Cell targets are referred to by name. Variable Cell targets are used in conjunction with Column Table Lists attached to the parent table. The Column Table List defines the possible columns for a table by name. Typically, the column width is derived from the database. If the column width is zero, then there will not be a column associated with a given name in the current table instance. That means, depending on what happens to the other columns, the column name and cell that a Variable Cell target refers to may be in column three for one table instance and in column five for another.

Both Table Cell and Variable Cell targets are compound targets. The Cell Content target list is the only valid target list for attaching subtargets to a cell target. See "Creating content for a cell target," for details on using this target list.

Creating content for a cell target

In the PatternStream application, a table cell is like an automatically resizing text frame that is attached to a larger table structure. Unlike most targets, cell targets do not generate a structural element; instead, they apply format parameters and insert content into cells already created by the parent target.

There are three basic way to create text content for a cell target:

- Attach one or more string templates to the cell target.
- Within the cell subtargets, create targets such as Anchor Frame or Paragraph in the Cell Content target list, then attach string templates or other subtargets to those targets. Targets in the Cell Content target list will generate in the current cell instance. Figure 116 shows how this structure might look in the PSet hierarchy:

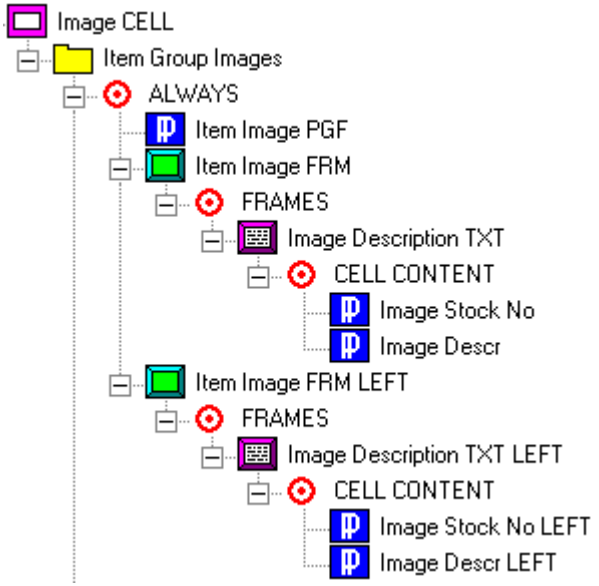


Figure 116: Cell content in a pset hierarchy

- Attach a subpattern to the cell target. Any content created by the subtree will be inserted into the current cell instance.

The Format tab

Use the **Format** tab, shown in Figure 117, to apply custom ruling and shading to a cell, rotate the cell, and specify a paragraph format for the cell's contents.

The **Format** tabs for the Table Cell and Variable Cell are identical. Table 34 describes the contents of the **Format** tab.

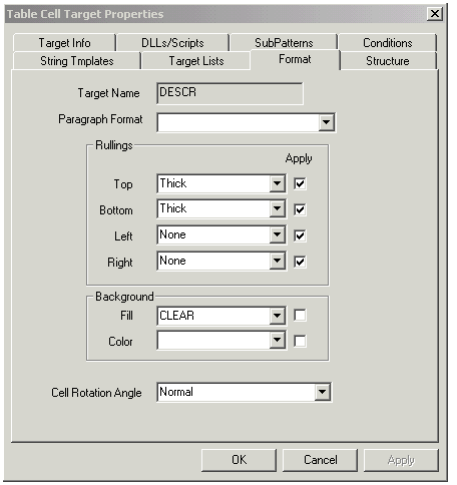


Figure 117: Format tab, Table Cell target

Table 34: Contents of the Format tab

Option	Description
Paragraph Format	Select the paragraph format to apply to the first paragon in the cell.
Rulings	
Top/Apply	Select the ruling to apply to the top of the cell. Check the **Apply** button to apply this ruling.
Bottom/Apply	Select the ruling to apply to the bottom of the cell. Check the **Apply** button to apply this ruling.
Left/Apply	Select the ruling to apply to the left of the cell. Check the **Apply** button to apply this ruling.
Right/Apply	Select the ruling to apply to the right of the cell. Check the **Apply** button to apply this ruling.

Table 34: Contents of the Format tab (continued)

Option	Description
Background	
Fill/Apply	Select the shading fill to apply to the cell. Check the **Apply** button to apply this setting.
Color/Apply	Select the shading color to apply to the cell. Check the **Apply** button to apply this setting.
Cell Rotation Angle	Select the amount of rotation to apply to the cell's contents. The options are Normal, 90 Degrees, 180 Degrees, and 270 Degrees.

The Structure tab for Table Cell targets

The **Structure** tab for the Table Cell Target Properties dialog box, shown in Figure 118, specifies the cell's location in the table and whether the cell straddles columns or rows. Table 35 describes the contents of the **Structure** tab for Table Cell targets.

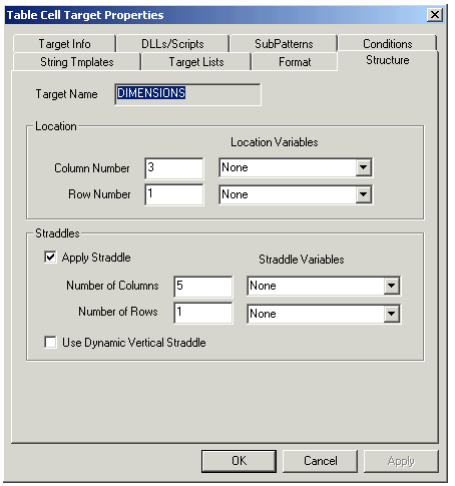


Figure 118: Structure tab, Table Cell target

Table 35: Contents of the Structure tab for Table Cell targets

Option	Description
Location	
Column Number	Specify the number of the column this cell is in.
Column Variable	Select the variable that will contain the column number. The value of this variable will override what is in the Column Number. *Note:* If the value of the variable is zero, then the Column Number will be used as a default.
Row Number	Specify the number of the row this cell is in. The row number refers to the relative position of the cell in the instance of the structural element generated by the parent. For example, if the parent target is a Row target creating two rows at a time and if row number = 2, then the 2 refers to the second row generated each time.
Row Variable	Select the variable that will contain the relative row number. The value of this variable will override what is in the Row Number. *Note:* If the value of the variable is zero, then the Row Number will be used as a default.
Straddles	
Number of Columns	Specify the number of columns to straddle the cell.
Column Straddle Variable	Select the variable that will contain the number of columns to straddle. The value of this variable will override what is in the Number of Columns. *Note:* If the value of the variable is zero, then the Number of Columns will be used as a default.
Number of Rows	Specify the number of rows to straddle the cell.

Table 35: Contents of the Structure tab for Table Cell targets (continued)

Option	Description
Row Straddle Variable	Select the variable that will contain the number of rows to straddle. The value of this variable will override what is in the Number of Rows. *Note:* If the value of the variable is zero, then the Number of Rows will be used as a default.
Use Dynamic Vertical Straddle	Check to set dynamic straddle, which makes the cell straddle all of the rows of the cell target instances for this column.

The Structure tab for Variable Cell targets

The **Structure** tab for the Variable Cell Target Properties dialog box, shown in Figure 119, specifies the cell's location in the table and whether the cell straddles columns or rows. Table 36 describes the contents of the **Structure** tab for Variable Cell targets.

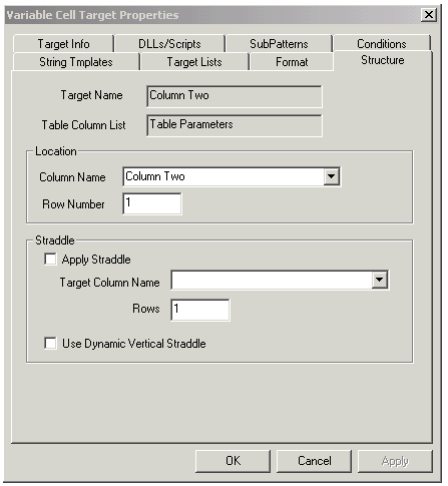


Figure 119: Structure tab, Variable Cell target

Table 36: Contents of the Structure tab for Variable Cell targets

Field or check box	How to use/Purpose
Location	
Column Name	Select the name of the column in which to place this cell.
Row Number	Specify the number of the row this cell is in. The row number refers to the relative position of the cell in the instance of the structural element generated by the parent. For example, if the parent target is a Row target creating two row at a time and if row number = 2, then the 2 refers to the second row generated each time.
Straddle	
Target Column Name	Specifies the name of the straddled column. Because the column number for a named column can vary for each instance, the number of columns straddled will typically vary also.
Number of Rows	Specify the number of rows to straddle the cell.
Column Straddle	Select to set dynamic straddle, which makes the cell straddle all of the rows of the cell target instances for this column. Works like the dynamic vertical straddle for static table cells.

Chapter 12: Frame targets

This chapter describes targets that are used to create frame elements. These frame elements are fundamental to understanding FrameMaker (that is where it gets its name) and almost certainly will be used to create any really complex output. There are three basic types of frame element:

Unanchored Frames

These are logical rectangular elements that hold other graphic elements, such as text frames and other unanchored frames. The important thing to remember is that they are places directly on the page by assigning them an coordinate. If the unanchored frame is being placed in another frame element, then these x,y coordinates are relative to the parent frame. Figure 120 illustrates this.

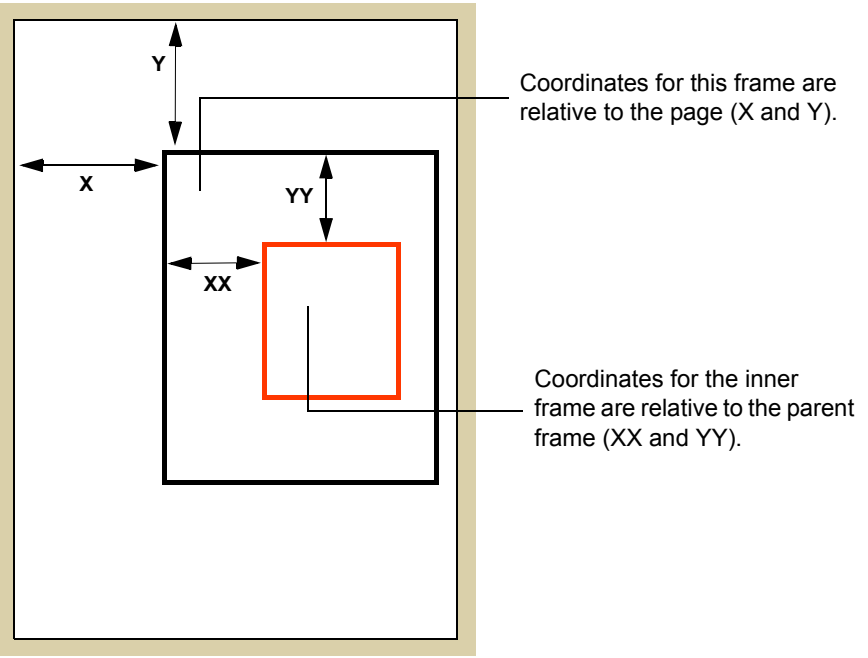


Figure 120: XY coordinates for an unanchored frame.

Anchored Frames

These elements are similar to unanchored frames except that, rather than specifying an X,Y location on a page, the frame is anchored to the text, very similar to

the way a table is inserted into the text flow. For manual users of FrameMaker, the frame will move from one page to another as text gets added or deleted. Anchored frames have type and alignment attributes that determine how the FrameMaker formatting engine will place the frame on the page.

Text Frames

These elements are frames specifically for providing rectangular areas that will contain formatted text. They essentially contain an unnamed flow where any flow insertion element can be used. Like unanchored frames, they are positioned by assigning an X,Y location (although in PatternStream, these coordinates are represented as left-right margins). You can create the effect of an anchored text frame by first inserting an anchored frame, then placing a text frame in it.

The targets described in this chapter are listed in Table 37.

Table 37: Frame targets

Target name	Description	For details, see...
Unanchored Frame 	Generates an unanchored frame in three different modes: 1. Identical frames at a given x, y location on every body page in a sequence of connected body pages. 2. Single frame in a given x, y location on the current page. 3. Single frame in a location determined by a page grid algorithm.	"Unanchored Frame target" on page 214
Anchored Frame 	Creates an anchored frame to which you can attach unanchored frames, text frames, and imported graphics.	"Anchored frame targets" on page 223
Text frame 	Creates a text frame inside an anchored or unanchored frame.	"Text Frame Target" on page 231

Unanchored Frame target

An *unanchored frame target* generates an unanchored frame. There are three possible modes for this target and which mode applies for any instance of an

unanchored frame target depends on its location in the PSET hierarchy and on various attributes assigned.

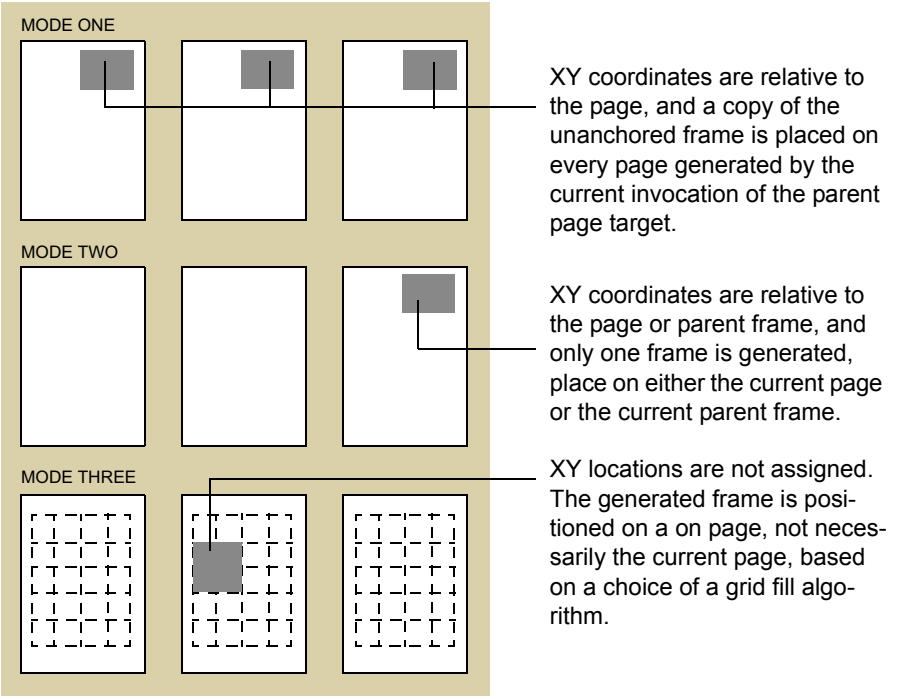


Figure 121: The different modes for an unanchored frame target.

- When placed in the FRAMES list of a Page target, the target generates an unanchored frame for each connected page in the current pattern set. The X and Y coordinates are explicitly declared (you can specify a different x, y pair for left and right pages), and the contents of each unanchored frame are identical for each page in the current set. The effect is like having variable (data-driven) background pages.
- If the unanchored frame target is placed in a pattern (or as a subtarget) that requires a flow insertion target and the grid option is turned off, the unanchored frame will be placed on the current page using the specified x, y coordinates. If two unanchored frames from the same Unanchored Frame target are placed on the same page, the second frame will be placed on top of the first.
- If the unanchored frame target is placed in a pattern (or as a subtarget) that requires a flow insertion target and the grid option is turned on, the unan-

chored frame will be placed on the page based on a selected page filling algorithm. In this mode, the x, y coordinates are not given. The target works in conjunction with a *page target* grid (see "Page targets" on page 168).

The Unanchored Frame target is a compound target with one available target list called FRAMES. Valid targets for this list include *text frame* and other *unanchored frame* targets, along with targets that do not generate output.

Parameter settings and frame dimensions

There are many ways to control the dimensions of the resulting unanchored frame. The behavior will depend on the resize option selected and whether there will also be an imported image file. In general, when there is an imported graphic, the frame dimensions will be determined by the size of the graphic. Table 38 lists the various options and behavior.

Table 38: Unanchored Frame settings

Resize Option	Behavior
Has Graphic	
Adjust Frame to Graphic	• Inset dimensions are determined by the image dimensions. • Frame dimensions will be determined by the formulae $FrameWidth = InsetWidth + (2 \times XOffset)$ $FrameHeight = InsetHeight + (2 \times YOffset)$
Adjust Graphic to Frame	• Inset dimensions are determined according to the formula: $InsetWidth = FrameWidth - (2 \times XOffset)$ • Height of the inset is scaled to keep the image in proportion. • Frame dimensions are what is declared in the target attributes.
Import As Is	• Frame dimensions are what is declared in the target attributes. • Inset dimensions are determined by the image dimensions.

Table 38: Unanchored Frame settings

Resize Option	Behavior
Declare Size	• Frame dimensions are what is declared in the target attributes. • Inset dimensions are what is declared in the target attributes. • If inset height = 0, then the height is scaled to keep image in proportion. • If inset width = 0, then the width is scaled to keep the image in proportion. • If the frame width = 0, then the frame width is determined by the formula: $FrameWidth = InsetWidth + (2 \times XOffset)$ • If the frame height = 0, then the frame height is determined by the formula: $FrameHeight = InsetHeight + (2 \times YOffset)$ *Note:* If after all this, any of the dimensions are zero, then an error will be generated.
No Graphic	
Adjust Parent to Frame	• Frame dimensions are what is declared in the target attributes. • Adjust the parent frame to the size of the frame according to the formula: $ParentFrameWidth = FrameWidth + (2 \times XLoc)$ $ParentFrameHeight = FrameHeight + (2 \times YLoc)$
Adjust Frame to Parent	• Adjust the frame to the size of the parent frame according to the formula: $FrameWidth = ParentFrameWidth - (2 \times XLoc)$ $FrameHeight = ParentFrameHeight - (2 \times YLoc)$
Import As Is	NA
Declare Size	• Frame dimensions are what is declared in the target attributes.

UFrame tab

Use the **UFrame** tab, shown in Figure 122, to specify the unanchored frame's basic properties, such as border and runaround properties. Table 39 describes the contents of the **UFrame** tab.

Table 39: Contents of the UFrame tab

Option	Description
Units	Select the units to use for the values of dimensional attributes. Choose from Inches, Millimeters, Picas, and Points.
Common Frame Properties	
Pen Type	Select a pen type for the frame's border (Clear specifies an invisible border).
Border Width	Specify the width of the frame's border.
Frame Extension	If the final height of the frame is dependent on attributes of some other object, such as an imported graphic or a child text frame, then this amount is added to that final computed height.
Fill	Select a background fill for the frame.
Color	Select a background color for the frame.
Measure Y Loc from Bottom of Parent	Check to place the frame's Y coordinate at the bottom of the frame generated by its parent target.
Angle	Select an angle to rotated the frame. Choose from Normal, 180 degrees, 270 degrees, and 90 degrees.
Relative Position	Select the frame's relative position to other frames. Choose from Bring to Front, Send to Back, and Keep Current Draw Order.
Runaround Properties	
Type	Specify how text should run around the frame. Choose from None, Bounding Box, and Follow Contours.

Table 39: Contents of the UFrame tab (continued)

Option	Description
Gap	If you selected Bounding Box or Follow Contours as the runaround behavior, specify the gap between the frame and runaround text.

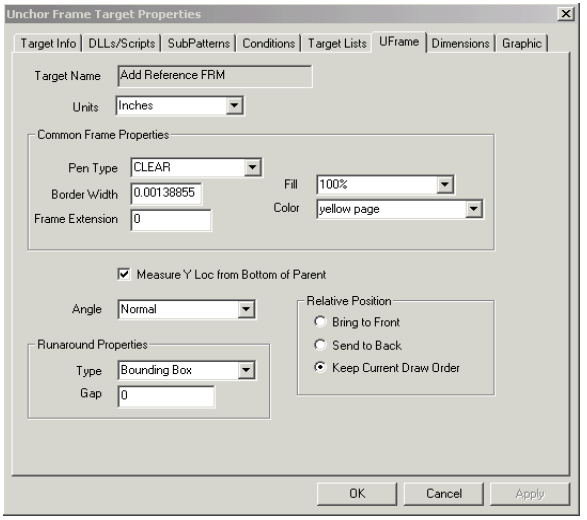


Figure 122: UFrame tab, Unanchored Frame target

Graphic tab

Use the **Graphic** tab, shown in Figure 123, to specify the graphic to place in the unanchored frame. Table 40 describes the contents of the **Graphic** tab.

Table 40: Contents of the Graphic Tab

Option	Description
Has Graphic	Check to indicate that a graphic inset will be created along with the unanchored frame.

Table 40: Contents of the Graphic Tab

Option	Description
Filename Variable	The name of the variable that contains the filename of the graphic. To select the variable, select the **Select Variable** button and choose a variable from the Select Variable dialog box.
Default File	The name of the graphic file to use if the file indicated by the filename variable does not exist. select the >> button to browse for the file.
Directory Variable	The name of the variable that contains the directory where the graphic file resides. To select the variable, select the **Select Variable** button and choose a variable from the Select Variable dialog box.
Default Directory	The name of the directory to use if the directory indicated by the directory variable does not exist. select the >> button to browse for the file.
Inset Properties	
Inset Height	If the height is set to 0, the graphic is scaled proportionally to the given height. Otherwise, specify a height.
Inset Width	If the width is set to 0, the graphic is scaled proportionally to the given width. Otherwise, specify a width.
Dots per inch	The DPI setting for the graphic.
Border width	Specify the graphic inset's border width.
X Offset	The X coordinate of the upper left corner of the graphic inset within the anchored frame.
Y Offset	The Y coordinate of the upper left corner of the graphic inset within the anchored frame.
Pen Type	The pen type for the border.

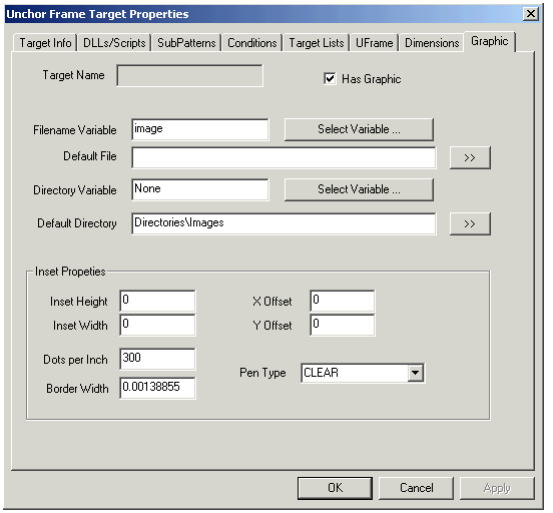


Figure 123: Graphic tab, Unanchored Frame target

Dimensions tab

Use the **Dimensions** tab, shown in Figure 124, to specify the unanchored frame's position. Table 41 describes the contents of the **Dimensions** tab.

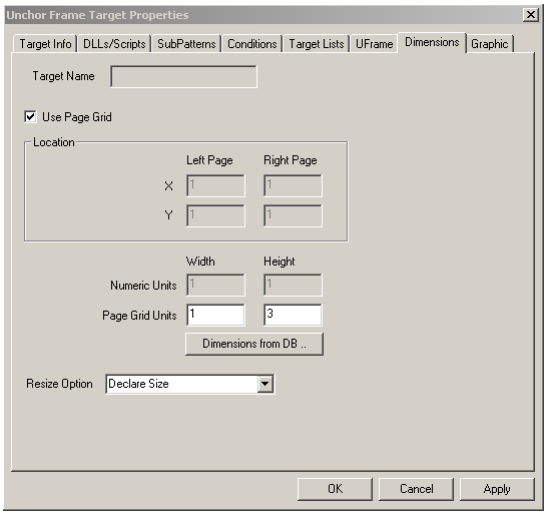
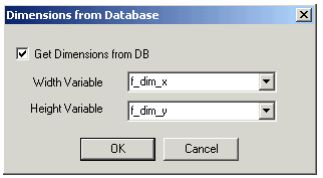


Figure 124: Dimensions tab, Unanchored Frame target

Table 41: Contents of the Dimensions tab

Option	Description
Location	
Use Page Grid	Check to use the current page grid for determining where to place the unanchored frame. See "Page targets" on page 168 for details on page grids.
Note: The following parameters take effect when the **Use Page Grid** option is turned off.	
X/Left Page	The distance from the upper left corner of the frame to the left side of the page or parent frame on a left side page.
Y/Left Page	The distance from the upper left corner of the frame from the top of the page or parent frame on a left side page.
X/Right Page	The distance from the upper left corner of the frame to the left side of the page or parent frame on a right side page.
Y/Right Page	The distance from the upper left corner of the frame from the top of the page or parent frame on a right side page.
Numeric Units/ Width	The declared width of the unanchored frame.
Number Units/ Height	The declared height of the unanchored frame.
Note: The following parameters take effect when the **Use Page Grid** option is turned on.	
Page Grid Units/ Width	The width of the frame in terms of grid cells.
Page Grid Units/ Width	The height of the frame in terms of grid cells.

Table 41: Contents of the Dimensions tab (continued)

Option	Description
Dimensions from DB	If the frame's dimensions are located in the database, select the **Dimensions from DB** button. In the dialog box, select the **Get Dimensions from DB** button then select the variables for the frame's width and height from the drop-down lists,  *Note:* The values of these variables will be interpreted as actual dimensions or in terms of grid cells depending on the **Use Page Grid** option that is chosen.
Resize Option	Select whether the graphic and/or frame should be resized. Choose from Adjust Frame/Parent Width, Adjust Graphic/Frame Width, Declare Size, and Import As Is.

Anchored frame targets

Anchored frame targets generate anchored frames along with an optional graphic inset for imported images. You can have the frame dimensions track the image size or scale the image to a specific size. When importing a graphic, the behavior of the anchored frame target is almost the same as a *graphic segment* (see "Graphic segment" on page 269). However, anchored frame targets, unlike, graphic segments, are part of the pattern set hierarchy. This allows you to attach subtargets and subpatterns.

Parameter settings and frame dimensions

Table 42:

Resize Option	Behavior
Has Graphic	
Adjust Cell Width	• Inset dimensions are determined by the image dimensions. • Frame dimensions will be determined by the formulae $FrameWidth = InsetWidth + (2 \times XOffset)$ $FrameHeight = InsetHeight + (2 \times YOffset)$ • If the frame is a table cell, then the column width will be adjusted to the inset width. • Ignores the frame dimensions.
Adjust Graphic	• Adjust the frame width to the width of the containing table cell or text column. • Inset dimensions are determined according to the formula: $InsetWidth = FrameWidth - (2 \times XOffset)$ • Height of the inset is scaled to keep the image in proportion. • Frame height is determined by the formula: $FrameHeight = InsetHeight + (2 \times YOffset)$
Import As Is	• Inset dimensions are determined by the image dimensions. • Frame dimensions are what is declared in the target attributes. • If the frame width = 0, then the frame width is determined by the formula: $FrameWidth = InsetWidth + (2 \times XOffset)$ • If the frame height = 0, then the frame height is determined by the formula: $FrameHeight = InsetHeight + (2 \times YOffset)$

Table 42:

Resize Option	Behavior
Declare Size	• Frame dimensions are what is declared in the target attributes. • Inset dimensions are what is declared in the target attributes. • If inset height = 0, then the height is scaled to keep image in proportion. • If inset width = 0, then the width is scaled to keep the image in proportion. • If the frame width = 0, then the frame width is determined by the formula: $FrameWidth = InsetWidth + (2 \times XOffset)$ • If the frame height = 0, then the frame height is determined by the formula: $FrameHeight = InsetHeight + (2 \times YOffset)$ *Note:* If after all this, any of the dimensions are zero, then an error will be generated.
No Graphic	
Adjust Cell Width	• Frame dimensions are what is declared in the target attributes. • If the frame is a table cell, then the column width will be adjusted to the inset width.
Adjust Frame	• Adjust the frame width to the width of the containing table cell or text column. • Frame height is what is declared in the target attributes.
Import As Is	NA
Declare Size	• Frame dimensions are what is declared in the target attributes.

The Aframe tab

The **Aframe** tab contains settings for the anchored frame. Most of the options are similar to those found in FrameMaker. Table 43 describes the **Aframe** tab options.

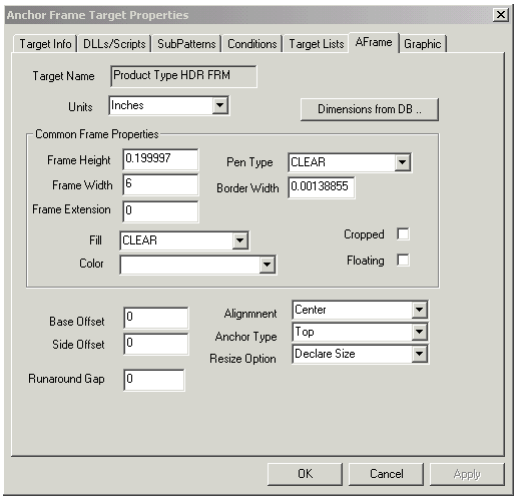


Figure 125: Aframe tab, Aframe target

Table 43: AFrame tab options

Option	Description	Related FrameMaker option
Units	The unit of measure for the anchored frame dimensions.	Display units under the **View** menu/**Options** selection.
Dimensions from DB		
Get Dimensions from DB	Retrieves the dimensions of the frame from the database.	None
Width Variable	The variable that contains the anchored frame width.	None

Table 43: AFrame tab options (continued)

Option	Description	Related FrameMaker option
Height Variable	The variable that contains the anchored frame height. *Note:* The height and width variables must be floating point digits, such as "1.5," "2.9," and so on.	None
Common Frame Properties		
Frame Height	The height of the anchored frame.	Height in Anchored Frame properties
Frame Width	The width of the anchored frame.	Width in Anchored Frame properties
Frame Extension	Amount to extend the height of the frame when using a program generated height (see "Resize Option" on page 229).	None
Pen Type	The pen to use for the borders of the anchored frame.	Pen Pattern in Tools Palette
Border Width	The width of the anchored frame border.	Border Width field in Object Properties
Fill	The color that shades the anchored frame.	Tint in Object Properties
Color	The color of the anchored frame (select from several predefined colors).	Color in Object Properties
Cropped	Displays the anchored frame graphic inside the column. A graphic that extends past the column is cropped.	Cropped Anchored Frame properties
Floating	Moves the anchored frame to the next available space; frame isn't anchored to a paragraph.	Floating Anchored Frame property

Table 43: AFrame tab options (continued)

Option	Description	Related FrameMaker option
Alignment	The alignment of the anchored frame: • Center—Centers the frame on the page. • Inside—Aligns the frame to the side closest to the binding. On the left page of a double-sided document, the frame aligns on the right side. On the right page, the frame aligns on the left page. • Left—Left aligns the frame. • Outside—Aligns the frame to the side farthest from the binding. On the left page of a double-sided document, the frame aligns on the left side. On the right page of a double-sided document, the frame aligns to the right side. • Right—Right aligns the frame. *Note:* The alignment options only apply to the Below, Bottom, and Top anchor types. If you try to set the alignment for any other anchor type, the alignment is ignored.	Alignment in Anchored Frame properties

Table 43: AFrame tab options (continued)

Option	Description	Related FrameMaker option
Anchor Type	The type of anchored frame. The most common positions are: • Below—Below current line • Bottom—Bottom of column • Inline—At Insertion Point (alignment setting is doesn't work) • Run Into Paragraph—Run into paragraph, whatever alignment is selected • Top—Top of column, whatever alignment is selected *Note:* If you select an anchor type other than Bottom, Below, or Top, the alignment setting is ignored.	Anchoring Position in Anchored Frame properties
Base Offset	The offset from the baseline of the anchored frame. This number doesn't correspond to the numbers in the Graphic tab of an Aframe target.	Distance above Baseline in Anchored Frame properties
Side Offset	Space between anchored frame and the edge of the page.	Distance from Text Column in Anchored Frame properties
Resize Option	Changes the cell width or frame width to the size of the cell or column that contains the anchored frame. To avoid resizing, select **Declare Size.**	None

The Graphic tab

The anchored frame target's **Graphic** tab sets the parameters for how the graphic inset will be positioned on the parent anchored frame (Figure 126). You

specify the graphic name and the location of the graphic. Table 44 describes the **Graphic** tab options.

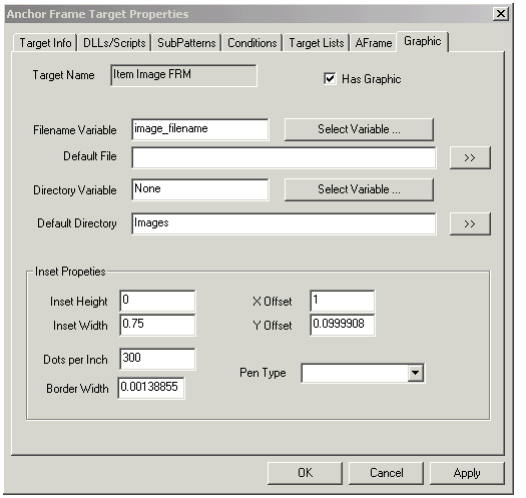


Figure 126: Graphic tab, Anchored Frame

Table 44: Graphic tab options

Option	Description	Corresponding FrameMaker Option
Has Graphic	Anchored frame contains an imported graphic.	None
Filename Variable	The name of the variable that contains the graphic filename.	None
Default File	The name of a file to use as the default graphic if the file indicated by the filename variable does not exist.	None
Directory Variable	The name of the variable that contains the subdirectory where the graphic file resides.	None

Table 44: Graphic tab options (continued)

Option	Description	Corresponding FrameMaker Option
Default Directory	The name of the directory that contains the graphic.	Import File dialog box
Inset Properties		
Inset Height	Graphic is scaled proportionally to the given width if set to 0.	Height in Object Properties
Inset Width	Width of the graphic inset.	Width in Object Properties
Dots per Inch	Number of dots per inch the graphic is imported at.	Scaling in Object Properties
Border Width	Width of graphic border.	Border Width in Object Properties
X Offset	X coordinate of the upper left corner of the graphic inset.	Offset from Left in Object Properties
Y Offset	Y coordinate of the top corner of the graphic inset.	Offset from Top in Object Properties
Pen Type	The pen to use for the borders of the anchored frame.	Pen Pattern in Tools Palette

Text Frame Target

Text frame targets generate text frames, which can be used for multiline callouts, paragraphs of text, and any other text that you want to wrap automatically from line to line. Text frames can occur only within anchored or unanchored frames, so Text Frame targets must be created as a subtarget of an Anchored Frame or Unanchored Frame target.

Text Frame targets are compound targets. The Contents target list is the only valid target list for attaching subtargets to a Text Frame target. The subtargets created in the Contents target list must be flow insertion targets.

Frame tab

Use the **Frames** tab, shown in Figure 127, to correspond to those in FrameMaker's Customize Text Frame dialog box. Table 46 describes the contents of the **Frame** tab.

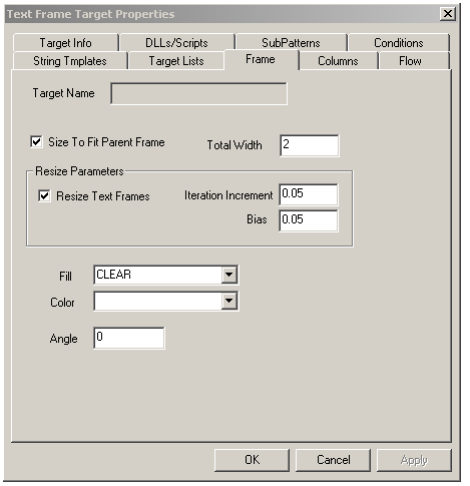


Figure 127: Columns tab, Text Frame target

Table 45: Contents of the Columns tab

Option	Description
Units	Select the units to use for measurements in this dialog box. Choose from Inches, Picas, and Points.
Size to Fit Parent Frame	Check if you want the text frame created by the Text Frame target to be the same size as the anchored frame it's attached to.
Total Width	Specify the width of the text frame here. This setting is ignored if **Size to Fit Parent Frame** is checked.

Table 45: Contents of the Columns tab (continued)

Option	Description
Resize Parameters	
Resize Text Frames	Check if you want the text frame's height to automatically adjust so all text inserted into the frame (by subtargets, subpatterns, etc.) will fit in the frame without being hidden.
Iteration Increment	Resizing the text frame is an iterative process. This parameter specifies how much the height is adjusted on each iteration.
Bias	A constant height that is added to the frame when the iteration is complete to insure that no text remains hidden.
Background color, fill and rotation angle	
Fill	The fill pattern to use for the text background.
Color	The color to use for the text background.
Angle	The angle to rotate the text frame.

Columns tab

Use the **Columns** tab, shown in Figure 128, to specify the number and appearance of text columns within the text frame. Some of the fields in this tab

correspond to those in FrameMaker's Customize Text Frame dialog box. Table 46 describes the contents of the **Columns** tab.

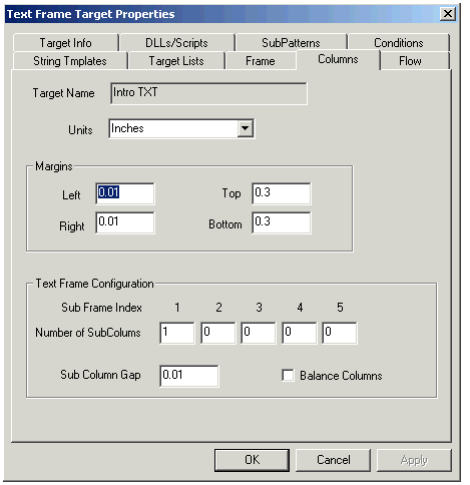


Figure 128: Columns tab, Text Frame target

Table 46: Contents of the Columns tab

Option	Description
Margins	
Left	Distance of the left side of the leftmost text frame from the left side of the parent frame.
Right	Distance of the right side of the rightmost text frame to the right side of the parent frame.
Top	Distance of each text frame to the top of the parent frame.
Bottom	Distance of each text frame to the bottom of the parent frame.
Sub Frame Configuration	
Sub Column Gap	Specify a value for the space (gap) you want to appear between subcolumns and text frames.

Table 46: Contents of the Columns tab (continued)

Option	Description
Balance Columns	Check if you want text to be balanced evenly across the columns of a text frame that isn't full of text.
Number of SubColumns	Since a Text Frame target generates an unnamed flow, you can generate up to five text frames whose flows are connected. These settings determine how many subcolumns each of the five separate text frames will have.

Flow tab

Use the Flow tab, shown in Figure 129, to specify flows in and between the columns defined in the **Columns** tab. Table 47 describes the fields and check boxes in the **Flow** tab.

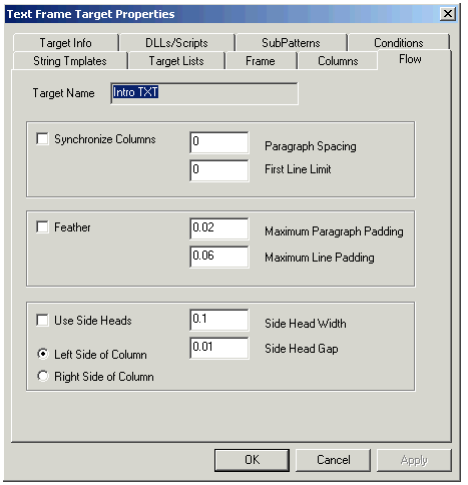


Figure 129: Flow tab, Text Frame target

Table 47: Contents of the Flow tab

Option	Description
Synchronization Parameters	
Synchronize Columns	If your text frame has multiple columns, check this box to ensure that the grid lines in adjacent columns are aligned.
Paragraph Spacing	If you selected the **Synchronize Columns** check box, specify a value for the spacing between paragraphs.
First Line Limit	If you selected the **Synchronize Columns** check box, specify a value for
Feathering Parameters	
Feather	Check if you want to vertically justify your columns so that text is aligned across the bottoms of both columns.
Maximum Paragraph Padding	If you selected the **Feather** check box, specify the maximum amount of space you want to appear between paragraphs.
Maximum Line Padding	If you selected the **Feather** check box, specify the maximum amount of space you want to appear between text lines
Side Head Parameters	
Use Side Heads	If you want your document to use side headings, select this check box. This section corresponds to options found in FrameMaker's Column Layout dialog box.
Side Head Width	If you selected the **Use Side Heads** check box, specify a value for the side heading width here.
Side Head Gap	If you selected the **Use Side Heads** check box, specify a value for the gap between the side heading area and the body-text area here.

Table 47: Contents of the Flow tab (continued)

Option	Description
Left Side of Column/Right Side of Column	Select one of these radio buttons to specify the side of the page on which you want to place side headings.

Chapter 13: Execution control

Several targets do not directly generate content, nor do they deal with formats or layout. These targets can be divided into two groups:

- Execution control

These targets behave analogous to the structured programming constructs of a standard programming language. Table 48 describes the execution control targets.

Table 48: Execution control targets

Target name	Description	For details, see...
Blank 	Provides a placeholder for attaching a subpattern directly below another pattern without any intermediate output.	"Blank target" on page 240
If-Else 	Provides an IF-ELSE control structure in the form of two target lists, IF and ELSE.	"If-Else target" on page 240
Case 	Lets you attach a string variable to a target list so that target list is executed when the value of the variable string is the same as the name of one of its target lists.	"Case target" on page 241
Pattern Set 	Allows you to run another pattern set from the current one. You can pass parameter values to this pattern set from the calling pattern set.	"Pattern Set target" on page 242
Call 	Part of the mechanism for placing a single pattern multiple places in the pattern set hierarchy.	"Call target" on page 245
Target Set 	Provides a way to organize target lists in a single target. A condition attached to the Target Set target applies to all target lists in that target.	"Target Set target" on page 240

Blank target

A fundamental rule in building a PSet is that subpatterns cannot be directly attached to another pattern—there must be a target in between the pattern and its subpattern. Blank (BLANK) targets provide a way to sequence one or more subpatterns in the PSet hierarchy without generating any intermediate output. For example, if the body rows of a table are generated from a subpattern instead of directly from a pattern/query, you would need to attach the subpattern to an intermediate target. The Blank target fills this role.

Note: Refer to the EncompassA.pst demo to see how the Blank target is used to set up the contact information section.

The list of valid child targets for any subpattern attached to the Blank target is derived from the hierarchy above the Blank target, not from the target itself.

Target Set target

Target Set (SET) targets provide a way to group targets. They are analogous to compound statements in a standard programming language (such as the {} brackets in C). Target Set targets share all of the properties of a Blank target, with the addition of a target list called Set. The list of valid targets you can attach to the Set target list is determined by what lies above the Target Set target in the PSet hierarchy.

Target Set targets are useful for grouping targets together that logically form a block. For example, if the same conditions need to be applied to each target in the block, then applying these conditions to the parent Target Set target would be faster and less prone to error than having to apply the same conditions to each target.

If-Else target

The If-Else (IF ELSE) target creates a logical if-else structure through its two available target lists—If and Else. When a condition is attached to an If-Else target, the targets in the If list are invoked if the condition evaluates to True, and the targets in the Else list are invoked if the condition evaluates to False. If more than one condition is attached to the target, the rules for a logical AND apply—the targets in the If list are invoked only if *all* of the conditions evaluate to True. If even one of the conditions evaluates to False, then the targets in the Else list are invoked.

If there are no targets in the Else list, the If-Else target acts exactly like a Target Set target.

Case target

Case (CASE) targets let you set up more complex flow control switching than the If-Else target. To use the Case target, attach a String variable to the target. You then create your own target lists for the Case target—the target lists should match possible values for the attached variable.

Note: To create your own custom target list, go to the **Target Lists** tab and select the **Edit** button. The Edit Target Lists dialog box is displayed (Figure 130 on page 241). Under Selected Target List, select **DEFAULT** (the word, not the arrow), type in the name of your target list, then select the **Add List** button.

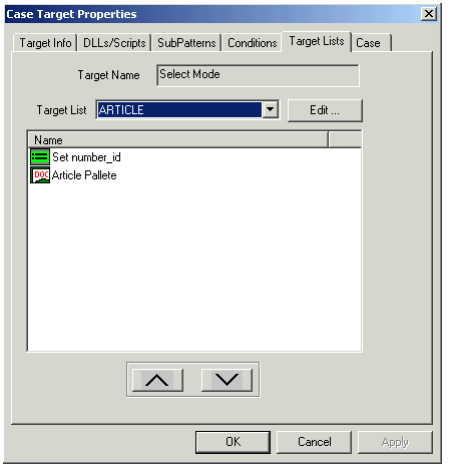


Figure 130: Edit Target Lists dialog box

When the Case target is invoked, only the targets in the target list whose name matches the current value of the variable will be invoked. If you add targets to the target's Default target list, those targets will be invoked only when none of the other custom target lists matches the current value of the variable.

Note: If you do not supply a case variable, only targets in the Default list will be invoked.

Use the **Case** tab, shown in Figure 131, to select a string variable whose current value controls the switching of the Case target. Select a variable from the drop-down list.

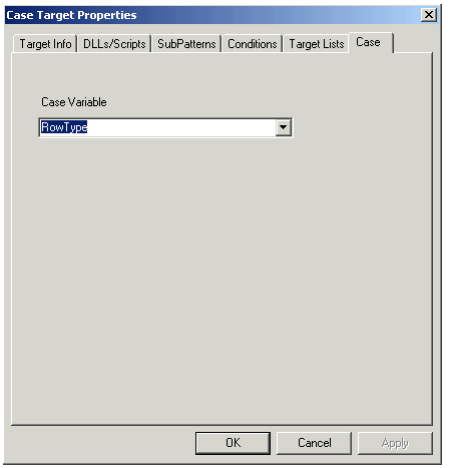


Figure 131: Case tab, Case target

Pattern Set target

Pattern Set (PSET) targets run another PSet file. When a Pattern Set target is invoked, the following occurs:

1. The child PSet file (specified in the **Pset File** tab of the Pattern Set Target Properties dialog box—see "Pset File tab" on page 243) is loaded into memory. The current PSet is suspended.
2. The run state of the current session—the current book and document status—is passed to the loaded PSet file so it can create documents and add corresponding book components to the parent PSet's book. The child PSet can also insert items into the current document.
3. The loaded PSet file is run. Once the child PSet has finished running, the parent PSet resumes running.

   In some cases, the child PSet may not contain standard pattern set elements, such as Document or Page targets. For example, if the Pattern Set target is placed in a subpattern whose parent is a Table target, the child PSet file could start by generating rows using a Row target. A nonstandard child PSet file will generate an error if you try to run it independently of its parent PSet file.

   Note: The target at the top of the of the child PSet's hierarchy must be valid for the location of the Pattern Set target in the parent PSet.

To pass values from the parent PSet file to the child, create variable assignments using the **Variable Assignments tab** of the Pattern Set Target Properties dialog box (see “Variable Assignments tab” on page 244).

Formatting the Pattern Set target

The Pattern Set Target Properties dialog box includes two tabs unique to Pattern Set targets—the **Pset File** tab and the **Variable Assignments** tab.

Pset File tab Use the **Pset File** tab, shown in Figure 132, to specify the name and location of the child PSet file. Table 49 describes the contents of the tab.

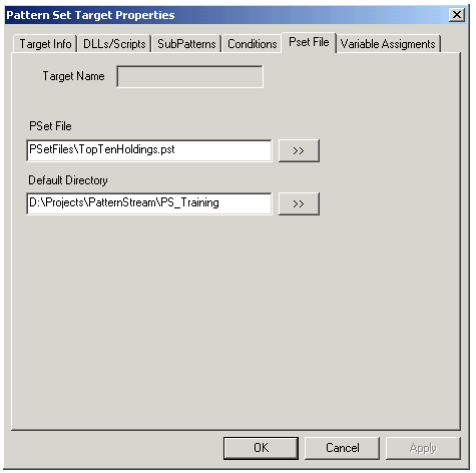


Figure 132: Pset File tab, Pattern Set target

Table 49: Contents of the PSet File tab

Option	Description
PSet File	The name of the PSet file to run. Select the >> button to browse for the file.
Default Directory	The name of the directory that contains the PSet file. Not necessary if the full pathname is specified in the PSet File field. Select the >> button to browse for the directory.

Variable Assignments tab

Use the **Variable Assignments** tab, shown in Figure 133, to specify how variables are passed from the parent PSet file to the child. Table 50 describes the contents of the tab.

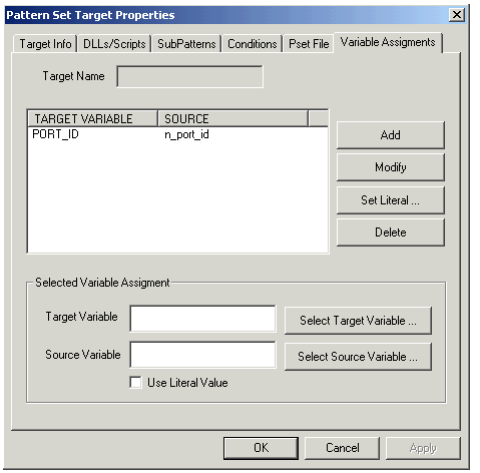


Figure 133: Variable Assignments tab, Pattern Set target

Table 50: Contents of the Variable Assignments tab

Option	Description
Add button	After you configure information in the Selected Variable Assignment area, select this button to create a new assignment and add it to the list.
Modify button	To modify an existing assignment, select an assignment pair from the list to display their parameters in the Variable Assignment area. Make your changes then select this button to save the modifications.
Delete button	To delete an assignment, select an assignment pair from the list then select this button.

Table 50: Contents of the Variable Assignments tab (continued)

Option	Description
Selected variable assignments	
Target Variable	Name of the variable in the child PSet file to assign a value to. (The variable must also be defined in the parent PSet.) Select the **Select Target Variable** button to select a variable from the list.
Source Variable	Name of the variable in the parent PSet file whose value will be assigned to the target variable. Select the **Select Source Variable** button to select a variable from the list.
Constant value	If you want to pass a constant value to the child PSet, use this section to specify the value (do not specify a variable in the Source Variable section). Select the value's type (String, Integer, or Float), then enter the value in the corresponding field.

Call target

The Call (CALL) target provides a way to use the same pattern in more than one place in the PSet hierarchy. With a Call target, the subpattern is not attached to the target itself; instead, the subpattern is attached to a PSet Tree extension object, which is in turn attached to the Call target. Unlike patterns and targets, extension objects can be used in multiple places in the PSet hierarchy. When a Call target is invoked, the PSet Tree extension is referenced, and the associated subpattern is executed as if it were actually attached to the target. You must use a Call target in each location you want to run the subpattern.

Note: You must insure that the targets invoked by the called subpattern are valid for the Call target's location in the PSet hierarchy.

Use the **Call** tab, shown in Figure 134, to attach a PSet Tree extension object to the Call target. Select the **Attach PSet Tree** button and select an extension from the list.

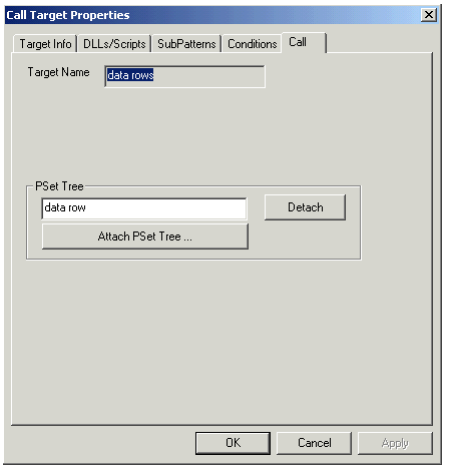


Figure 134: Call tab, Call target

Chapter 14: Data manipulation targets

Several targets do not directly generate content, nor do they deal with formats or layout. These targets take the place of simple data assignment statements in normal programming. The targets can be inserted anywhere in the PSet hierarchy. Table 51 describes the data manipulation target.

Table 51: Data manipulation targets

Target name	Description	For details, see...
Assignment 	Takes the current value of a source variable and applies it to a target variable (X := Y).	"Assignment target" on page 247
Empty 	Acquires data without generating content.	"Empty target" on page 249
Lookup 	Lets you store data in an associative or indexed array.	"Lookup target" on page 250

Assignment target

Some composition problems require that you store the contents of one variable in another variable, as you would in a standard procedural programming language. Assignment (ASSIGNMENT) targets are used for this purpose.

The Assignment target takes the current value of one variable (the source) and applies it to another variable (the target). For example, your query returns entries of different data types, and you want something to happen every time the type changes. You can store the previous value in a variable called PREV_TYPE and use the Assignment target to initialize the variable. The database returns the type value in a variable called CURRENT_TYPE, and you can attach a condition that is true when CURRENT_TYPE <> PREV_TYPE. At the end of each loop you use an assignment to set PREV_TYPE to CURRENT_TYPE, where PREV_TYPE is the target and CURRENT_TYPE is the source.

Formatting the Assignment target

The Assignment Target Properties dialog box includes one tab unique to Assignment targets—the **Assignment** tab.

Use the **Assignment** tab, shown in Figure 135, to specify the source and target variables, the target's literal value, and a variable transform. Table 52 describes the contents of the **Assignment** tab.

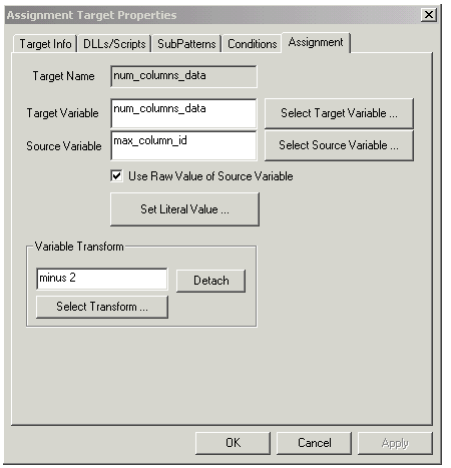


Figure 135: Assignment tab, Assignment target

Table 52: Contents of the Assignment tab

Option	Description
Target Variable	The name of the variable to assign a value to. Select the **Select Target Variable** button and select a variable from the list.
Source Variable	The name of the variable to get the value from. Select the **Select Source Variable** button and select a variable from the list.
Use Raw Value of Source Variable	

Table 52: Contents of the Assignment tab (continued)

Option	Description
Set Literal Value	To assign a constant value to the target variable, select this button. The Edit Value dialog box is displayed (Figure 136). In the dialog box, select the literal value's type then specify the value in the corresponding field(s).
Variable Transform	
Select Transform	Select this button to select a transform from the Select Transform dialog box. Only transforms valid for the selected target variable will be shown (see Chapter 17, "Transforms").
Detach	Detaches the current transform.

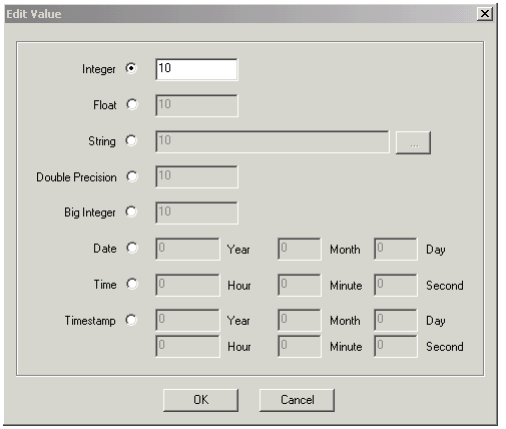


Figure 136: Edit Value dialog box

Empty target

Empty (EMPTY) targets are like Blank targets in that they provide a handle to which you can attach sub-patterns. However, you can use an Empty target only in cases where you need to acquire data for use in other targets without generating output.

For example, if you want to determine how many rows a query will return before the pattern associated with the query is executed, you can run a query derived from the original query that returns the number of rows (SELECT count (*)…). The derived query is assigned to a pattern, and this pattern is attached to an Empty target.

The use of an Empty target signifies that the subtree attached to the target generates no output; therefore, the subtree attached to an Empty target cannot contain output-generating targets. Only the following targets can be children of an Empty target:

- Assignment
- Lookup
- Empty
- Target Set
- Case
- If-Else
- Blank
- Call (associated subpattern cannot generate any output)
- Pattern Set (associated PSet file cannot invoke an output-generating target).

Lookup target

Lookup targets allow you to insert entries into a lookup table or associative array that can be used in a different part of the output sequence. These arrays are always single valued but can have multi-valued keys. The lookup target uses three associated objects to determine the entry it will insert:

- Variable
  Called the result variable, it contains the values of the array.
- Lookup Table
  An extension object that is the array itself (see "Lookup Table" on page 353).
- Schema
  An Argument List parameter list (see "Defining an Argument List" on page 345) that contains a list of variables that create the lookup for each value in the array.

For example, suppose you have a pattern (with its associated query) where three variables are updated for each loop in the pattern. The three variables that are bound to the columns in the query's SELECT statement are VAR_A, VAR_B, and VAR_C. Every time a pattern loops, these variables have different values. By attaching these variables to an Argument List and using the Argument List in the Lookup target, you can have columns 1 and 2 return the lookup key through the

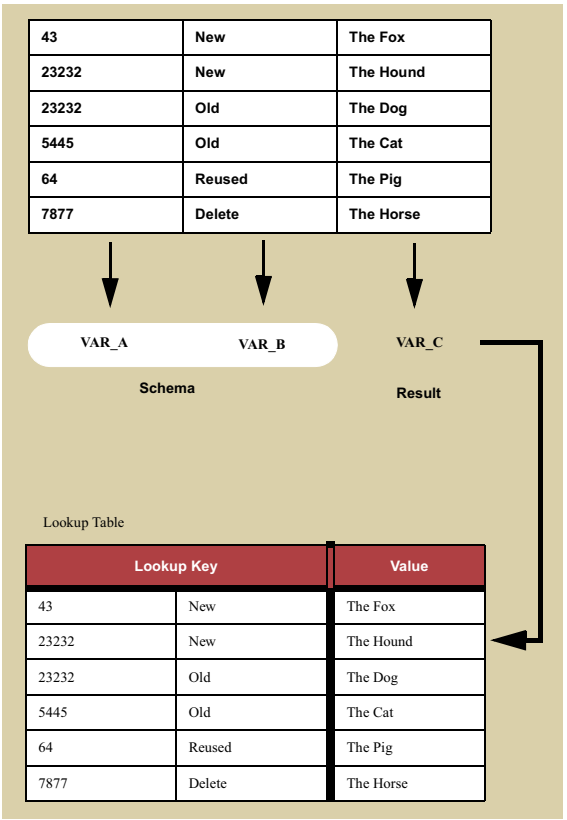
current values of VAR_A and VAR_B. You can then use VAR_C to obtain the value of the result. Every time the Lookup target is invoked, it inserts a new entry into the array based on the values in VAR_A, VAR_B, and VAR_C.

This sets up a functional relation between VAR_A, VAR_B and VAR_C.

VAR_C = F(VAR_A, VAR_B).

To access the values in the Lookup Table (or array), you use transforms with the Lookup Table Directive (see Chapter 17, “Transforms”). Example 2 shows how the Lookup Table works.

Example 2: How the Lookup Target works



During each pattern cycle, the values returned from the database are stored in variables VAR_A, VAR_B and VAR_C.

VAR_A and VAR_B are used to create the key, and VAR_C is used for the value.

Formatting the Lookup target

The Lookup Target Properties dialog box includes one tab unique to Lookup targets—the **Array** tab.

Use the **Array** tab, shown in Figure 135, to specify the Result variable, Lookup Table extension, and Argument List parameter list. Table 52 describes the contents of the **Array** tab.

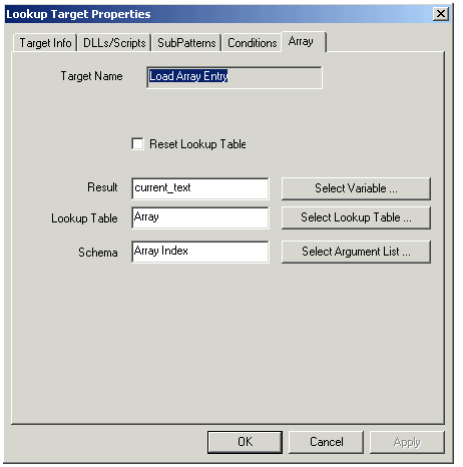


Figure 137: Array tab, Lookup target

Table 53: Contents of the Array tab

Option	Description
Reset Lookup Table	
Result	Select the **Select Variable** button to select the Result variable from the Select Variable dialog box.
Lookup Table	Select the **Select Lookup Table** button to select the Lookup Table extension from the Select Extension dialog box.
Schema	Select the **Select Argument List** button to select the Argument List parameter list from the Select Parameter List dialog box.

Chapter 15: String templates

The PatternStream application uses *string templates* to create text content. String templates are phrase- or sentence-level objects composed of small building blocks that specify how to construct text from data extracted from an external data source. These text elements might include current data values expressed as strings, any characters that surround the data (such as hyphens or parentheses), any FrameMaker markers you want to insert (such as index or cross-reference markers), and frame anchors for imported graphics.

String templates are composed of smaller building blocks called *segments*. Each segment generates one unit of text, such as a variable or constant string, a marker or cross-reference, and so on. Most segments contain the following:

- An attached variable that controls the output. This variable could contain the actual text, the file name for imported graphics or text insets, or the text content of a marker or cross-reference.
- A character format that is applied (when appropriate) to the content created by a particular segment.
- An attached *condition* (when invoking the segment is data-driven).

This chapter starts by describing how to create and modify string templates. For an overview of the various segment types, see Table 54 on page 262.

String template basics

Example 3 illustrates how a string template builds an instance of text. The desired text is a person's full name, which is derived from four database fields, the lastname, the firstname, the prefix and the suffix. To each data field there corresponds a PatternStream variable that stores the retrieved data value. Every time the string template is called upon to generate some text, the values of each variable will be different, and a different text string, in this a case a person's name, will be created.

Example 3: Structure of a string template

In this example, a person's name is created from various data elements. The name itself is stored in the database in four fields—last name, first name, prefix, and suffix. A typical name is

Prof. John Doe, Ph.D.

Example 3: Structure of a string template (continued)

Broken down into segments, it would look like this

Prof.		John		Doe	,	Ph.d
a	b	c	d	e	f	g

Each segment has been labeled **a** though **g** and is described briefly below:

a Variable String segment that has the *prefix* variable attached.

b Constant String segment for the space between the prefix and the first name. To avoid inserting an extra space if there is no prefix, a condition is attached to the segment; the condition is true only if the prefix is not blank (if the condition is false, the segment is not generated).

c Variable String segment that has the *firstname* variable attached.

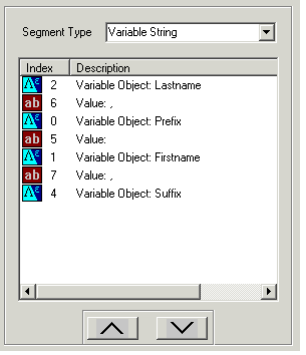
d Constant String segment for the space between the first and last names with a possible condition if entries without a first name are possible.

e Variable String segment that has the *lastname* variable attached.

f Constant string segment for the comma and space between the last name and the suffix. An attached condition is true only if a suffix exists.

g Variable string segment that has the *suffix* variable attached.

After you define these segments, the string template definition is displayed as follows in the **String Templates** tab:



Note: Don't worry about the index numbers on each segment. They are for reference purposes only. The segments are invoked in the orderly they appear in the list.

Example 3: Structure of a string template (continued)

This string template is attached to the appropriate Paragraph or Table Cell target to create content. Typically, the text the string template generates will be different for each invocation, because the values of the variable segments and the evaluation of the conditions will be different each time.

Here are some tips to help you construct string templates:

- To insert data from a database, use the Variable String segment type.
- To insert graphics, use the Graphic segment type.
- To insert tabs, thin spaces, or other special characters, use a Constant String segment type.
- String templates can be used as many times as necessary in a pattern set. You can also attach more than one string template to a content target, so breaking a string into reusable chunks might be appropriate.
- A string template does not have to contain any variable data. It can be composed entirely of constant strings. However, these constant string segments could have conditions attached that are data-driven, and hence, the final output string could still depend on the data.

Creating string templates

To create a new string template, follow these steps:

1 Select the **String Templates** tab from the PatternStream main dialog (Figure 138).

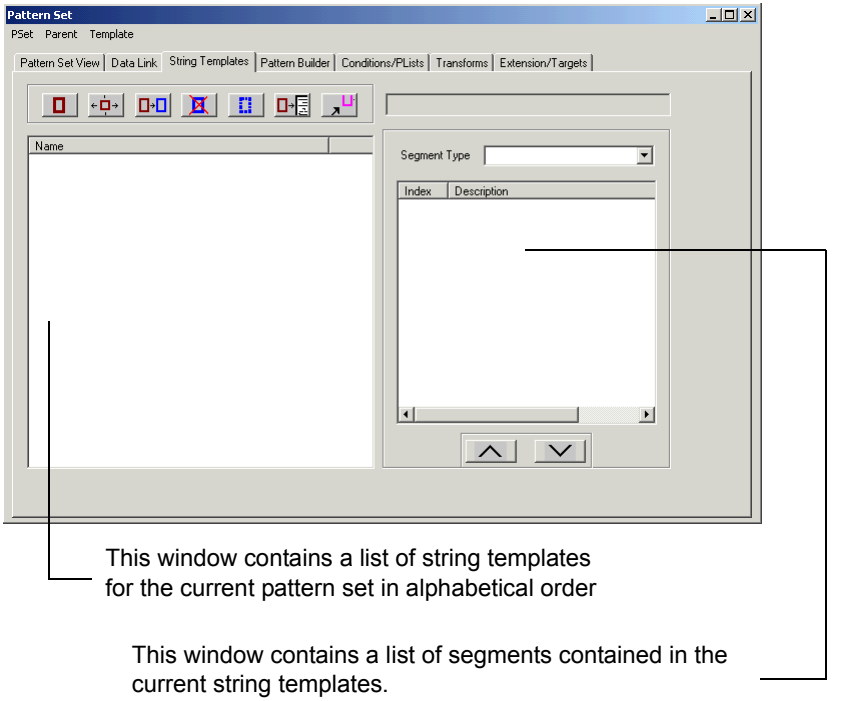


Figure 138: String Template tab

2 Select the **Parent** menu, then select **New Object.** The Create New String Template Object dialog box is displayed (Figure 139).

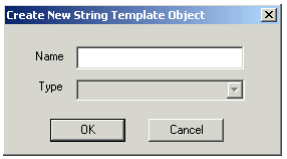


Figure 139: Create New String Template Object dialog box

3 In the Name field, type a name for the new string template.

4 Select **OK**. The new string template and its icon are displayed in the string template list. String templates are listed in alphabetical order. The new string template is now the current string template.

After you create a string template, you define its contents by adding segments. See “Inserting segments” on page 262 for details.

Working with string templates

To work with a string template, you must select it in the **String Templates** tab of the **PatternStream** main dialog. On the left side of the tab, existing string templates for the current pattern set are displayed in alphabetical order. By clicking on the string template’s icon, you make it the current string template and its name appears in the upper right of the dialog. More importantly, the segments that makeup the string template are displayed in the list on the right side of the dialog (Figure 140).

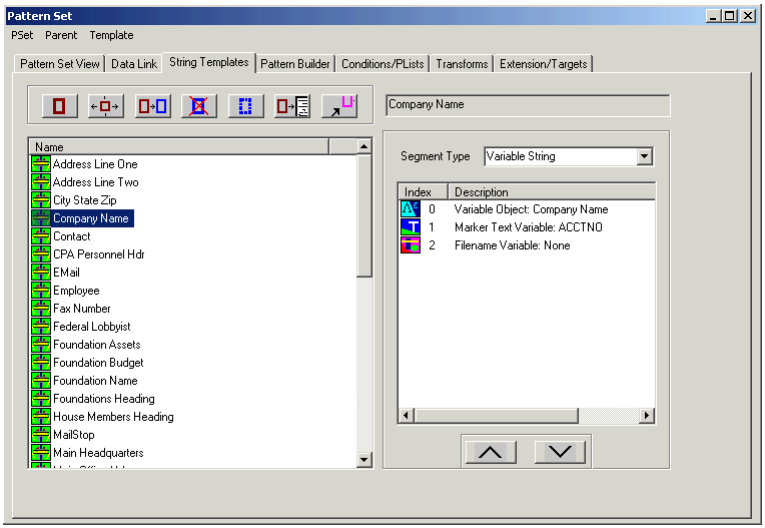


Figure 140: String template and associated segments

Modifying string template name and description

The segments that make up a string template contain most of the string template properties, such as the string value, character format, condition, and so on. Two properties—the name and description—are modified through the PatternStream application **Parent** menu.

To modify the name and description, follow these steps:

1 From the **Parent** menu, then select **Object Info.** The **Edit String Template Properties** dialog box is displayed (Figure 141).

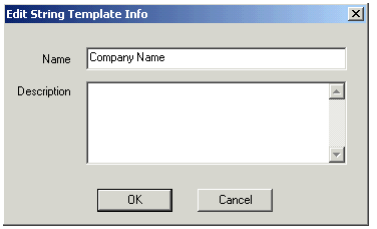


Figure 141: Editing the string template name and description

2 To change the string template name, type the new name.

3 You can also enter a description that will appear when the object's properties are printed out.

4 Select **OK.** The changes are then incorporated into the current string template.

Finding references

Do one of the following to view the a list of objects that use a particular string template:

- Select the **String Templates** tab, select the **Parent** menu, then select **References.** The Reference Count Objects dialog box is displayed (Figure 142). If the string template is attached to an object, the object is displayed.

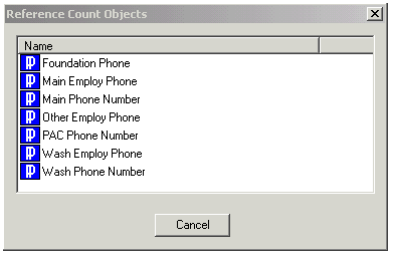


Figure 142: Objects that references the current string template

- Select the **Pattern Set View** tab. At the bottom of the tab, the string templates are listed in their own folder (Figure 143). Right-click the string

template and select **References** from the pop-up menu. The Reference Count Object dialog box is displayed (Figure 142).

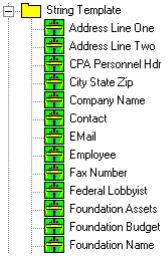


Figure 143: String templates listed in Pattern Set View tab

Copying string templates

Rather than building a new string template from scratch, sometimes it is easier to first make a copy of a similar one that already exists, then use this copy as the starting point for the new string template.

To copy a string template, follow these steps:

1 Select **String Templates** tab, then select the string template you want to copy.

2 Select the **Parent** menu, then select **Copy.** The Copy String Template dialog box is displayed (Figure 144).

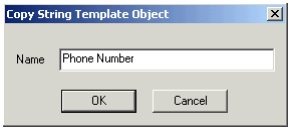


Figure 144: Copying a string template

3 Type the name of the new string template and select **OK.** The new copied string template becomes the current string template.

Note: Two string templates cannot have the same name, so you must change the name in the copy object dialog.

You can also copy string templates from the **Pattern Set View** tab. Right-click the string template, then select **Copy** from the pop-up menu.

Exporting string templates

In the PatternStream application, you export objects to use them in other pattern set files. To export the current string template to the export buffer, select

the **Parent** menu, then select **Export.** You can also export string templates from the Pattern Set View. Right-click the string template in the Pattern Set View and select **Export** from the pop-up menu.

Printing string template attributes

You can print string template properties by selecting **Print Object** from the **Parent** menu. A FrameMaker document is created listing the attributes of segments that make up the current string template. You can also print the attributes of all of the string templates in the current pattern set by selecting **Print Class** from the **Parent** menu.

String templates and target objects

String templates are attached to target objects (see "Kinds of targets" on page 128 for an explanation of text vs. non-text targets) that generate structural elements which can contain text. These include paragraphs, table cells, text frames and markers. Before you can use a string template in a target, you must be able to access the particular target's properties dialog. This can be done by double-clicking on the target's icon in:

- the **Pattern Builder** tab after making the target's parent pattern the current pattern.
- the **Target List** tab of the parent target's properties dialog.
- the pset hierarchy in the **Pattern View** tab.
- the target list window on the Extension/Targets tab of the **PatternStream** main dialog.

After the target's properties dialog is displayed, select the **String Templates** tab (Figure 145).

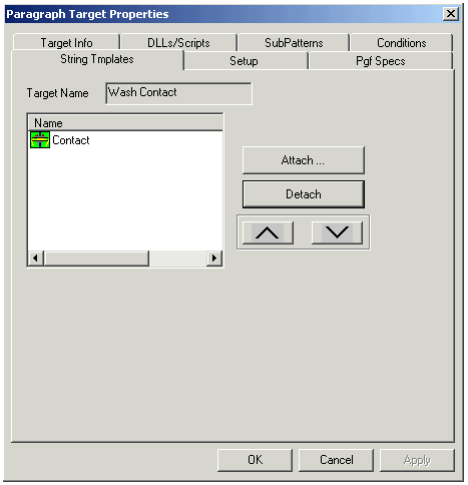


Figure 145: The String Templates tab of the target properties dialog.

Attaching a string template to a target.

1 To attach a string template, click on the **Attach** button. This displays the **String Template** select dialog.

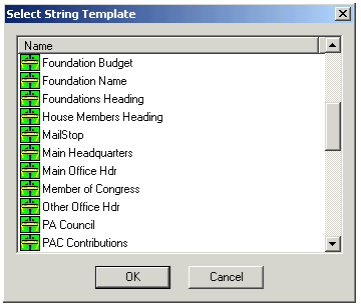


Figure 146: Attaching and detaching a string template

2 Select the desired string template, then click on the **OK** button. It now appears in the list of attached string templates for the current target.

Note: You can attach a particular string template to more than one target.

Detaching a string template from a target

To detach a string template from the current target, select the string template you wish to remove, then click on the **Detach** button.

Inserting segments

A string template is composed of one or more *segments.* Segments generate formatted text, anchored frames and graphics, index markers, FrameMaker variables, and more. Table 54 describes the different types of segments:

Table 54: Segment types

Type	Use to insert	Description	For details see...
Variable String 	A value stored in a PatternStream variable.	Generates a formatted string from one data element.	"Variable String segment" on page 267
Constant String 	Constant text	Specifies headings, characters (such as commas and parentheses surrounding a variable string), and special formatting (such as hard returns).	"Constant String Segment" on page 268
Graphic 	An anchored frame and graphic	You must specify the size and location of the frame and graphic.	"Graphic segment" on page 269
String Template 	Text generated by another string template.	Allows you to attach one string template to another, creating a more complex, hierarchical structure.	"Defining an Index segment" on page 277
Index 	An index marker	Generate the syntax that will be inserted into an index marker.	"Defining an Index segment" on page 277

Table 54: Segment types (continued)

Type	Use to insert	Description	For details see...
X Reference 	A FrameMaker cross-reference	Inserts cross-reference information from the current document or an external one.	"Defining a Cross-Reference segment" on page 281
Marker 	A marker of any type	Inserts a marker (cross-reference, conditional text, or custom) into the document.	"Defining a Marker segment" on page 283
Text MarkUP 	A specialized segment that parses markup within a text string.	Lets you interpret markup that is embedded in text data and apply appropriate formatting. Used in conjunction with the Segment Transform.	"Defining a Text Markup segment" on page 284
Flow 	Parts of other FrameMaker documents as a text inset	You can specify that the main flow or named flows on another Frame document's reference page to be imported into the current document. The name of the source document can be derived from the database.	"Defining a Flow segment" on page 286
Variable Format 	A FrameMaker variable	Inserts a FrameMaker variable.	"Defining a Variable Format segment" on page 288
Text Inset 	Imported text from other applications (i.e.,. Microsoft Word).	Inserts text from another application, such as a word processing program, as a text inset.	"Defining a Text Inset segment" on page 289
New Paragraph 	New paragraph with a specified format.	Creates a new paragraph with the selected format. The name of the paragraph format can be data-driven.	"Defining a New Paragraph segment" on page 290

To insert segments into your string template, follow these steps:

1 From the **String Templates** tab, select the string template you want to define.

Note: You can also select the string template from the **Pattern View** tab, right-click, and select **Properties** from the pop-up menu. The **String Templates** tab is displayed.

2 Select a segment type from the Segment Type drop-down list. Table 54 describes the available segment types.

3 Right-click anywhere in the Segments area, then select **Insert** from the pop-up menu. The new segment and its icon are added to the segment list. Segments are invoked in the order they are listed. (To change a segment's place in the list, select the segment, then select the up or down arrow.)

Note: When you insert a segment, the PatternStream application assigns it an Index number, which helps the system track the segments being used. The index number is arbitrary and does not indicate order. Index numbers are used when modifying pattern set attributes with the PatternStream Runtime Library.

Defining common segment properties

Segments have certain things in common. They almost always have a variable associated with them, they mostly have an associated character format and all segments can have a condition attached to them. After you create a segment, you must define its properties by right-clicking the segment's icon and selecting **Properties** from the pop-up menu. The Segment Properties dialog box specific to the segment type is displayed.

To modify the segment for a string template, follow these steps:

1 Select the segment's icon, right-click, then select **Properties** from the pop-up menu. The Segment Properties dialog appropriate for that type of segment will appear.

2 You can attach a variable by doing one of the following:

- In the Variable field, specify the variable you want to insert by typing the variable's name.

- Select the **Select Variable** button and select the variable name from the **Select Variable** dialog box. This method is preferable because you don't have to worry about spelling the name correctly.

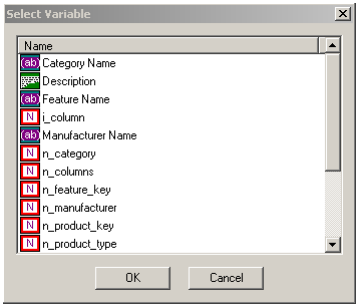


Figure 147: Select Variable dialog.

See Chapter 5, "Variables" for details on variables.

3 To specify a character format for the current segment (if appropriate for that segment type), select a character format from the Character Format drop-down list. The formats displayed are those defined in the FrameMaker document template attached to the current pattern set.

Note: If the expected character formats are not listed, you may need to synchronize your FrameMaker template with the PatternStream application. Select the **Template** menu from the PatternStream main dialog, then **Extract Formats.**

Note: When a pattern set uses more than one document target, each using a different FrameMaker template file, then the named formats used in the string template must be valid for the output document that the text is being inserted into. The formats displayed in the various drop-down lists are derived from the current document template. This is the ancestor document target of the last visited target object in the pset hierarchy. To make another document target the current one, just find in the pset hierarchy and select it.

4 To attach a condition to the segment, select the **Attach Condition** button, select the condition you want from the Select Condition dialog box, then select

OK. (To detach a condition from the segment, select the **Detach Condition** button.) See Chapter 16, “Creating conditions” for details about conditions.

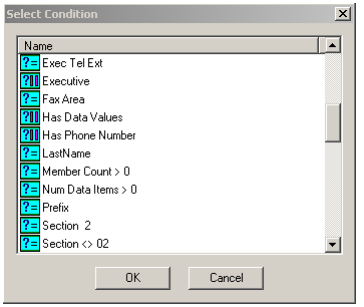


Figure 148: Select Condition dialog.

Note: When you attach a condition to a segment, that segment will no longer generate any output unless the condition evaluates to TRUE when the segment is invoked.

5 Select **OK** to save the changes.

6 Select **Cancel** to revert back to the previous settings.

Some segments allow you to attach another string template. For example, you may wish to use a string template to generate the syntax for a hypertext marker. Table 55 describes why you would want to attach a string template to a segment.

Table 55: Segments to which you can attach string templates

Segment type	Attaching string template…
String template	Creates hierarchical, complex string templates.
Marker	Generates the marker text from a string template rather than a variable. This would allow you to create hypertext syntax that is partially data-driven.
Index	Generates the syntax of an index marker from the string template rather than the assigned parameter values entered through the **Index Segment Properties** dialog.
XRef	Generates the cross-reference from a string template instead of a simple variable.

Using Segments

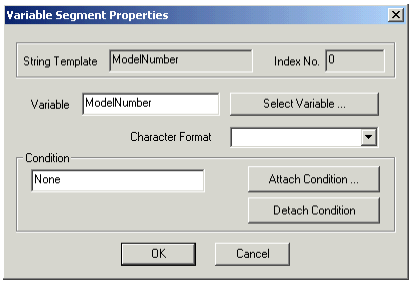


Figure 149: Variable Segment Properties dialog box

Table 56: Contents of the Variable Segment Properties dialog box

Option	Description
Variable	Type in the name of the variable directly.
Select Variable	Check this button to choose a variable from the Select Variable dialog box.
Character Format	Select a character format from the drop-down list. Available formats are determined by the template attached to the parent Document target. This format will then be applied to all of the text generated by this segment.
Condition	To attach an optional condition to the segment, select the **Attach Condition** button and choose a condition from the Select Condition dialog box. To detach the condition from the segment, select the **Detach Condition** button.

The following sections describe how to configure each type of segment.

Variable String segment

A *variable string segment* is typically used as a placeholder for data items retrieved from the database. In this type of segment, you specify the variable associated with the data you want to insert. Each Variable String segment can contain only one variable. The actual string created is a combination of the current value of the variable, the application of a transform (if any) attached to the variable, and the application of the format object used by the variable.

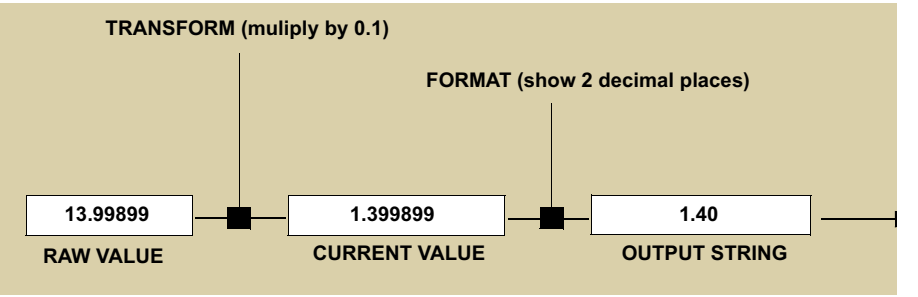


Figure 150: How Variable String segments create a text string.

Constant String Segment

A Constant String segment is a string of text not dependent on the database. You would use constant strings for headings, text or symbols around variables (such as the parentheses around an area code), and special formatting characters (such as tabs, m-dashes, hard spaces and soft returns, etc.).

The Constant String Segment Properties dialog box is shown in Figure 151. Table 57 describes its contents.

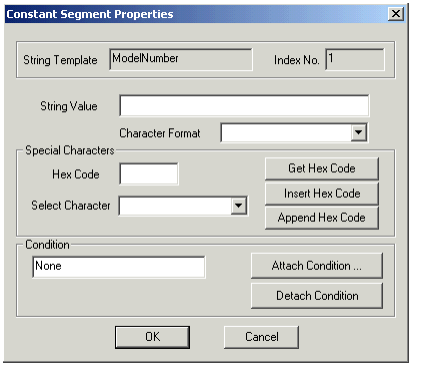


Figure 151: Constant Segment Properties dialog box

Table 57: Contents of the Constant Segment Properties dialog box

Option	Description
String Value	Type the text you want in the field. Refer to the fields under "Special Characters" for information on inserting special characters into the string.

Table 57: Contents of the Constant Segment Properties dialog box (continued)

Option	Description
	Sometimes the amount of constant text is quite large and using a normal edit control is inconvenient. Clicking on this button brings a larger, scrolling text window where you can enter large amounts of text.
Special Characters	
Hex Code	If the special character you want to insert into your text string is not available in the Select Character drop-down list, type the character's hex code in this field. (A chart containing character hex codes is available in FrameMaker's online help. In FrameMaker 5.5.6, select the **Help** menu, **Online Manuals,** and then **FrameMaker Character Sets.** In FrameMaker 6.0, look in the Online Manuals directory.)
Select Character	Select the special character from the drop-down list. Its hex code is automatically displayed in the Hex Code field.
Get Hex Code button	To find out the hex code value of a character, select the symbol in the string value field (usually appears as a vertical bar) and select the **Get Hex Code** button. The hex code value of the selected character appears in the Hex Code field. Hex codes do not display well in the text edit field, so don't worry if the display is distorted. You can verify the code by running the pattern set and viewing the output.
Insert Hex Code button	To insert the hex code into the current string value, position your cursor in the String Value field where you want the character to appear then select this button. *Note:* The symbol that is displayed may not match the symbol you are inserting.
Append Hex Code button	To append the hex code to the end of the current string value, select this button. *Note:* The symbol that is displayed may not match the symbol you are inserting.

Graphic segment

Use a Graphic segment to insert a graphic inside an anchored frame. When you define a Graphic segment, you specify the characteristics of the anchored frame

(such as alignment and offset), the variable that refers to the graphic's file name in the database, and the graphic's size and position. You can also apply conditions to graphic segments.

Graphic segments import images by first creating an anchored frame, then creating a graphic file inset within the anchored frame. You can control both the size and type of the anchored frame, as well as its other attributes like color and fill type (see Table 60 on page 276 for details). For the inset, you specify either the inset's explicit size (by typing in the size or by specifying the variables that contain the inset's height and width) or its sizing/scaling criteria. You also position the graphic inset that will contain the image relative to this anchored frame.

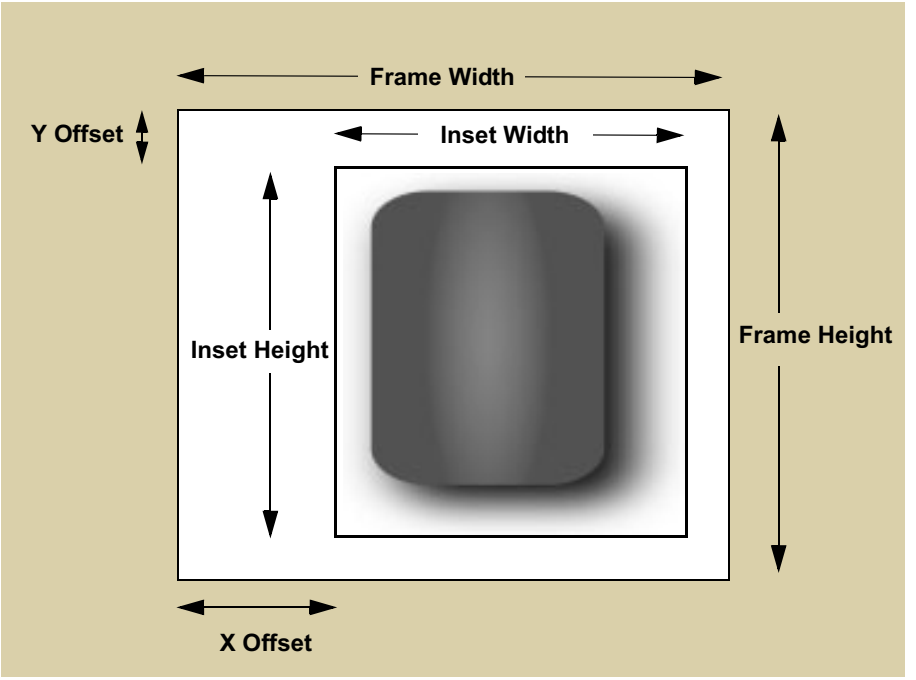


Figure 152: Dimensions and position of the graphic inset and the anchored frame

When you import a graphic image, you can specify various resizing options. Unlike anchored frame targets, these options explicitly refer to the final dimensions of the graphic inset rather than the enclosing anchored frame. The size of the anchored frame, however, can be slaved to the final size of the graphic inset.

If the anchored frame dimensions are set equal to zero, then the actual dimensions of the anchored frame will be

$$FRAMEWIDTH = INSETWIDTH \times 2 \times XOFFSET$$

$$FRAMEHEIGHT = INSETHEIGHT \times 2 \times YOFFSET$$

The following table lists some possible combinations of the resize options, the graphic inset dimensions and the result

Table 58:

Resize Option	Behavior
Adjust Cell Width	• Inset dimensions are determined by the image dimensions. • Frame dimensions will be determined by the formulae: $FrameWidth = InsetWidth + (2 \times XOffset)$ $FrameHeight = InsetHeight + (2 \times YOffset)$ • If the frame is a table cell, then the column width will be adjusted to the inset width. • Ignores the frame dimensions.
Adjust Graphic	• Adjust the frame width to the width of the containing table cell or text column. • Inset dimensions are determined according to the formula: $InsetWidth = FrameWidth - (2 \times XOffset)$ • Height of the inset is scaled to keep the image in proportion. • Frame height is determined by the formula: $FrameHeight = InsetHeight + (2 \times YOffset)$

Table 58:

Resize Option	Behavior
Import As Is	• Inset dimensions are determined by the image dimensions. • Frame dimensions are what is declared in the target attributes. • If the frame width = 0, then the frame width is determined by the formula: $FrameWidth = InsetWidth + (2 \times XOffset)$ • If the frame height = 0, then the frame height is determined by the formula: $FrameHeight = InsetHeight + (2 \times YOffset)$
Declare Size	• Frame dimensions are what is declared in the target attributes. • Inset dimensions are what is declared in the target attributes. • If inset height = 0, then the height is scaled to keep image in proportion. • If inset width = 0, then the width is scaled to keep the image in proportion. • If the frame width = 0, then the frame width is determined by the formula: $FrameWidth = InsetWidth + (2 \times XOffset)$ • If the frame height = 0, then the frame height is determined by the formula: $FrameHeight = InsetHeight + (2 \times YOffset)$ *Note:* If after all this, any of the dimensions are zero, then an error will be generated.

The Graphic Segment Properties dialog box is shown in Figure 153. Table 59 describes its contents.

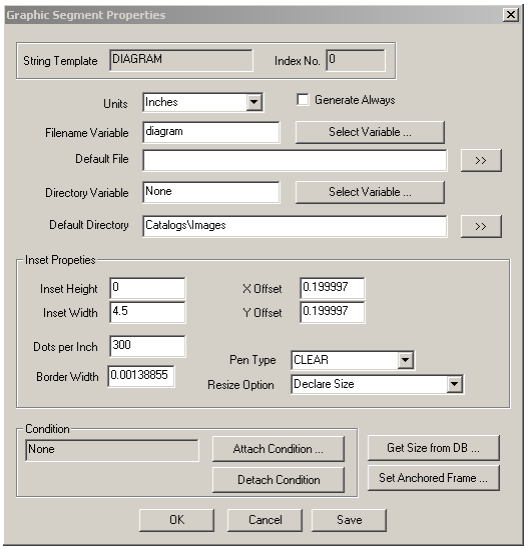


Figure 153: Graphic Segment Properties dialog box

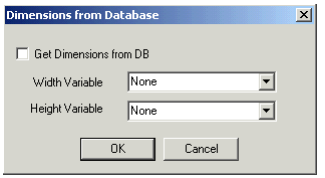
Table 59: Contents of the Graphic Segment Properties dialog box

Option	Description
Units	Select the units to use for measurements within this dialog box. Choose from Inches, Picas, Millimeters and Points.
Generate Always	If the variable that points to the file equals NULL, the Graphic segment won't generate anything. Equivalent to applying condition.
Filename Variable	The name of the variable that contains the filename of the graphic. Either type in the name of the variable directly or click the **Select Variable** button and choose a variable from the **Select Variable** dialog box.
Default File	The name of the file to use if the file specified by the filename variable does not exist. Select the **>>** button to browse for the file.

Table 59: Contents of the Graphic Segment Properties dialog box (continued)

Option	Description
Directory Variable	The name of the variable that contains the subdirectory where the graphic file resides. *Note:* If a variable is used, the value contained in this variable will override what is set in the Default Directory.
Default Directory	The directory where the image files are located.
Inset Properties	
Note: If the graphic's height and width are stored in the database, you can specify the variables that hold this data (by clicking the **Get Size from DB** button) instead of setting the graphic size in the Inset Properties area.	
Inset Height	Height of the graphic inset. If height is set to 0, the graphic is scaled proportionally to the given width.
Inset Width	Width of the graphic inset.
Dots per inch	The graphic's DPI (dots per inch) setting. For bitmaps, this value determines what the default height and width of an imported graphic will be. *Note:* This value is ignored for vector graphics such as EPS (encapsulated PostScript).
Border width	The width of the border around the graphic.
X Offset	The amount the graphic is offset from the left side of the anchored frame.
Y Offset	The amount the graphic is offset from the top of the anchored frame.
Pen Type	Specify the visibility and line weight of the border around the graphic (Clear=invisible).

Table 59: Contents of the Graphic Segment Properties dialog box (continued)

Option	Description
Resize Option	Select the resizing option for the graphic: • Adjust Cell Width—Adjust the anchored frame width and the containing table cell width (if a applicable) to the width of the imported graphic (based on the DPI). • Adjust Inset Width—Adjust the width of the inset and its parent frame to the containing cell or text column. • Declare size—You declare the dimensions of the inset in the Inset Height and Inset Width field or in the database's Width and Height variables. The inset is resized to these dimensions. • Import As Is—The graphic is imported at the specified DPI. Other dimensions are ignored.
Condition	To attach an optional condition to the segment, select the **Attach Condition** button and choose a condition from the Select Condition dialog box. To detach the condition from the segment, select the **Detach Condition** button.
Get Size from DB button	If the graphic's height and width are stored in the database, select this button to display the Dimensions from Database dialog box.  *Note:* Check **Get Dimension from DB** then select the variables that specify the graphic's width and height. The variables must be floating point, such as "1.5," "2.9," and so on.
Set Anchored Frame button	Displays the Anchored Frame Properties dialog box, which lets you define the properties of the anchored frame (Figure 154).

The Anchored Frame Properties dialog box is shown in Figure 154. Most of the fields in this dialog box correspond to fields in FrameMaker's Anchored Frame dialog box. Table 60 describes its contents.

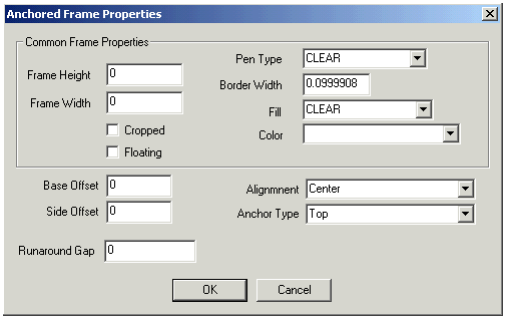


Figure 154: Anchored Frame Properties dialog box

Table 60: Contents of the Anchored Frame Properties dialog box

Option	Description
Common Frame Properties	
Frame Height	Specify the height of the frame in inches. If set to zero, the frame height is set to the height of the inset plus twice the Y Offset.
Frame Width	Specify the width of the anchored frame in inches. If set to zero, the frame width is set to the width of the inset plus twice the X Offset.
Pen Type	The visibility and line weight of the border around the anchored frame (Clear=invisible) in inches.
Border Width	The width of the border around the anchored frame.
Fill	The fill percentage to apply to the background color of the anchored frame.
Color	The color of the background of the anchored frame.
Cropped	Check to prevent a wide frame from extending beyond the edge of a column.
Floating	Check to let the frame float to the next column that can hold it if the frame and its anchor symbol won't fit in the same column.

Table 60: Contents of the Anchored Frame Properties dialog box (continued)

Option	Description
Alignment	Alignment options for the anchored frame relative to the column: • Center—Centers the frame on the page. • Inside—Aligns the frame to the side closest to the binding. On the left page of a double-sided document, the frame aligns on the right side. On the right page, the frame aligns on the left page. • Left—Left aligns the frame. • Outside—Aligns the frame to the side farthest from the binding. On the left page of a double-sided document, the frame aligns on the left side. On the right page of a double-sided document, the frame aligns to the right side. • Right—Right aligns the frame. *Note:* The alignment options only apply to the Below, Bottom, and Top anchor types. If you try to set the alignment for any other anchor type, the alignment is ignored.
Anchor type	Select the anchor position. The most common positions are: • Below—Below current line • Bottom—Bottom of column • Inline—At Insertion Point (alignment setting is doesn't work) • Run Into Paragraph—Run into paragraph, whatever alignment is selected • Top—Top of column, whatever alignment is selected *Note:* If you select an anchor type other than Bottom, Below, or Top, the alignment setting is ignored.
Base Line Offset	The offset from the baseline of the anchored frame.
Side Offset	Space between anchored frame and the edge of the page.

Defining an Index segment

An Index segment inserts an index marker in the text. You need to specify the headings under which the entry is found in the index and the variable that contains the marker's value. There are two ways to build the syntax for an Index segment:

- Using an attached string template.
- Setting parameters in the Index Segment Properties dialog box.

Figure 155 illustrates how the Index Segment Properties parameters build the syntax of an index marker.

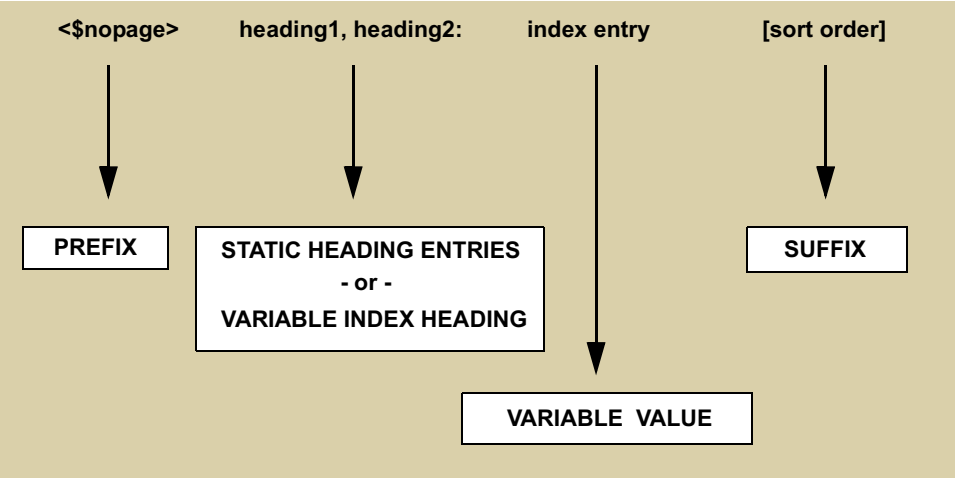


Figure 155: Building blocks for the syntax of an index marker

The Index Segment Properties dialog box is shown in Figure 156. Table 61 describes its contents.

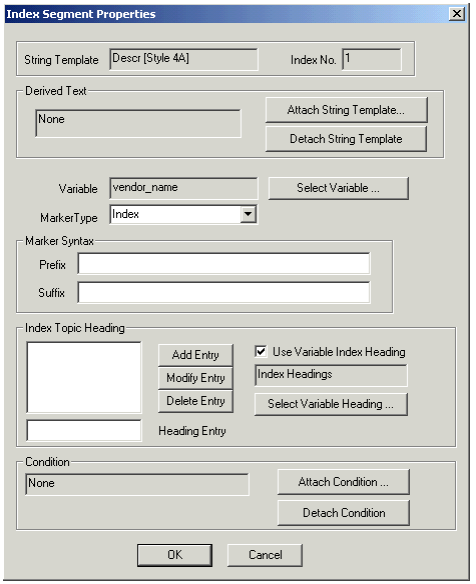
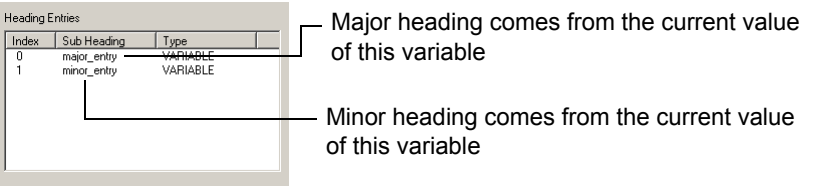


Figure 156: Index Segment Properties dialog box

Table 61: Contents of the Index Segment Properties dialog box

Option	Description
Attach String Template and Detach String Template	Select **Attach String Template** to add a string template that generates the entire syntax of an index marker. Select **Detach String Template** to remove the string template. If a string template is attached to an index segment, then *all* of the syntax settings in the Index Segment Properties dialog box are ignored, including the value of the attached variable. **Warning:** You should verify the syntax generated by the string template. As with all string templates, the syntax can be made up of both constant and variable parts, and if the syntax is invalid, the PatternStream application ignores the segment.
Marker Type	Unless you are creating a specialized index, select **Index** from the list.

Table 61: Contents of the Index Segment Properties dialog box (continued)

Option	Description
Marker Syntax	
Prefix	If you need to add syntax to precede each index entry, type the text here.
Suffix	If you need to add syntax to appear after each index entry, type the text here. *Note:* If you want to specify that a comma appear after the index entry, you can specify it here or on the Index Specification reference page of your FrameMaker template. Other sorting and formatting options are also available on this reference page. Consult the FrameMaker documentation for details.
Index Topic Heading	
FrameMaker allows you to group index entries under heading and subheading topics. To declare static headings and subheadings to group a particular index entry follow these steps: **a** In the Heading Entry field, type the heading you want. **b** Select the **Add Heading** button. The heading is displayed in the Index Topic Heading box. **c** Repeat steps a–b for each heading you want to add. You can also modify or delete a heading: • To modify a heading in the list, select the heading, make your corrections in the Heading Entry field, then select the **Modify Entry** button. • To delete a heading from the list, select the heading, then select the **Delete Entry** button.	
Use Variable Index Heading	If the index headings come from the database, check this box.
Select Variable Heading	Select this button then select the variable containing the index heading from the **Select Index Heading** dialog box. Below is an example of a list of variables (parameter list) that is acting as an index heading. 

Defining a Cross-Reference segment

The Cross-Reference segment inserts a cross-reference in your text. You specify the cross-reference source and the FrameMaker cross-reference format. The source must be an existing FrameMaker document.

The PatternStream application uses the marker method for determining cross references. The source document must contain a marker of type Cross-Ref. The marker text then must match the text in the cross reference. This should not be difficult, since the marker text and the cross-reference text are both derived from the database.

The Edit Cross Reference Segment dialog box is shown in Figure 157. Table 62 describes its contents.

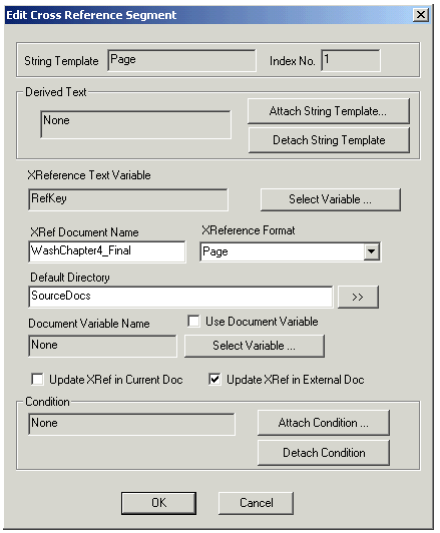


Figure 157: Edit Cross Reference Segment dialog box

Table 62: Contents of the Edit Cross Reference Segment dialog box

Option	Description
Attach String Template and Detach String Template	Select **Attach String Template** to add a string template that generates the text for a cross-reference marker. Select **Detach String Template** to remove the string template.

Table 62: Contents of the Edit Cross Reference Segment dialog box (continued)

Option	Description
XReference Text Variable	The variable containing the text of the cross-reference. To attach a variable to the segment, select the **Select Variable** button and choose a variable from the Select Variable dialog box.
XRef Document Name	You can specify the document containing the text you want to reference by specifying the file name and location in this and the Default Directory fields or by attaching the variable associated with the file name. If you are not using a variable, type the document's file name and extension in this field.
XReference Format	Choose a reference format. Cross-references formats are determined by the FrameMaker template attached to the parent Document target.
Default Directory	If you specified the document's file name in the XRef Document Name field, type the path name of the directory where the file is stored. You can also select the directory by selecting the **>>** button and browsing for its location.
Document Variable Name	If you did not specify a file name and location in the XRef Document Name and Default Directory fields, specify the document containing the text you want to reference by attaching the variable associated with the file name. To attach a variable to the segment, select the **Select Variable** button and choose a variable from the Select Variable dialog box. *Note:* Typically this option is used when the source document is also being generated using the PatternStream application. In this case, the document name is data-driven. This allows other documents to extract the source name from that same database.
Update XRef in Current Doc/Update XRef in External Doc	Check if you want the cross-reference updated in the current document, external document, or both. This update occurs when each cross-reference is being generated. If neither is checked, the cross-reference is not updated. *Note:* Sometimes turning off the update function can significantly decrease the execution time for large documents. You can use the Book target function to update the cross-references globally after the document is complete.

Table 62: Contents of the Edit Cross Reference Segment dialog box (continued)

Option	Description
Condition	To attach an optional condition to the segment, select the **Attach Condition** button and choose a condition from the Select Condition dialog box. To detach the condition from the segment, select the **Detach Condition** button.

Defining a Marker segment

Like the Index segment, the Marker segment inserts a marker and uses the value of the variable to create marker text. You use the Marker segment to embed markers in your output. For example, you may need to create cross-reference markers that resolve to a cross-reference segment you've created. By attaching a string template to a marker, you can create complex marker text, such as hypertext markup.

The Marker Segment dialog box is shown in Figure 158. Table 63 describes its contents.

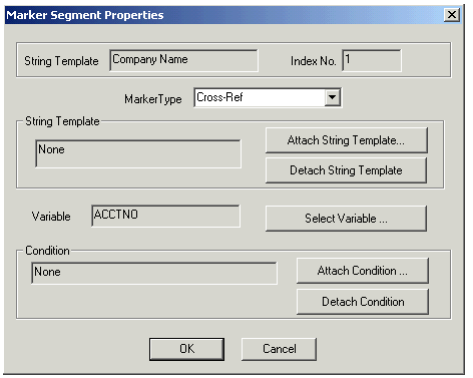
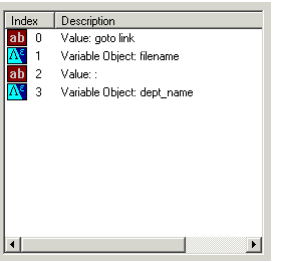


Figure 158: Marker Segment Properties dialog box

Table 63: Contents of the Marker Segment Properties dialog box

Option	Description
Attach String Template and Detach String Template	Select **Attach String Template** to add a string template that generates the marker text. Select **Detach String Template** to remove the string template. An example where you might want to use a string template to create the marker text is for generating syntax for a hypertext marker. If the syntax required is <gotolink *filename*:*linkname*> the string template might look like this: 
Marker Type	Choose a marker type. Available default markers and custom-defined ones are determined by the FrameMaker template attached to the parent Document target.

Defining a Text Markup segment

Like the Variable segment, the Text Markup segment inserts text from the database into your document. It differs from the Variable segment in that you can parse the inserted text for special character formatting (such as bold or italics). The segment works in conjunction with the Segment Transform ("Segment transforms" on page 334), in which you specify begin and end tokens that frame the text you want to format, much like the begin and end tags in HTML. When you apply this transform to the Text Markup segment, the transform scans the current value of the variable for these tokens and applies the character formatting

you specify in the transform definition. (See Chapter 17, “Transforms” for details on how to create the transforms markup table).

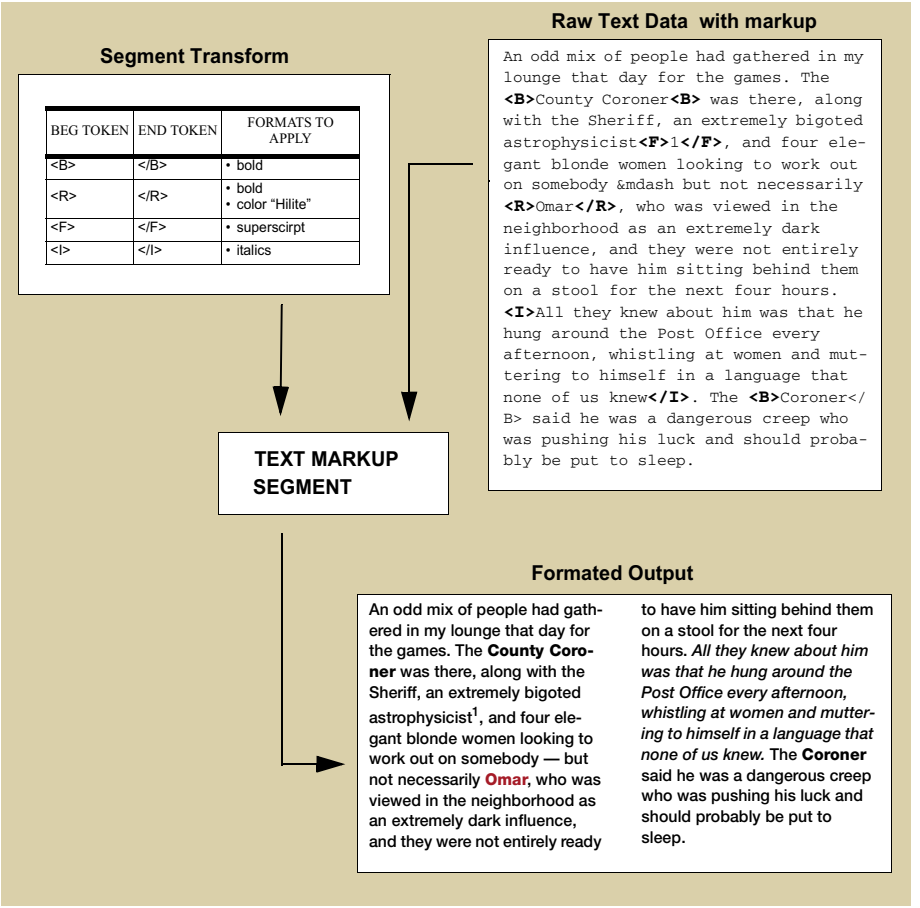


Figure 159: Action of a text markup segment

The Text Markup Segment dialog box is shown in Figure 160. Table 64 describes its contents.

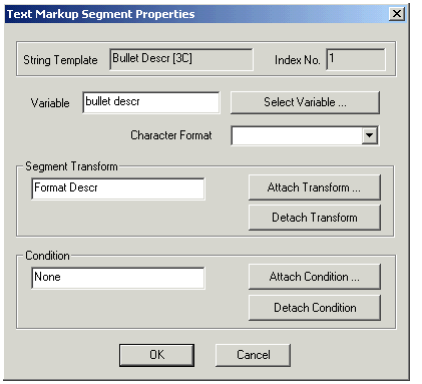


Figure 160: Text Markup Segment Properties dialog box

Table 64: Contents of the Text Markup Segment Properties dialog box

Option	Description
Segment Transform	To attach a segment transform to the segment, select the **Attach Transform** button and select the transform you want from the Select Transform dialog box. To detach the transform from the segment, select the **Detach Transform** button.

Defining a Flow segment

The Flow segment lets you import part or all of another FrameMaker document as a text inset. To insert part of a document, the document you're referencing must have named flows defined on its reference page. For information about defining these flows in FrameMaker, consult your FrameMaker documentation.

The Flow Segment dialog box is shown in Figure 161. Table 65 describes its contents.

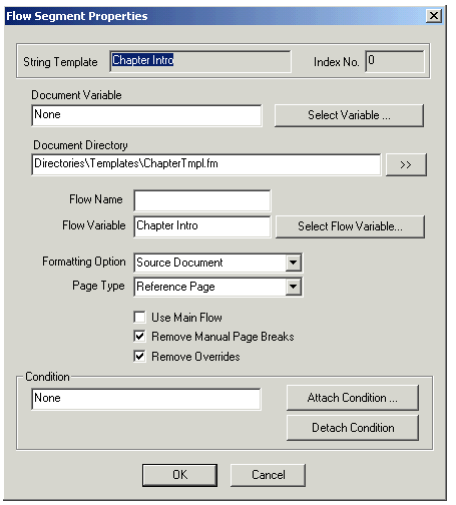


Figure 161: Flow Segment Properties dialog box

Table 65: Contents of the Flow Segment Properties dialog box

Option	Description
Document Directory	Type the path name of the directory where the document you want is stored. You can also select the directory by selecting the >> button and using the Select Directory dialog box to locate the directory.
Flow Name	If you are inserting only part of the document, specify the flow name either by typing the name here or, if you are extracting the flow name from the database, attaching a flow variable.
Flow Variable	If you are inserting only part of the document and are extracting the flow name from the database, select the **Select Flow Variable** button and select a variable from the Select Variable dialog box.

Table 65: Contents of the Flow Segment Properties dialog box (continued)

Option	Description
Formatting Option	Select a formatting option: • **Enclosing document:** The imported text picks up the paragraph formatting of the target document (that is, the document the pattern set creates). • **Plain text:** All paragraph and character formatting is stripped from the imported text. • **Source document:** The imported text retains the character and paragraph formatting of its source document.
Page Type	Select a page type: • **Body Page:** The text comes from the source document's main flow. • **Reference Page:** The text comes from named flows on a reference page.
Use Main Flow	Check if the text is coming from the main flow.
Remove Manual Page Breaks	Check if you want manual page breaks removed from the imported text.
Remove Overrides	Check if you want overrides removed from the imported text. Overrides occur when a user manually modifies character or paragraph formats or the document's layout. By removing overrides, those changes are replaced by the document's defined formats and layout.

Defining a Variable Format segment

The Variable Format segment inserts a FrameMaker variable into the text. You can use standard variables, such as Current Date or Filename, or you can create custom variables in the FrameMaker template attached to the current pattern set. You input the name of the variable directly or use a PatternStream application variable that contains the name.

The Variable Format Segment dialog box is shown in Figure 162. Table 66 describes its contents.

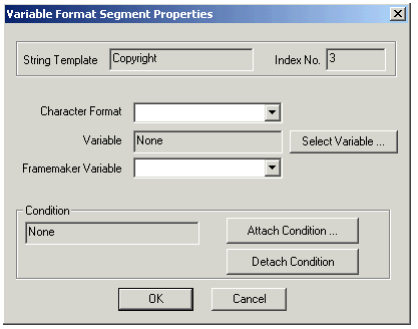


Figure 162: Variable Format Segment Properties dialog box

Table 66: Contents of the Variable Format Segment Properties dialog box

Option	Description
FrameMaker Variable	If you did not specify the FrameMaker variable using a PatternStream application variable, select the FrameMaker variable name from the list. Available default and custom-defined variables are determined by the FrameMaker template attached to the parent Document target.

Defining a Text Inset segment

You can import text from other applications such as Microsoft Word or WordPerfect using a Text Inset segment. You can enter import hints to assist FrameMaker in deciding which filter to use. Consult the FrameMaker documentation for details on using import hints.

The Text Inset Segment dialog box is shown in Figure 163. Table 67 describes its contents.

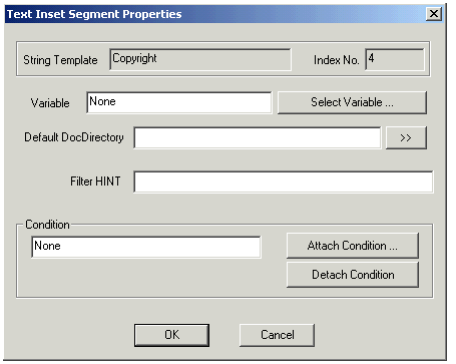


Figure 163: Text Inset Segment Properties dialog box

Table 67: Contents of the Variable Format Segment Properties dialog box

Option	Description
Default DocDirectory	Type the path name of the directory where the document you want is stored. You can also select the directory by selecting the >> button and using the Select Directory dialog box to locate the directory.
Filter Hint	A hint to help FrameMaker decide which filter to use for importing the file.

Defining a New Paragraph segment

To create a new paragraph with a particular format, insert a new paragraph segment. The segment works the same as inserting a carriage return except that you specify the format of the new paragraph. You can either select the paragraph format, as with a Paragraph target, or the value of a variable can determine the format.

The Paragraph Segment Properties dialog box is shown in Figure 164. Table 68 describes the properties.

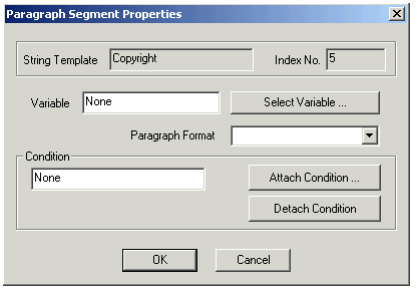


Figure 164: Paragraph Segment Properties dialog box

Table 68: Contents of the Paragraph Segment Properties dialog box

Option	Description
Paragraph Format	To assign the paragraph format here, select it from the Paragraph Format drop-down list.

Defining a String Template segment

A String Template segment lets you create complex, hierachical string templates from simple segments. You select the string templates you want to attach to the segment, and you can also attach conditions.

Note: Be sure you do not create a circular condition where the child of a string template has its parent attached to it.

The String Template Segment Properties dialog box is shown in Figure 164. Table 68 describes the properties.

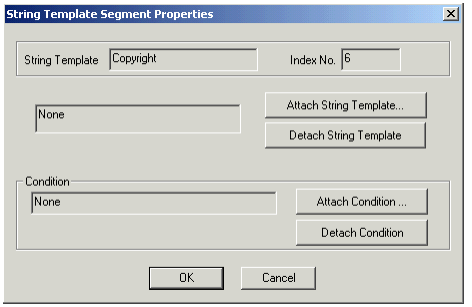


Figure 165: String Template Segment Properties dialog box

Table 69: Contents of the Paragraph Segment Properties dialog box

Option	Description
Attach String Template and Detach String Template	Select **Attach String Template** to add a string template that generates the text for a cross-reference marker. Select **Detach String Template** to remove the string template.

Chapter 16: Creating conditions

Conditions are objects that evaluate to either TRUE or FALSE which you usually attach to targets or string template segments. The segment or target is generated only if the condition is TRUE. The different types of conditions are list in Table 70.

Table 70: Condition types

Type	Description	For details see...
Expression 	A simple expression, with an operator and two operands, that tests whether a condition is true or false. If the current value of the expression is true, then the condition is TRUE. $(NumStates > 0)$	"Defining an expression" on page 299
AND List 	A list of conditions, all of which must be TRUE for the condition to be true. You must first create the conditions you want to add to the AND list.	"Defining an AND Condition" on page 302
OR List 	A list of conditions, any of which can be TRUE for the condition to be TRUE. You must first create the conditions you want to add to the OR list.	"Defining an OR Condition" on page 304

Note: AND and OR condition lists can be as complex or hierarchical as you need; however, the PatternStream application does not verify conditions. Check your conditions carefully to prevent circular references.

Working with Condition Objects

You normally access condition objects from the Conditions/PList tab of the PatternStream main dialog (Figure 166).

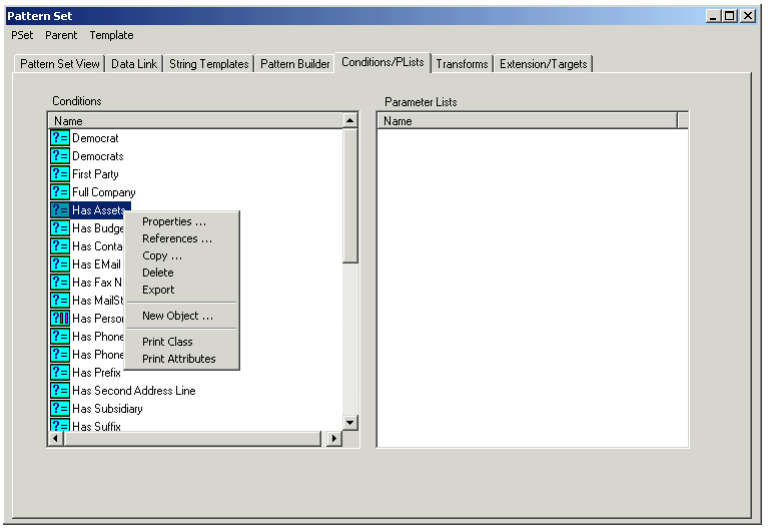


Figure 166: Conditions/PLists tab

Creating a condition

To create a condition, follow these steps:

1 Click on the white area of the condition window with the right mouse button and select New Object from the popup menu. The Create New Condition Object dialog box is displayed (Figure 167).

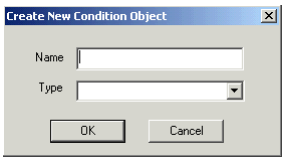


Figure 167: Create New Condition Object dialog box

2 In the Name field, type the name you want for the condition.

3 Select the condition type from the Type drop-down list (see Table 70)

4 Select **OK**. The condition edit dialog for the type of condition chosen is displayed.

5 Enter the appropriate attributes and Click on the OK button.

6 The new condition and its icon are displayed in the conditions list. Conditions are listed in alphabetical order.

Modifying a condition

You can access the properties dialog of a condition by double clicking either:

- its icon in the conditions list of the Conditions/PList tab.
- its icon in the conditions folder of the Pattern View tab.

Copying a condition

If you are creating a condition that is similar to an existing one, you can save time by copy the existing one and modifying its properties.

To copy a condition, follow these steps:

1 From the **Conditions/PList** tab, select the condition you want to copy.

2 Right mouse-click, then select **Copy** from the object popup menu.

3 The Copy Condition Object dialog box is displayed, showing the name of the copied condition (Figure 168).

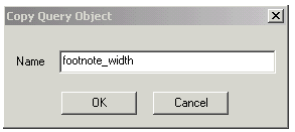


Figure 168: Copy Condition Object dialog box

4 Type the name you want for the new condition, then select **OK**. The condition properties dialog for the type of condition copied will be displayed.

5 Modified the attributes appropriately then close the dialog.

6 The new condition now appears in the condition list on the **Conditions/PList** tab.

Finding references

Many times it is necessary to find out which objects (string template or target) are using a particular condition. To find these references do one of the following:

- Select the **Conditions/PList** tab of the PatternStream main dialog, select the particular condition, right-click and select **References** from the object popup menu.

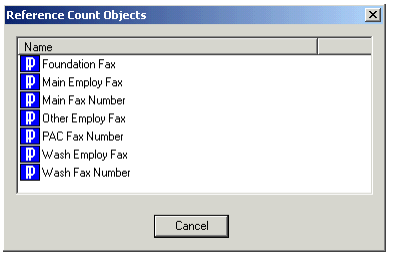


Figure 169: Condition Reference dialog

- Select the Pattern Set View tab of the PatternStream main dialog. Below the PSET hierarchy, there is a folder containing a list of all condition objects for the current pattern set in alphabetical order. Select the desired condition object, click on the right mouse button, then select **References** from the popup menu.

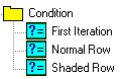


Figure 170: Condition list from the Pattern Set View tab.

Deleting a condition

Since you can delete only those conditions not currently used by other objects in the PatternStream application, you must remove the condition's reference from all objects before deleting the condition. See "Finding references" on page 296 for details about determining if a condition is currently in use.

To delete a condition, follow these steps:

1. Remove references to the condition from all other objects in the PatternStream application. See "Using Conditions with other PatternStream Objects" on page 297 for details.
2. Select the condition you want to delete from the Conditions section of the **Conditions/PLink** tab.

3 Select the **Delete Condition** button. A confirmation pop-up box is displayed.

4 To delete the condition, select **OK**.

Note: You can also delete conditions from the **Pattern Set View** tab by selecting the condition in the pattern set then selecting the **Object Delete** button.

Using Conditions with other PatternStream Objects

Targets

When you attach a condition to a target, that target will generate a structural element only if the condition is TRUE. If a target has more than one condition attached to it, all conditions must be TRUE before the target is generated. To attach a condition to a target, follow these steps:

1 Open the properties dialog for the target you wish to attach the condition to. See "Working with Targets" on page 142 for details on the various ways to access target properties.

2 Select the **Conditions** tab.

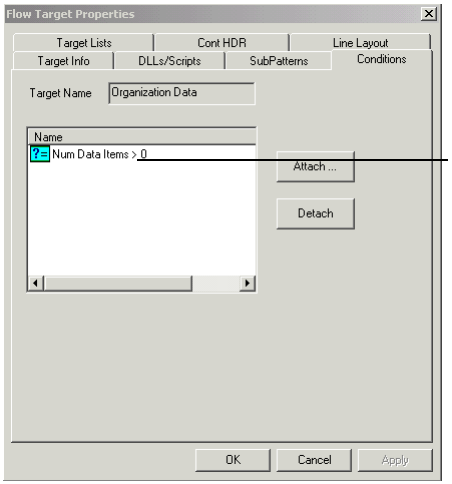


Figure 171: Conditions tab of the target properties dialog.

3 Select the **Attach Condition** button. The **Select Condition** dialog box is displayed.

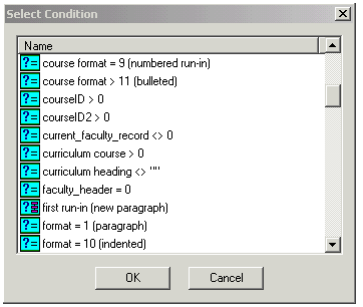


Figure 172: Select conditions dialog.

4 Select the condition you want to attach, then select **OK**. This close the **Select Condition** dialog.

5 To incorporate the changes to the current target, click on the **Apply** button, or click **OK** to close the properties dialog.

Segments of string templates

When you add a condition to a segment of a string template, the segment generates text only if the condition is TRUE. If it is FALSE, the segment is skipped. You can add only one condition to a segment. If you need to apply more than one condition to a segment, combine them using an AND or OR condition. To attach a condition to a string template segment, follow these steps:

1 From the **String Templates** tab, select the string template containing the segment to which you want to apply a condition. The segments are displayed in the list box on the right side of the dialog.

2 Select the icon or index number of the segment you want and double click on it. The properties dialog box for that segment type is displayed.

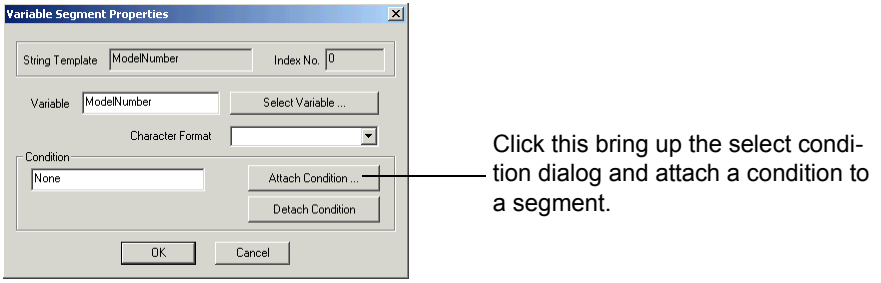


Figure 173: Variable Segment Properties dialog box

3 In the Conditions section of the dialog box, select the **Attach Condition** button. The Select Condition dialog box is displayed.

4 Select the condition you want to attach, then select **OK**.

5 To save your changes, select **OK**.

Defining an expression

An expression object is a logical statement made up of a left hand side, a right hand side and a logical operator between them.

When you create an expression, the Edit Expression dialog box is displayed (Figure 174).

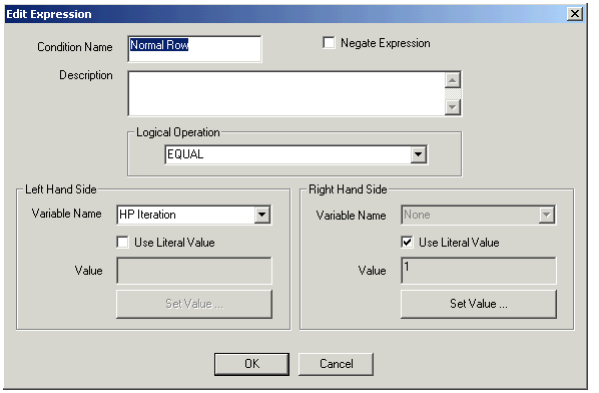


Figure 174: Edit Expression dialog box

Table 71: Expression parameters

Parameter	Description
Condition Name	Enter the name of the expression.
Description	Enter an optional description for the condition.
Negate Expression	Check this box if you wish to reverse the logic of the expression.
OK	Click this button to incorporate changes into the current pattern set.
Cancel	Click this button to exit the dialog without incorporating any changes made.
Left Hand Side	
Variable Name	Either type in the name or use the drop-down list to select a variable for the left hand side of the expression.
Use Literal Value	Check this button if the left hand side will contain a constant or literal value. This will disable the Variable Name control.
Set Value	Click this button to set the literal value. This will bring up the Value Edit dialog (Figure 175).

Table 71: Expression parameters

Parameter	Description
Right Hand Site	
Variable Name	Either type in the name or use the drop-down list to select a variable for the left hand side of the expression.
Use Literal Value	Check this button if the left hand side will contain a constant or literal value. This will disable the Variable Name control.
Set Value	Click this button to set the literal value. This will bring up the Value Edit dialog (Figure 175).
Expression connector	
Logic Operation	Select the desired logic operation from drop-down list. Table 72 list the valid logical operators and describes their effect.

Table 72: Logical operator options

Operator	Condition is true if[a]
EQUAL	`LHS = RHS`
NOT EQUAL	`LHS <> RHS`
GREATER THAN	`LHS > RHS`
GREATER THAN EQUAL TO	`LHS >= RHS`
LESS THAN	`LHS < RHS`
LESS THAN EQUAL TO	`LHS <= RHS`
NO OP	Always. NO OP is a placeholder that lets you create a condition and define it later without affecting the execution of the pattern set.
SUBSTRING	If `LHS` is a substring of the `RHS`.
IS NULL	`LHS` is NULL.

Table 72: Logical operator options (continued)

Operator	Condition is true if[a]
IS NOT NULL	LHS is not NULL.

a. In this table, LHS refers to the left side of the expression, and RHS stands for the right side.

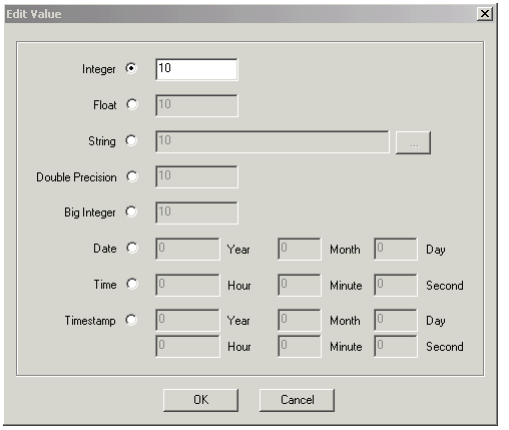


Figure 175: Edit Value dialog box

Defining an AND Condition

1 An AND list is a list of conditions (expressions, OR conditions, or other AND conditions), *all* of which must test TRUE for the condition to be true.

Figure 176 is an example of the AND condition dialog, while Table 73 lists each

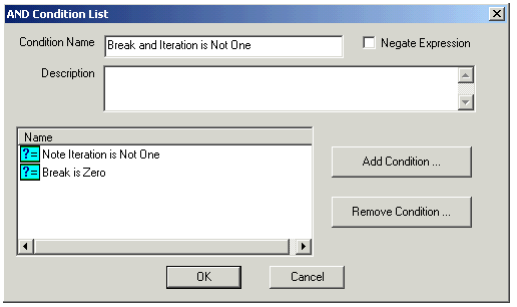


Figure 176: AND Condition List dialog box

parameter and a description.

Table 73: AND condition parameters

Parameter	Description
Condition Name	Enter the name of the and condition.
Description	Enter an optional description for the condition.
Negate Expression	Check this box if you wish to reverse the logic of the expression. **DeMorgan Rules** Sometimes is easily to negate a complex AND condition rather than work out the equivalent logical statement. $\neg(A \wedge B \wedge C) = \neg A \vee \neg B \vee \neg C$
OK	Click this button to incorporate changes into the current pattern set.
Cancel	Click this button to exit the dialog without incorporating any changes made.

Add a Condition

1. Click on the Add Condition button. This brings up the Select Condition dialog.
2. Select the desired condition then click OK. The selected condition is now displayed in the list of conditions connected by a logical AND.

Remove a Condition

1 Select the condition in the condition list that you wish to remove.

2 Click on the Remove button. The select condition is now removed from the list of conditions connected by a logical AND.

Defining an OR Condition

An OR list is a list of conditions (expressions, AND conditions, or other OR conditions), *any* of which can test true for the condition to be true.

Figure 177 is an example of the OR condition dialog, while Table 74 lists each

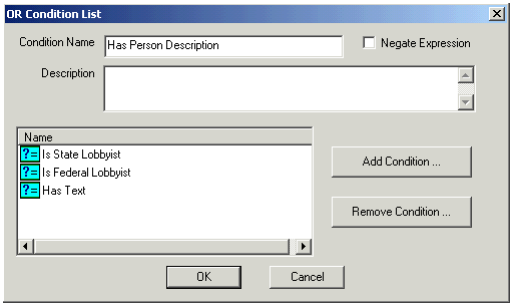


Figure 177: OR Condition List dialog box

parameter and a description.

Table 74: AND condition parameters

Parameter	Description
Condition Name	Enter the name of the OR condition.
Description	Enter an optional description for the condition.

Table 74: AND condition parameters

Parameter	Description
Negate Expression	Check this box if you wish to reverse the logic of the expression. **DeMorgan Rules** Sometimes is easily to negate a complex OR condition rather than work out the equivalent logical statement. $\neg(A \vee B \vee C) = \neg A \wedge \neg B \wedge \neg C$
OK	Click this button to incorporate changes into the current pattern set.
Cancel	Click this button to exit the dialog without incorporating any changes made.

Add a Condition

1. Click on the Add Condition button. This brings up the Select Condition dialog.
2. Select the desired condition then click OK. The selected condition is now displayed in the list of conditions connected by a logical OR.

Remove a Condition

1. Select the condition in the condition list that you wish to remove.
2. Click on the Remove button. The select condition is now removed from the list of conditions connected by a logical OR.

Chapter 17: Transforms

A transform is a set of directives that changes, or transforms, the value contained in a PatternStream variable. For example, using a String transform, you can remove extra carriage returns, do string substitution and remove trailing blanks. With Numeric transforms, you can perform mathematical operations on a returned numeric value so that the operation's result is displayed in your document instead of the original value. Each transform is made of one or more directives that specify what operation to perform on the current value of a variable. These directives are performed in the order they are listed in the Directive list box on the **Transforms** tab.

This chapter describes the different types of transforms and details how to create, edit, and attach them.

Table 75 describes each type of transform.

Table 75: Transform types

Type	Description	For details, see...
String 	Transform the values of the following type of variable: • String • Text • Reference (when referencing a string or text variable)	"String transforms" on page 319
Numeric 	Transform the values of the following type of variable: • Integer • Float • Big Integer • Double • Reference (when referencing a numeric variable)	"Numeric transforms" on page 329

Table 75: Transform types

Type	Description	For details, see...
Date/Time 	Transform the values of the following type of variable: • Date • Time • Timestamp • Reference (when referencing a date type variable)	"Date transforms"
Segment 	Manipulate the format characteristics within a variable that is being used to generate text in a segment of a string template.	"Segment transforms" on page 334

Transform basics

Transforms can be divided into two sub-classes:

- Variable transforms
  Transforms that operate on PatternStream variables. Numeric, String, and Date transforms are all Variable transforms.
- String Template transform
  A special kind of transform that works in association with string templates. The Segment transform is a String Template transform.

The following discussion refers only to Variable transforms. For a discussion of String template transforms see "Segment transforms" on page 334. For details on PatternStream variables, see Chapter 5, "Variables."

A transform can be thought of as a mapping (in the mathematical sense) that takes a value, performs a series of manipulations on that value, then returns another value. Table 76 provides some examples of Variable transforms.

Table 76: Variable transform examples

Initial Value	Transform Description	Result Value	Directive(s) Used
10	Increment by 1	11	Unary Operation (add 1)

Table 76: Variable transform examples (continued)

Initial Value	Transform Description	Result Value	Directive(s) Used
1	Increment by 1, perform division modulo 2	0	Unary Operation (add 1) Unary Operation (mod 2)
Leopold Bloom ate with relish the inner organs of beasts and fowls.	Remove carriage returns.	Leopold Bloom ate with relish the inner organs of beasts and fowls.	Character Replace (0d ----> <nothing>)
“New York ”	Remove trail blanks	“New York”	String Operation (TRIM)
NULL	Extract the month from a Date variable with the value (1999, 4, 1)	4	Date Component (MONTH)
71489	Concantenate the file extension “.gif” to form a graphic filename.	71489.gif	String Concatenation (<object_ide> + “.gif”)

In order to use transforms effectively, you need to understand a few basic things about PatternStream variables. Each variable contains two values, the Raw Value and the Current Value. The Raw Value contains the actual data bytes extracted from the database and encapsulates the computer memory that is actually bound to the database driver. The Current Value is the value that other objects, such as Conditions and String Templates, will actually see. In most cases, the two values are the same. However, if an object modifies the Raw Value, it must do it in a non-destructive way in order to maintain the connection with the database driver. This difference can affect the way a transform behaves and hence the method chosen to apply a given transform. Variable transforms can be applied in two different ways:

- by attaching them directly to a variable
- by using them in an Assignment target

Attaching Variable transforms directly to a variable

When you use this method, the transform is applied to the Raw Value in order to arrive at the Current Value. Other objects (with one exception, see "Assignment target" on page 247) have access only to the Current Value when they need the value a variable.

Note: The transform is applied each time the variable is evaluated, even though the Raw Value may not have changed.

The application of the transform does not change the Raw Value. Only the Current Value as seen by the other objects is modified. Consider the following example:

Example 4: Raw Value versus Current Value

> An integer variable is used is used as an iteration counter for some query (see Chapter 6, "Constructing queries"). The Raw Value of the variable takes on the values {1, 2. 3. 4. 5...} in sequence as the query object returns rows from the database. Suppose a transform which does a modulo 2 operation is attached to the variable. Then the Current Value will take on the values {1, 0, 1, 0, 1, ...}.If the variable is used in a condition, then that condition should test for only the values 0 or 1, since those are the only two values it will have access to.

To attach a Variable transform to a PatternStream variable, follow these steps:

1. From the Variables area of the **Data Link** tab, select the variable's name or icon, right-click, then select **Properties** from the pop-up menu, or double click on the variable's icon. The properties dialog box for that variable type is displayed.
2. In the Variable Transform area of the dialog box, select the **Attach Transform** button. The Select Transform dialog box is displayed.
3. Select the transform you want to attach, then select **OK**.

   Note: The only transforms shown in the list are those valid for that type of variable.

Using Variable transforms in Assignment targets

When a transform is used in an Assignment target, it actually modifies the Raw Value of the target variable ("Assignment target" on page 247). Instead of being applied every time the variable is evaluated, the transform is only applied when the Assignment target is invoked. This is an important distinction when the

transform itself uses a variable that can change its value, such as in string concantenation and binary operations.

To use a transform in an assignment target, you must first access the target object's properties dialog. See Chapter , "Assignment target" for more details on attaching transforms to assignment targets.

Directives

Directives are the objects that actually do the transforming. A transform can be thought of as a composite operation, where each directive is applied to the target value in sequence. It is important to distinguish between two types of directive, cumulative and destructive.

A cumulative directive will operate on the target value, and the result depends on what the original value was. The following transformations are examples of cumulative operations:

- changing a string to upper case.
- incrementing the value of a counter variable by one.
- concatenating the file extension to a variable that contains a filename.

A destructive directive will completely replace the current value with a new one. This new value will depend on the directive itself, without regard to what the original value was. The following transformations are examples of destructive operations:

- finding the length of string.
- extracting a value from a lookup table (array).
- checking for the existence of a given file.

Table 77 classifies each directive in terms of cumulative or destructive.

Table 77: Cumulative vs. destructive directives

Cumulative	Destructive
String Operation	File Exists
String Substitute	Current Time
String Replace	Date Component
Character Replace	Date Convert
Unary Operation	Numeric Convert
Binary Operation	String Convert
String Operation	Lookup
String Concatenation	Function
Parse Markup	

Creating Transforms

To create a new transform, follow these steps:

1 Select the **Transform** tab on the PatternStream main dialog.(Figure 178).

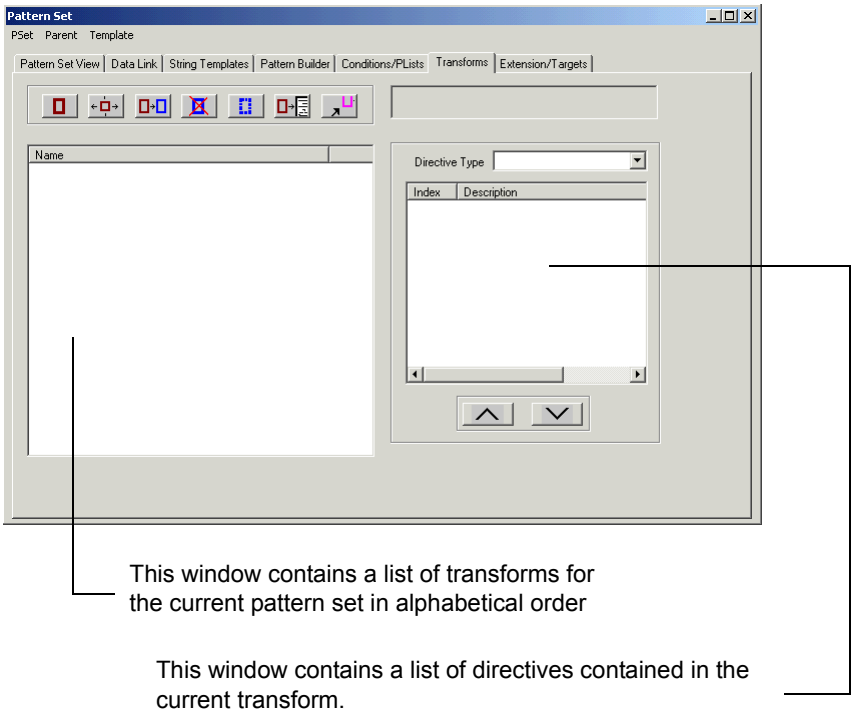


Figure 178: Transform tab

2 Select the **Parent** menu, then select **New Object.** The Create New Transform Object dialog box is displayed (Figure 179).

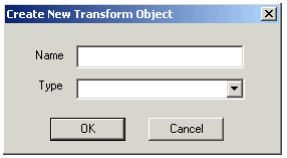


Figure 179: Create New Transform Object dialog box

3 In the **Name** field, type a name for the new transform.

4 Select the type of transform from the **Type** drop-down list.

5 Select **OK**. The new transform and its icon are displayed in the transform list and is now the current transform

After you create a transform, you define it by adding directives.

Working with transforms

To work with a transform, you must first select the **Transform** tab of the main dialog. On the left side of the dialog, existing transforms for the current PSet are displayed in alphabetical order. To select a transform and make it the current transform, click on the icon of the desired transform. Its name is displayed in the Current Transform field.and the list of directives that make up the transform are displayed in the right window (Figure 180).

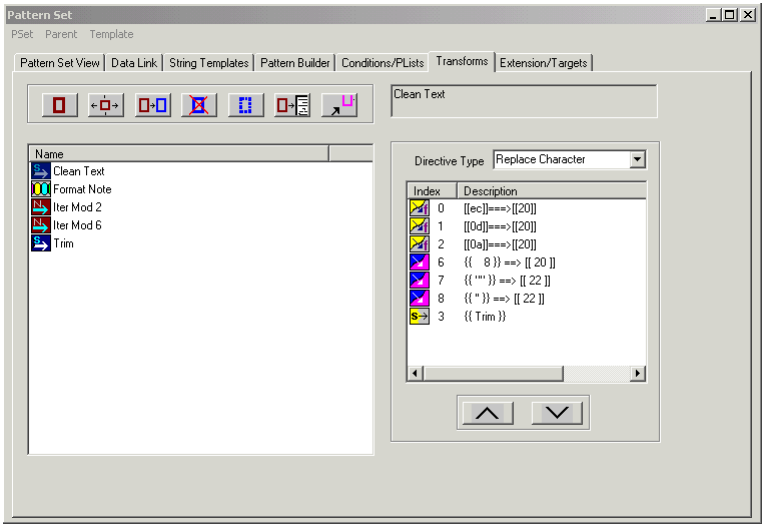


Figure 180: Current transform and its associated directives

Modifying transform name and description

The directives that make up a transform contain the essence of the transform's definition. However, two properties—the name and description—belong to the transform itself.

To modify the name and description, follow these steps:

1 Make the desired transform the current transform**.**

2 Select Object Info from the parent menu.

3 The Edit Transform Properties dialog box is displayed (Figure 181).

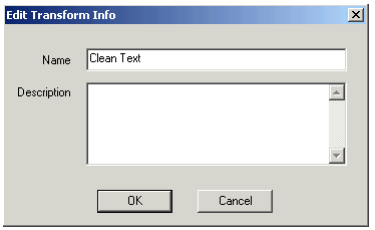


Figure 181: Editing a transforms name and description

4 To change the transform name, type the new name in the **Name** field.

5 To include a description with the transform, type it in the Description field. This description is for documentation purposes and will appear when the transform's attributes are printed out.

6 Select **OK.** The current transform will appear in the transform list under the new name.

Note: Any object that references the current transform will now indicate the new name.

Finding references

Many times it is useful to determine what other objects are using a specific transform. There are two ways you can view the list of referencing objects:

1 Through the Transform tab by making the desired transform the current transform, then selecting References from the **Parent** menu.

2 From the Pattern Set View tab, select the desired transform from the list of transforms, click the right mouse button and select References from the popup menu.

In Figure 182, we see an example of an object reference list for a given transform.

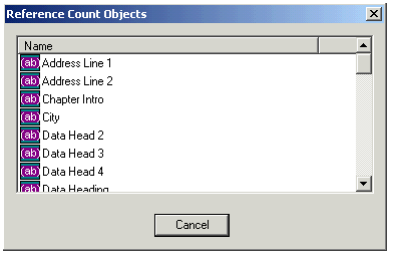


Figure 182: Objects that reference the transform

Deleting transforms

You can delete any transform that isn't attached to an object. If the transform is attached, you need to find the object and detach the transform. There are two ways you can delete a transform:

1. Through the Transform tab by making the desired transform the current transform, then selecting **Delete** from the **Parent** menu.
2. From the Pattern Set View tab, select the desired transform from the list of transforms, click the right mouse button and select **Delete** from the popup menu.

Copying transforms

Copying a transform not only copies the transforms itself but all of its directives and their attribute settings.

To copy a transform, follow these steps:

1. Select the **Transform** tab, then select the transform you want to copy, making it the current transform.
2. Select the **Parent** menu, then select **Copy.** The Copy Transform dialog box is displayed (Figure 183).

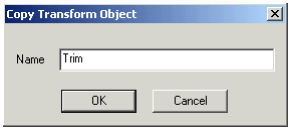


Figure 183: Copying a transform

3 Type the name of the new transform and select **OK.** The new copied transform becomes the current transform.

Note: Two transforms cannot have the same name, so you must change the name in the copy object dialog.

You can also copy transforms from the **Pattern Set View** tab. Select then right-click the transform, then select **Copy** from the pop-up menu.

Exporting transforms

In the PatternStream application, you export objects to use them in other PSet files. To export the current transform to the export buffer, select the **Parent** menu, then select **Export.**

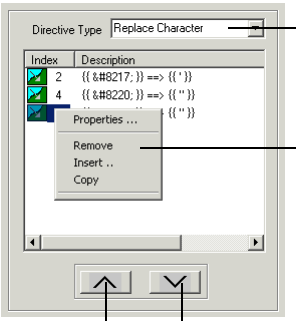
You can also export transforms from the **Pattern Set View** tab. Select then right-click the transform, then select **Export** from the pop-up menu.

Printing transform attributes

You can print transform properties by selecting **Print Object** from the **Parent** menu. A FrameMaker document is created listing the attributes of directives that make up the current transform. You can also print a list of the transforms in the current pattern set by selecting **Print Class** from the **Parent** menu.

Working with Directives

A transform is defined by one or more directives. Directives operate on the target variable in the order they appear the directive list.



Select the type of directive to insert from the drop-down list.

Select a directive, then right click to bring up the directive popup menu, then select the operation you wish to perform.

After selecting a directive, use these arrow buttons to move that directive up or down in the list. The order a directive appears in the list is the order that its operation will be performed.

Figure 184: Creating and modifying directives.

Insert a new directive

To insert directives into the current transform:

1 From the **Transforms** tab, select the transform you want to edt, making it the current transform.

Note: You can also select the transform from the **Pattern View** tab, right-click, and select **Properties** from the pop-up menu. The **Transform** tab is automatically selected.

1 Select a directives type from the Directive Type drop-down list. The directive types that appear in this list will depend on the type of transform the directive will belong to.

2 Right-click anywhere in the Directives area, then select **Insert** from the pop-up menu. The new segment and its icon are added to the directive list. Directives are invoked in the order they are listed. (To change a directives position in the list, select the directive, then select the up or down arrow.)

Copying a directive

Right mouse click on the directive you wish to copy, select Copy from the directive popup menu.

Removing a directive

Right mouse click on the directive you wish to copy, select Delete from the directive popup menu.

Editing a directive

Right mouse click on the directive you wish to edit, select Properties from the directive popup menu or double click on the desired directive.

String transforms

String transforms alter or replace a character or character string. You can attach string transforms to String, Text, or Reference variables (the Reference variable must reference a String or Text variable).

Table 78 describes the directive types specific to the String transform.

Note: The Function and Table Lookup directives are common to all transforms and are used only for advanced features. See “Directives for advanced features” on page 336 for details.

Table 78: String transform directive types

Directive	Description	For details see...
Replace Character 	Replaces one character in a string with another character	“Replace Character directive” on page 320
String Operation 	Performs an operation on a character string, such as making all characters upper- or lowercase	“String Operation directive” on page 321
Replace String 	Replaces a character string with a character you specify	“Replace String directive” on page 323

Table 78: String transform directive types (continued)

Directive	Description	For details see...
Substitute String 	Substitutes the occurrence of one string with another.	
String Concatenation 	Concatenates a variable or constant string to the current variable.	
Convert Number 	Converts a numeric variable to a string using a specified format.	
Convert Date 	Converts a date, time or timestamp variable to a string using a specified format.	
Convert String 	Converts a string variable to a another string using a specified format.	

Replace Character directive

The Replace Character directive replaces all of the occurrences of a specified character in the value of a string variable with another character. The most common use of this directive is to clean data strings of junk characters, such as extraneous carriage returns, inserted as a by-product of some database front-end applications.

Use the Character Replace Directive dialog box, shown in Figure 185, to define the directive. Table 79 describes the contents of the dialog box.

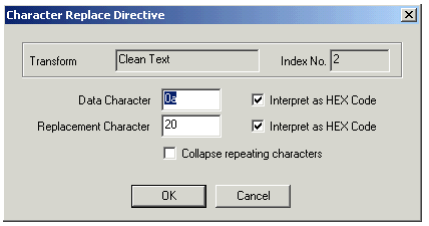


Figure 185: Character Replace Directive dialog box

Table 79: Contents of the Character Replace Directive dialog box

Option	Description
Data Character	Type the character you want to replace. For extended characters, type the character's hex code and check the **Interpret as HEX Code** check box.
Replacement Character	Type the replacement character. For extended characters, type the character's hex code and check the **Interpret as HEX Code** check box.
Interpret as HEX Code.	Check if you entered the character's HEX code in the Data Character or Replacement Character field
Collapse Repeating Characters.	This will replace repeating sequences of the data character with only one instance of the replacement character.

Note: A chart containing character hex codes is available in FrameMaker's online help. In FrameMaker, select the **Help** menu, then **Online Manuals,** and then **FrameMaker Character Sets.**

String Operation directive

The String Operation directive applies the operation you specify to the entire character string of the variable to which the directive's transform is attached. For example, the values in a database column containing employee's last names are in initial capitals only (that is, "Jones"), but in your company directory, you want

last names in all capitals (that is, "JONES"). Use the Upper Case operation in this directive to capitalize all of the characters in the string.

Use the String Operation Directive dialog box, shown in Figure 186, define the directive.

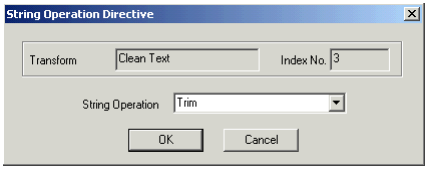


Figure 186: String Operation Directive dialog box

From the String Operation drop-down list, select the operation you want to apply to a character string (see Table 80).

Table 80: String operations

Operation	Description
Capitalize	Set the first character in each word (i.e., characters delimited by a space) to upper case.
Frame Char Set	Converts ANSI character values to their corresponding FrameMaker counterparts. The numeric values Frame uses for characters in some cases differs from the character's ANSI value. If any characters in the output differ from their values in the database, apply this directive's transform to the appropriate variable, and the PatternStream application will convert the ANSI value to the FrameMaker value.
Lower Case	Makes all characters in the string lowercase.
Trim	Removes the leading and trailing blanks of a string.
Upper Case	Makes all characters in the string uppercase.
Smart Quotes	
Smart Quotes - Single	

Replace String directive

This directive replaces each occurrence of a specific string of characters in the value of a string or text variable with a single character. For example, suppose you are retrieving text strings from a database that must be used for both print and HTML output. Any special characters in the text might be represented by special symbols such as &diaeresis, &trademarkserf, etc. In order to create valid print output we can create a transform with a series of string replace directives that together act as a mapping between the special symbols and the actual hex code those symbols represent.

Table 81: Symbol Mapping as a series of Replace String directives.

Symbol	Hex code	Printed character
“&copyrightserif”	a9	©
“&trademarkserf”	aa	™
“&dagger”	a0	†
“&sterling”	a3	£
“&bullet”	a5	•
“&iacute”	92	í

Use the String Replace Directive dialog box, shown in Figure 187, to define the directive. Table 82 describes the contents of the dialog box.

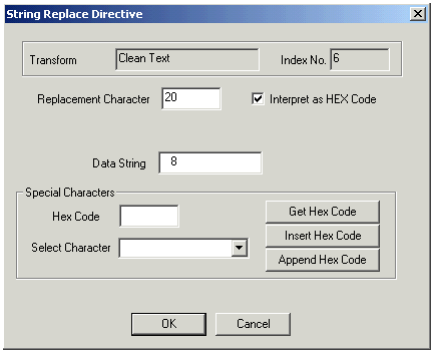


Figure 187: String Replace Directive dialog box

Table 82: Contents of the Character Replace Directive dialog box

Option	Description
Replacement Character	Type the replacement character. If the character is unprintable on the screen, you can paste it in from another document. You can also type in the actual hex code if you know it.
Interpret as HEX Code	With this box checked, PatternStream will interpret the character(s) in the replacement character field as a hexadecimal code rather than the actual character.
Data String	This is the data string you wish to replace.
Special Characters	
Hex Code	To enter a character that is not printable from the keyboard, type in the characters hex code here then click on the **Append Hex Code** or **Insert Hex Code** buttons to include this character in the data string.

Table 82: Contents of the Character Replace Directive dialog box (continued)

Option	Description
Select Character	This drop-down list provides a readily accessible list of commonly used special characters that can be inserted into the data string. Select the desired character, then click on the **Append Hex Code** or **Insert Hex Code** buttons to include this character in the data string.
Get Hex Code	Click this button to display the hex code of the first character selected in the data string. Special characters are usually displayed in dialogs as thick vertical bars. Use this to find out what the hex code of the special character actually is.
Insert Hex Code	Includes the character represented by the value in the Hex Code edit box into the data string at the insertion point.
Append Hex Code	Appends the character represented by the value in the Hex Code edit box at the end of the data string.

Note: A chart containing character hex codes is available in FrameMaker's online help. In FrameMaker, select the **Help** menu, then **Online Manuals,** and then **FrameMaker Character Sets.**)

Substitute String directive

This directive will substitute the occurrences of a particular substring in the value of a string variable with another.

Use the Substitute String Directive dialog box, shown in Figure 188, to define the directive. Table 83 describes the contents of the dialog box.

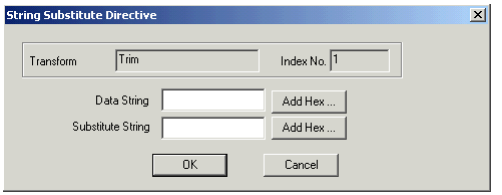


Figure 188: String Substitute Directive dialog box

Table 83: Contents of the String Substitute Directive dialog box

Option	Description
Data String	
Substitute String	
Add Hex button	

String Concatenation directive

Use this directive to concatenate a string to the value of a particular string variable. The string can be either a constant or a variable. If the concatenated string is derived from a non-string variable, then the value is converted to a string using the appropriate format settings before being concatenated to the target variable.

For example, suppose you are trying to publish a catalog from a database where each item has a unique id number, and each item could also have a graphic image associated with it. You could manage this by having the item with id of 4545 have an associated graphic file of “4545.eps”. The filename for each item would then be built by concatenating the variable value of the id with the constant string “.EPS.”

Use the String Concatenation Directive dialog box, shown in Figure 189, to define the directive. Table 84 describes the contents of the dialog box.

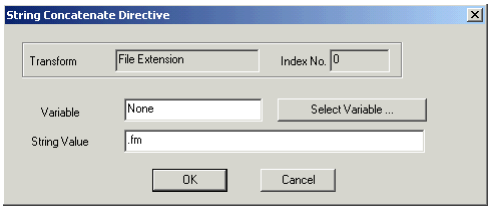


Figure 189: String Concatenation Directive dialog box

Table 84: Contents of the String Substitute Directive dialog box

Option	Description
Variable	Enter the name of the variable whose value you wish to use. If the variable is a non-string type, then the string representation of its value will be used.
Select Variable button	Clicking this button displays the variable selection dialog. There are no restrictions on the type of variable you can use.
String Value	Constant string value that will be concatenated to the target variable. *Note:* Having a valid variable name will override what is in the String Value field.

Convert Variable directives

In order to display a variable in a printed document, its value, whether numeric, date or string must be representable as a string of some kind. For this, a variable uses a format assignment to control how a value will look. However, this string representation is not accessible as value to other objects. You can use the convert variable directives to assign this representation to another (string) variable.

The three Convert Variable directives—Convert Number, Convert Date, and Convert String—share the same dialog box (Figure 190), although the selec-

tions in the Format and Variable drop-down lists differ. Table 85 describes the contents of the dialog box.

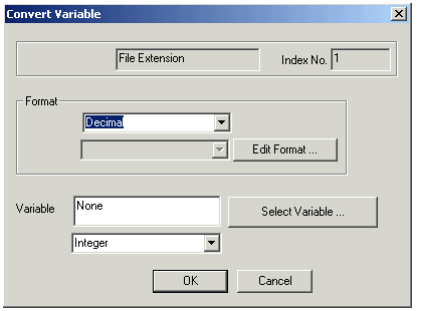


Figure 190: Convert Variable dialog box

Table 85: Contents of the Convert Variable dialog box

Option	Description
Format	Select the type of format object from the drop-down. The possible choices will vary depending on the type of convert variable directive.
Edit Format button	Certain format types have additional parameters. You can access these parameters by clicking on the Edit Format button.
Variable	The name of the variable to convert to a string value.
Select Variable	Clicking on this button displays the Select Variable dialog. Only variable types appropriate for the particular type of convert directive will be displayed.

Convert Number directive

Converts the following type of variables to a string value:

- Integer
- Float
- Big Integer
- Double

Convert Date directive

Converts the following type of variables to a string value:

- Date
- Time
- Timestamp

Convert String directive

Converts the following type of variables to a string value:

- String
- Text

Numeric transforms

The Numeric transform lets you perform a mathematical operation on a value held in a Float, Integer, or Reference type variable (the Reference variable must point to a Float or Integer variable). The result of the operation will be inserted in your document instead of the variable's original value.

Table 86 describes the directive types specific to the Numeric transform.

Note: The Function and Table Lookup directives are common to all transforms and are used only for advanced features. See "Directives for advanced features" on page 336 for details.

Table 86: Numeric transform directive types

Directive	Description	For details see...
Unary Operation 	Applies a constant to the target numeric variable using the indicated arithmetic operation (i.e., increment by one, modulo two, etc.)	"Unary Operation directive" on page 330
Binary Operation 	Applies the current value of the assigned variable to the target numeric variable using the indicated arithmetic operation.	"Binary Operation directive" on page 331
File Exist 	Assigns the target variable a value based on whether a specified file can be opened.	"File Exist directive" on page 331

Table 86: Numeric transform directive types (continued)

Directive	Description	For details see...
String Length 	Sets the target numeric variable the length of the current value of an attached string (or text) variable.	"String Length directive" on page 332
Date Component 	Assigns the target numeric variable the desired component of the value of an attached date, time or timestamp variable.	"Date Component directive" on page 333

Unary Operation directive

This directive applies a basic arithmetic operation to the value of a numeric variable using a specified constant.

Use the Unary Operation Directive dialog box, shown in Figure 191, to define the directive.

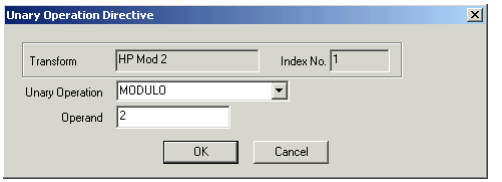


Figure 191: Unary Operation Directive dialog box

Table 87: Contents of the File Exist dialog box

Option	Description
Operand	The value of the constant operand to apply to the numeric variable.
Operation	Select from the drop-down list the desired arithmetic operation. The following are the valid operations: **ADD, SUBTRACT, MODULO, MULTIPLY,** or **DIVIDE.**

Binary Operation directive

Binary operations directives are similar to Unary directives except that the value used in the operation is not a constant but a variable, itself.

Use the Binary Operation Directive dialog box, shown in Figure 192, to define the directive.

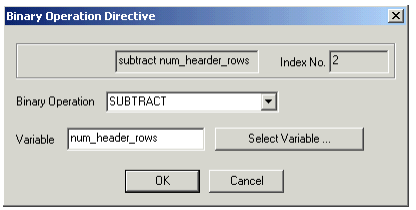


Figure 192: Binary Operation Directive dialog box

Table 88: Contents of the File Exist dialog box

Option	Description
Variable	The name of the variable operand to apply to the target numeric variable.
Select Variable	Click on this button to select the numeric variable to use as the operand.
Operation	Select from the drop-down list the desired arithmetic operation. The following are the valid operations: **ADD, SUBTRACT, MODULO, MULTIPLY,** or **DIVIDE.**

File Exist directive

This directive is used to test for the existence of a specific file. Notice that the various options for specifying a filename are the same as for graphic insets in graphic segments, anchored frame targets and unanchored frame targets. This directive will assign a value of 1 if the currently specified file can be opened and 0 otherwise.

Use the File Exist Directive dialog box, shown in Figure 193, to define the directive. Table 89 describes the contents of the dialog box.

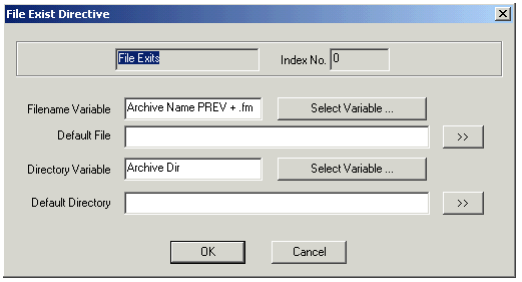


Figure 193: File Exist Directive dialog box

Table 89: Contents of the File Exist dialog box

Option	Description
Filename Variable	The name of the variable that contains the filename of the graphic. To attach a variable to the segment, select the **Select Variable** button and choose a variable from the Select Variable dialog box.
Default File	The name of the file to use if the file specified by the filename variable does not exist. Select the >> button to browse for the file.
Directory Variable	The name of the variable that contains the subdirectory where the graphic file resides.
Default Directory	The directory where the image files are located.

String Length directive

Sets the value of the target numeric variable to the actual string length of the current value of a specified string variable.

Use the String Length Directive dialog box, shown in Figure 194, to define the directive.

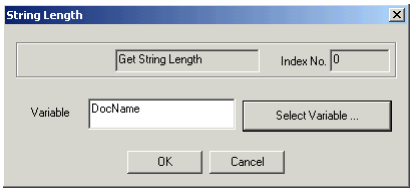


Figure 194: String Length Directive dialog box

Table 90: Contents of the String Length Directive dialog box

Option	Description
Select Variable button	To specify the variable containing the string length, select the **Select Variable** button and choose a variable from the list. Only string and text variables are valid.

Date Component directive

This directive assigns value of a date component for a specified variable to a target numeric variable. The specified variable can be a date, time or timestamp variable.

Use the File Exist Directive dialog box, shown in Figure 195, to define the directive. Table 89 describes the contents of the dialog box.

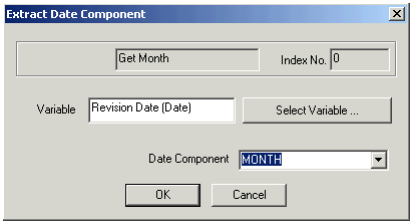


Figure 195: Extract Date Component dialog box

Table 91: Contents of the Date Component Directive dialog box

Option	Description
Date Component	Select the date component you wish to extract from the assigned variable.
Select Variable button	To specify the variable wish to extract a component from, click the **Select Variable** button and choose a variable from the list. Only date, time and timestamp variables are valid.

Date transforms

The Date transform lets you perform a mathematical operation on a value held in a Date, Time, or Timestamp variable. Table 92 describes the directive types specific to the Date transform.

Note: The Function and Table Lookup directives are common to all transforms and are used only for advanced features. See "Directives for advanced features" on page 336 for details.

Table 92: Date transform directive types

Directive	Description	For details see...
Binary Operation 	Performs a numeric operation on a date, time or timestamp variable.	
Current Time 	Assigns the target variable the value of the current system time.	

Segment transforms

The Segment transform works in conjunction with the Text Markup segment of a string template to specify special formatting (such as bold or superscript) to a section of text in your document. For example, you are extracting a large block of text from the database, and within that block of text are words that need to be italicized. This special formatting must already be marked in the database (where the markup might look like `<I> word </I>`); you use the Segment transform to

specify the begin and end tokens and the formatting to be applied to the text within the tokens.

The Segment transform is composed of Parse Markup directives, one for each instance of special formatting in the text block. You apply this transform to a string template's Text Markup segment, in which you have specified the variable bound to the chunk of text in question. (For more information on the Text Markup segment, see "Defining a Text Markup segment" on page 284.)

When the pattern set is run, the PatternStream application parses the chunk of text and applies the appropriate character formats to the text between the specified tokens. The tokens are not displayed in your final document.

Only one directive, Parse MarkUp, is available for a Segment transform.

Use the Parse MarkUp Directive dialog box, shown in Figure 196, to define the directive. Table 93 describes the contents of the dialog box.

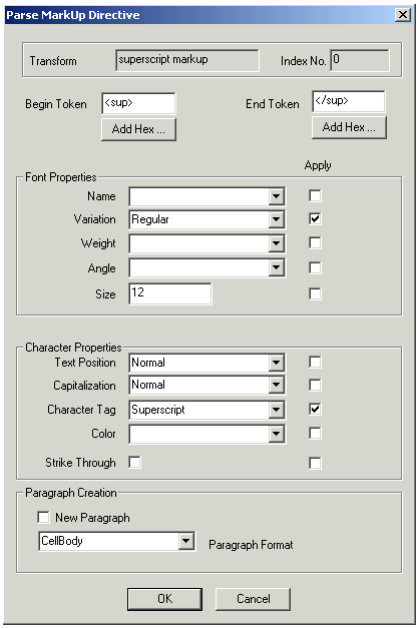
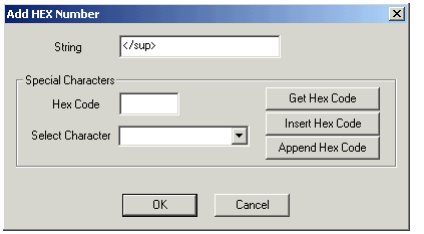


Figure 196: Parse MarkUp Directive dialog box

Table 93: Contents of the Parse MarkUp dialog box

Option	Description
Begin/End Token	Specify the begin and end tokens used in the source text to mark the text to be formatted. To use an extended character as a token, select the **Add Hex** button and specify the string.
Add Hex button	Clicking this button displays the Add Hex Number dialog box.  Type the string in the String field. Select an extended character from the Select Character list or type a character's hex code in the Hex Code field. Select the appropriate button to insert the code into the string.
Font Properties/ Character Properties fields	Specify the formatting attributes you want to apply to the text between the tokens, then check the corresponding **Apply** check box.
Paragraph Creation	If you want the markup to specify a paragraph break, select the **New Paragraph** check box, then select a paragraph format from the Paragraph format drop-down list.

Directives for advanced features

Lookup Directive

This directive is used to extract a value from a lookup table. A lookup table is an extension object that stores data values which can be retrieved for later use using a multi-valued lookup key. The lookup table (or associative array) mechanism is also described in "Lookup target" on page 250 and in "Lookup Table" on page 353, but because this is one of more difficult topics, we offer another way

to understand it. Table 94 lists the objects that cooperate in the lookup mechanism.

Table 94: Objects Types Required for the Lookup Table Mechanism

Directive	Class	Description
Lookup Table 	Extension	Object that actually stores the entries and defines the table.
Lookup Target 	Target	This target generates an entry into the attached lookup table whenever it is invoked.
Argument List 	Parameter List	A set of variables that defines the value of the key for a specific entry. The current value of the variables in the argument list are used in two ways: • defines the key when inserting an entry. • defines the key when extracting the value of an entry. *Note:* Do NOT use a lookup table as an item in an argument list (which is permitted) if that argument list is going to be used in a lookup table mechanism.
Lookup Directive 	Transform/ Directive	Used to extract a value from a lookup table based on the value of a key.

Use the Lookup Directive dialog box, shown in Figure 194, to define the

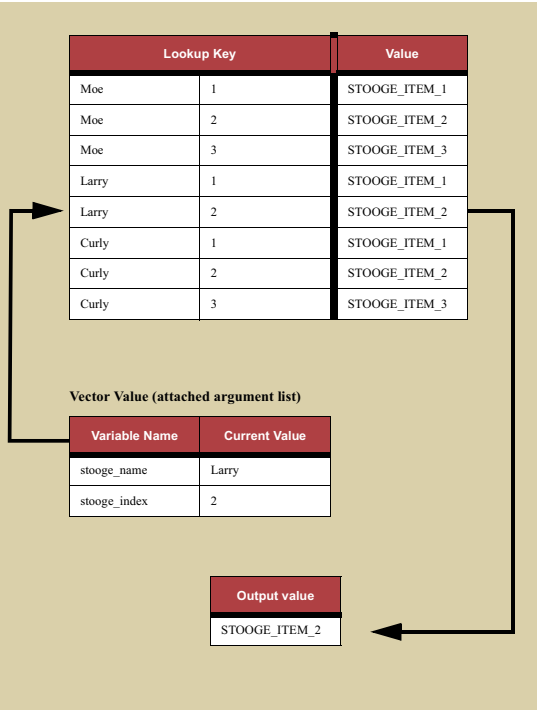


Figure 197: How the Lookup Directive Works

directive.

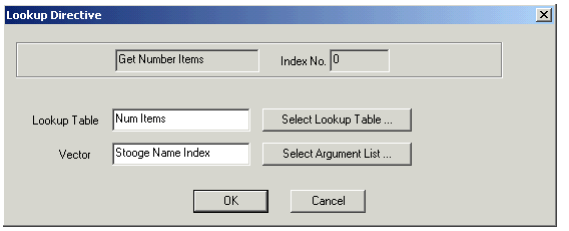


Figure 198: Lookup Directive dialog box

Table 95: Contents of the String Length Directive dialog box

Option	Description
Lookup Table	The name of the lookup table from which to extract the values from.
Select Lookup Table...	Click on this button to bring the Select Lookup table dialog.
Vector	The argument list that will define the value of the lookup key whenever the directive is invoked.
Select Argument List...	Click on this button to bring up the select argument list dialog box.

Chapter 18: Creating parameter lists

Parameter lists are lists of variables used by other objects in the PatternStream application for specialized tasks, such as generating table columns when the number and size of the columns is data-driven instead of fixed.

This chapter describes how to create a parameter list and define each of the three list types.

Working with Parameter Lists

Parameter lists are accessed through the **Conditions/PLists** tab of the PatternStream main dialog. (Figure 199).

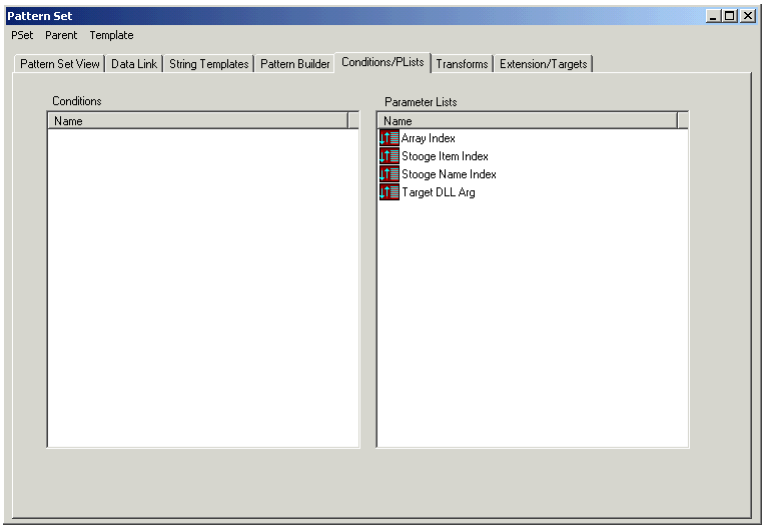


Figure 199: Conditions/PLists tab

Creating a parameter list

To create a parameter list, follow these steps:

1 Click on the white area of the plist window with the right mouse button and select New Object from the popup menu. The Create New Parameter List Object dialog box is displayed (Figure 200).

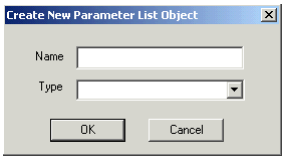


Figure 200: Create New Parameter List Object dialog box

2 In the Name field, type the name you want for the parameter list.

3 Select the desired type of parameter list. The various types are listed in Table 96.

Table 96: Parameter list types

Type	Attach to...	Use when...	For details, see...
Argument List 			"Defining an Argument List" on page 345
Index Heading List 	Index segment of a string template	your index headings and subheadings are data-driven.	"Defining an Index Heading List" on page 345
Table Column List 	a TABLE target (from the **Format** tab of the Table Target Properties dialog box)	the number of columns in a table you're creating is data-driven rather than fixed. Determines if a column is created and its width in the table.	"Defining a Table Column List" on page 347

Copying a parameter list

If you are creating a parameter list that is similar to an existing one, you can save time by copy the existing one and modifying its properties.

To copy a parameter list, follow these steps:

1 From the **Conditions/PList** tab, select the parameter list you want to copy.

2 Right mouse-click, then select **Copy** from the object popup menu.

3 The Copy Parameter List Object dialog box is displayed, showing the name of the copied parameter list (Figure 201).

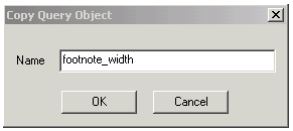


Figure 201: Copy Parameter List Object dialog box

4 Type the name you want for the new condition, then select **OK**. The condition properties dialog for the type of query copied will be displayed.

5 Modified the attributes appropriately then close the dialog.

6 The new parameter list now appears with the other parameter list define for the current patter set on the **Conditions/PList** tab.

Finding references

Many times it is necessary to find out which objects are using a particular parameter list. To find these references do one of the following:

- Select the **Conditions/PList** tab of the PatternStream main dialog, select the particular parameter list, right-click and select **References** from the object popup menu.

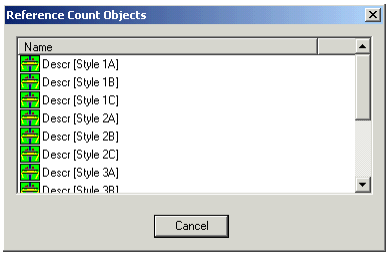


Figure 202: Parameter List Reference dialog

- Select the Pattern Set View tab of the PatternStream main dialog. Below the PSET hierarchy, there is a folder containing a list of all parameter list objects for the current pattern set in alphabetical order. Select the desired parameter

list object, click on the right mouse button, then select **References** from the popup menu.

Figure 203: Parameter List list from the Pattern Set View tab.

Deleting a parameter list

Before you can delete a parameter list object, you must detach it from all of the objects that reference it.

To delete a parameter list, follow these steps:

1. Select the particular query from the alphabetized list of parameter list objects in either:
   - the query window in the upper left corner of the **Conditions/PLists** tab
   - the parameter list folder on the **Pattern Set View** tab.
2. Click the right mouse button and select **Delete** from the popup menu.

Defining an Argument List

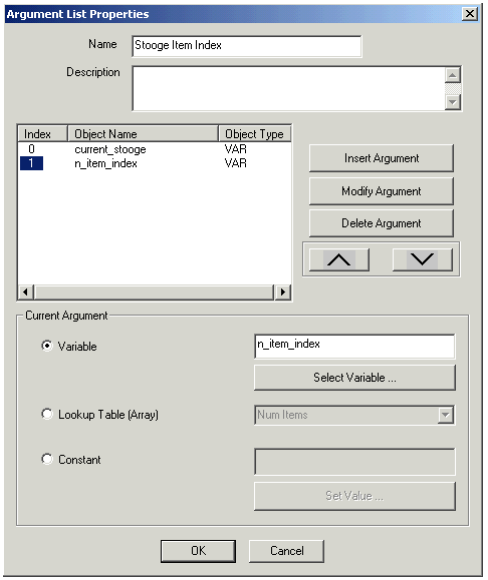


Figure 204: Argument List Properties dialog box

Defining an Index Heading List

Use Index Heading Lists when the headings (Level1IX in FrameMaker) and/or subheadings (Level2IX and Level3IX) of your document's index are data-driven. You specify the variables for the headings and subheadings in the Index Heading Properties dialog box, then attach the Index Heading List to the Index segment of a string template. Once the query returns the values of the variables in the Index Heading List, the segment refers to the list to generate the appropriate FrameMaker index marker syntax.

Example 5 shows a typical use of an Index Heading List.

Example 5: Using an Index Heading List

In this example, a pet supply company is producing a catalog with an index listing supplies by name and under the pet to which they apply. For example, "flea shampoo" would be listed both as a Level1IX entry and as subentries (Level2IX) of "cats" and "dogs."

To generate the Level1IX entries, use an Index segment in a string template. The variable attached to the segment is bound to the Item column in the database.

Example 5: Using an Index Heading List (continued)

Similarly, to generate the Level2IX entries, complete the Index Segment Properties dialog box the same as you would for the Level1IX entries; in addition, create an Index Heading List and attach it to the segment.

The variable `level1` is bound to the database column that identifies the supply's category. Once the value for this variable (and for the variable associated with the main entry) is returned from the database, the PatternStream application can construct and insert a marker that the FrameMaker application can understand.

For example, for one specific marker location, `level1` returns the string "dogs," and the variable attached to the Item column returns the string "flea shampoo." The PatternStream application can now create the index marker `dogs:flea shampoo` and insert it in the specified location in the document.

You use the **Index Heading Properties** dialog to define an index heading list. (Figure 205.

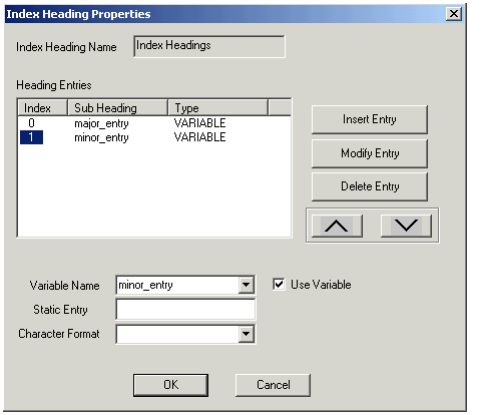


Figure 205: Index Heading Properties dialog box

The attributes you can assign to each heading entry are listed in Table 97.

Table 97: Index heading attributes

Attribute Name	Description
Variable Name	
Static Entry	
Character Format	

Table 98: Modifying the index heading list

Operation	Procedure
Create a heading	**1** select the appropriate variable from the Variable Name drop-down list and then click on the **Use Variable** check box. or If the entry is not data-driven, type the entry into the Static Entry field **2** To assign a character format to the entry (such as italics), select a format from the Character Format drop-down list. **3** Click the **Insert Entry** button. The new index heading entry will be added to the list.
Modify a heading	**1** Select the desired index heading entry by clicking on its index number in the entry list. The values for the select entry will be shown in the current entry area. **2** Modify the variable name, static entry, or character format as desired. **3** Click the **Modify Entry** button.
Delete a heading	**1** Select the desired index heading entry by clicking on its index number in the entry list. **2** Click on the **Delete Entry** button.

Defining a Table Column List

Most of the tables you create through the PatternStream application have a fixed number of columns. However, if the number of columns used is data-driven, you must use a Table Column List in conjunction with VARCELL targets. In the Table Column

List, you bind variables to the column names of a table in the database and define the absolute or relative width of the column in the table you generate.

For example, your document consists of several similar tables, all of which derive their data from the same database columns. The tables have 10 possible columns, but you want the column to appear only if it applies to the specific table. In this case, you would define the column name and its associated variable for each of the 10 possible columns. The column will be generated for a specific table only if its associated variable's returned value is greater than zero.

You attach Table Column Lists to a TABLE target through the **Format** tab of the target's Table Target Properties dialog box. VARCELL targets determine the content of the cells of the table. You define the location of the VARCELL target by specifying the name of the column where you want the target's content placed. If the column is not created, the VARCELL target is not generated. See "Cell targets" on page 206 for details.

Note: Before you can create VARCELL targets, you must first create and define the Table Column List and attach it to the VARCELL targets' parent TABLE target.

The attributes you can assign to each table column are listed inTable 97.

Table 99: Index heading attributes

Attribute Name	Description
Variable Name	
Column Name	
Column Width	
Relative vs. Absolute	

Chapter 19: Creating extensions

The Extension Class

Extensions are PatternStream objects that do not fall into any well defined category or perform similar functions. They are not part of the pset hierarchy, but frequently must cooperate with objects that are (i.e., patterns and targets).

The extensions described in this chapter are listed in Table 100.

Table 100: Extensions

Target	Description	For details see...
Lookup Table 	Object that stores data into a table (or associative array) that can be retrieved at a later time.	"Lookup Table" on page 353
PSet Tree 	Mechanism that allows you to use a pattern in more than one position in the pset hierarchy.	"PSet Tree" on page 355
Log File 	Opens an ASCII text file for auxiliary output of text generated by a string template.	"Log File" on page 357
Query DLL 	Connects with an exported set of subroutines from an external DLL to create a user defined query object.	See *PatternStream Programming Manual*
Scalar Function 	Connects with an exported function from an external DLL to create a user defined function that can operate on the value of any PatternStream variable.	See *PatternStream Programming Manual*
Target DLL 	Connects with an exported function from an external DLL which is then attached to a target object and allows costume processing of the generated structural element. These functions are typically FDK clients.	See *PatternStream Programming Manual*

Table 100: Extensions

Target	Description	For details see...
Grid Algorithm 	Connects with an exported set of functions from an external DLL which allows for user defined algorithms for placing unanchored frames on an output page using the grid method.	See *PatternStream Programming Manual*
FrameScript 	Contains a FrameScript script that can be attached to a target object and is executed when that target object is invoked.	See *PatternStream Programming Manual*

Creating an extension

To create an extension, follow these steps:

1 Select the **Extensions/Targets** tab (Figure 206).

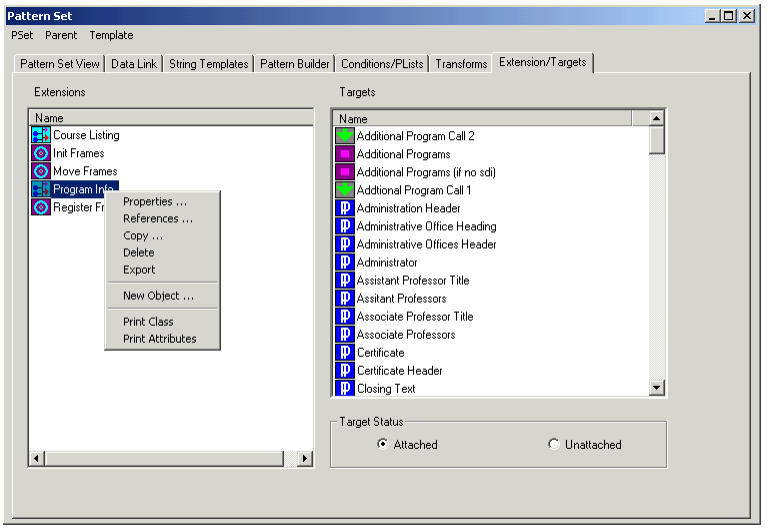


Figure 206: Extensions/Targets tab

2 Place the cursor in the white area of the extensions window and click with the right mouse button.

3 Select New Object from the popup menu.

4 The Create New Extension dialog is displayed.

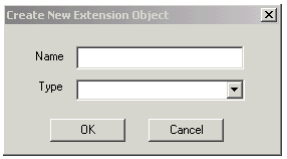


Figure 207: Lookup Table Properties dialog box

5 Type in the name of the new extension.

6 Select the type of extension you wish to create from the Type drop-down list.

Note: Although the different types of extension may be unrelated, the name you choose must be unique within the entire extension class.

Working with extension objects

Extension object can be accessed in two ways:

- First select the Extensions/Targets tab of the PatternStream main dialog. Then select the desired extension object and click on the right mouse button. From the popup menu, you can choose the operation you wish to perform.
- From the Pattern View tab of the PatternStream main dialog, scroll down to the Extensions folder. Select the desired extension object then click on the right mouse button. Choose the operation you want to perform from the popup menu.

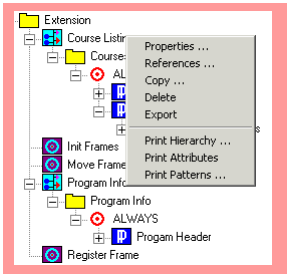


Figure 208: Lookup Table Properties dialog box

Editing extension properties

Select Properties from the object popup menu, or double click on the extension object's icon. This brings up the properties dialog for the selected extension type.

Viewing references

- To find out what other objects are using a given extension, select References from the object popup menu. Figure 209 is an example of the reference list of a Lookup extension.

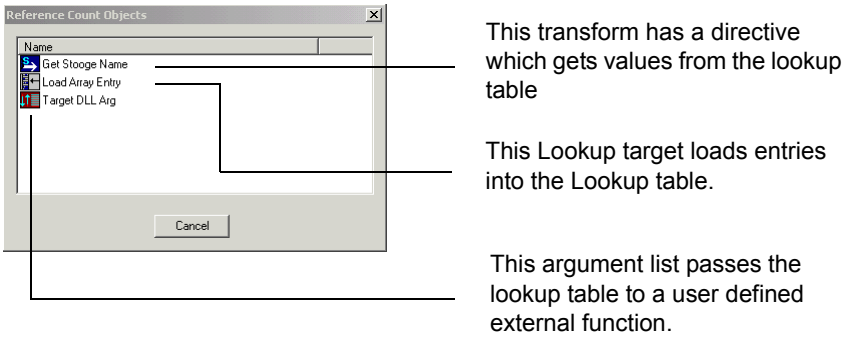


Figure 209: Extension Reference List

Deleting an extension object

To delete an extension object, select the object's icon, click the right mouse button, and select Delete from the popup menu.

Note: You must reduce the reference count to zero before you can delete an extension by detaching it from all objects that reference it. See View references to determine what these objects are.

Copying an extension object

1 To delete an extension object, select the object's icon, click the right mouse button, and select Copy from the popup menu. This will bring up the Copy Exten-

sion dialog, shown in figure Figure 210. The name to something different, then click OK.

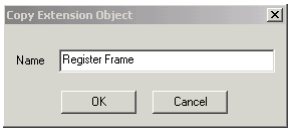


Figure 210: Copy Extension dialog box

Exporting an extension object

To delete an extension object, select the object's icon, click the right mouse button, and select Export from the popup menu.

Lookup Table

A Lookup Table is a multi-key associative array that is used to store data from a query in a manner that can be extracted for use later in the output flow. A type of target called a Lookup target is used to insert entries into the lookup table (see "Lookup target" on page 250 for more details.). To extract a value from the lookup table, Lookup directives are used (see "Lookup Directive" on page 336 for details).

In order to define the value of the key for each entry, Lookup tables use an object called an argument list. This is an ordered set of PatternStream variables (see "Defining an Argument List" on page 345 for more details). At any time, these variable can be evaluated, creating an n-tuple of values. Figure 211 illustrates how the current values of the variables in the argument list determine the lookup key and hence a specific entry in the lookup table. Notice that in the example, the length of the vector is 2, reflecting the fact that there are 2 variables in the argument list.

Note: It is important that the vector length, declared in the definition of the lookup table, is consistent the number of vectors in the argument list(s) being used.

Argument lists are used to reference existing entries and to insert new entries into the lookup table. Because it is the value of the variables in the argument list that determines the key, more than one argument list object can be used to reference the same lookup table.

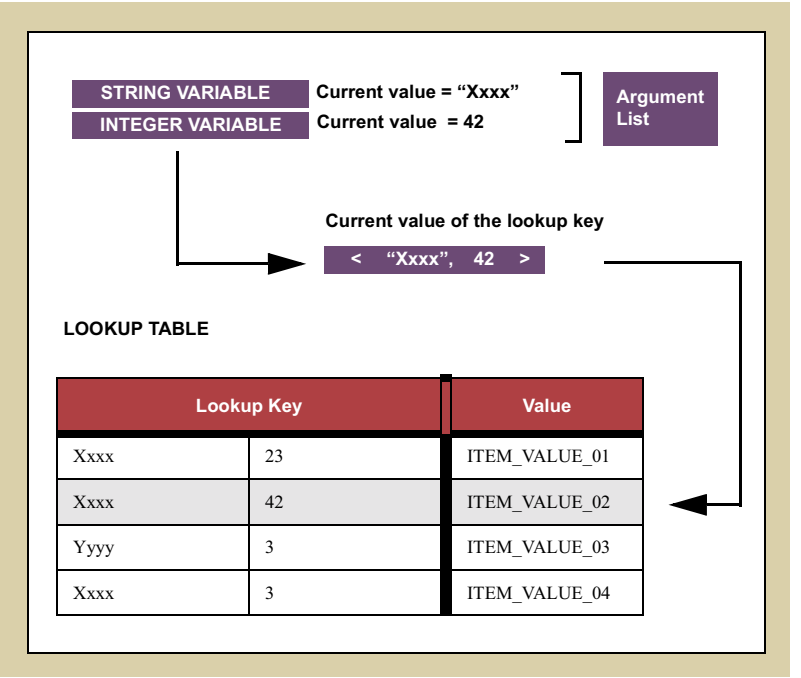


Figure 211: Argument List is used to determine the lookup key.

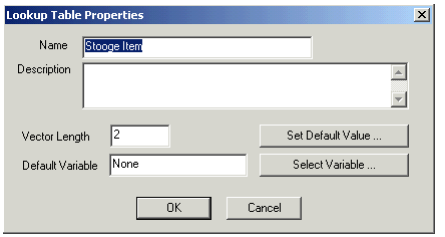


Figure 212: Lookup Table Properties dialog box

Table 101: Contents of the Lookup Table Properties dialog box

Option	Description
Name	Enter the name of the Lookup Table object. *Note:* The name must be unique for the class of extension objects, not just for lookup table objects.
Description	Enter a description if desired, which will be included in the object's attribute printout.
Vector Length	The number of variables in the lookup key. This number must correspond to the number of variables in the argument list(s) used to reference entries in the lookup table.
Select Default Value...	Displays the Value dialog. The default value is used whenever an entry for a specific key value does not exist.
Default Variable	The name of the variable that contains the default value.
Select Variable...	Displays the select variable dialog. Use this to choose a variable to use as the default.

PSet Tree

Sometimes it is necessary to use the same complex sub-pattern in more than one position in the pset hierarchy. Because of the restriction that a pattern can be used only once, it normally would be necessary to make as many copies of the sub-pattern as is needed. This not only complicates the name space of the pattern set, but any changes made would have to be implemented in multiple places.

Using a pset tree allows you to circumvent this restriction. Rather than attaching a sub-pattern to a target object, you instead attach it to a pset tree object. This pset tree object can then be attached to more than one call target object. When a call target is invoked, its associated pset tree is invoked, which then executes the

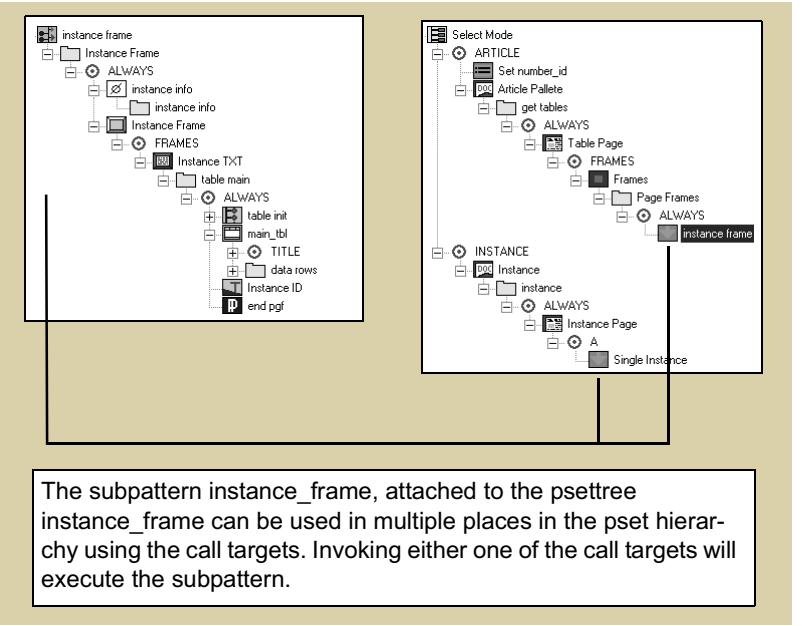


Figure 213: PSetTree and Call Target Mechanism.

attached sub-pattern. This mechanism is somewhat analogous to a subroutine call or a macro. (see Chapter , “Call target” for more details on call targets)

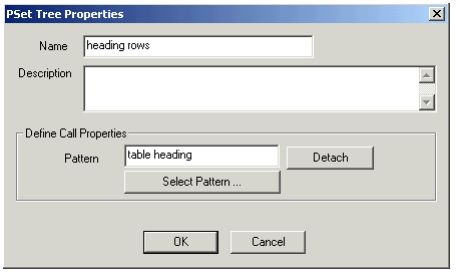


Figure 214: PSet Tree Properties dialog box

Table 102: Contents of the PSetTree dialog box

Option	Description
Name	Enter the name of the PSetTree object. *Note:* The name must be unique for the class of extension objects, not just for psettree objects.
Description	Enter a description if desired, which will be included in the object's attribute printout.
Pattern	The name of the pattern to attach to the PSetTree.
Select Pattern	Displays the Select Pattern dialog. *Note:* Only unattached patterns will appear in the this dialog.
Detach	Click this button to detach the current pattern.

Log File

Sometime is useful to output text to a secondary ASCII file. The Log File extension works in conjunction with LogFile target objects to generate ASCII output. The filename can be a constant string or be data-driven, using the value of an attached variable to determine the output filename. The LogFile object works in conjunction with the Log Target (see "Log target" on page 182). Log File targets use attached string templates to create an ASCII text string, which is then written to the opened log file.

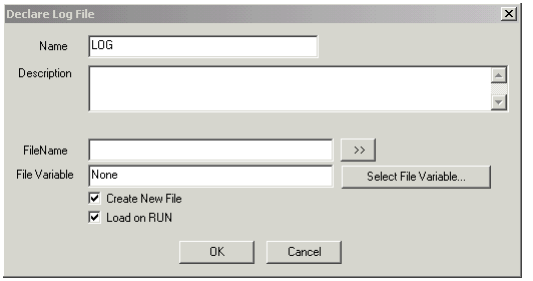


Figure 215: Declare Log File dialog box

Table 103: Contents of the Lookup Table Properties dialog box

Option	Description
Name	Enter the name of the Lookup Table object. *Note:* The name must be unique for the class of extension objects, not just for lookup table objects.
Description	Enter a description if desired, which will be included in the object's attribute printout.
Filename	The name of the log file.
File Variable	The name of the variable that has a current value will be used for the log file name.
Select Variable...	Displays the select variable dialog. Click on this button to choose a variable that will contain the filename.
Create New File	This flag determines whether PatternStream will start on new log file during each run, or append text to an existing file. *Note:* If this box is unchecked and the declared file does NOT exist then PatternStream will generate an error.
Load on Run	Un-check this parameter if you wish the log file to be opened at the beginning of the run.

FINITE MATTERS LTD.

• INFORMATION MANAGEMENT SOLUTIONS •

Main Office: 3064 River Road West • Suite B • Goochland, Virginia 23063
Sales Office: 3064 River Road West • Suite B • Goochland, Virginia 23063
North Carolina Office: 308 W. Rosemary Street • Suite 303A • Chapel Hill, NC 27516-2548
Mailing Address: P.O. Box 759 • Goochland, Virginia 23063
Toll Free: 1-888-230-1FML (1365) • **Voice:** 804-556-1180 • **Fax:** 804-556-1183
email: info@fml.com • **URL:** http://www.fml.com