

# Informed Designer<sup>®</sup>

## Forms Automation

Shana Corporation  
9744 - 45th Avenue  
Edmonton, Alberta, Canada  
T6E 5C5

Telephone: (403) 433-3690  
Fax: (403) 437-4381  
e-mail: [info@shana.com](mailto:info@shana.com)  
Web: <http://www.shana.com>



© Copyright Shana Corporation 1996  
All rights reserved.

Informed, Informed Designer, and Informed Manager are registered trademarks of Shana Corporation. Informed logo and Informed Filler are trademarks of Shana Corporation.

Portions of this software are copyright 1991-1996 by INTERSOLV, Inc.

International ProofReader™ English text proofing system © 1995 by INSO Corporation. All rights reserved. Reproduction or disassembly of embodied algorithms or database prohibited.

Portions of this software are copyright 1996 Altura Software, Inc.

QuickHelp™ is licensed from Altura Software, Inc. End-User is prohibited from taking any action to derive a source code equivalent of QuickHelp, including reverse assembly or reverse compilation.

All other trademarks are property of their respective owners.

00-0551-000  
9-4-96



Introduction

## Introduction

With the advancement of computer technology and forms software, electronic forms offer an efficient, intelligent, and secure alternative to the traditional paper based forms systems used in many organizations. Shana Corporation offers the complete cross-platform electronic forms solution for each step in the paper forms process—from the design stage to “fill, sign, and send.”

Informed Designer and Informed Filler together provide everything you need to design, distribute, fill, route, approve, submit, and track electronic forms.

### Informed Designer

Informed Designer gives you flexible tools to draw professional quality forms quickly and easily. With advanced drawing tools, powerful graphics manipulation commands, and precision control, you have all you need to produce appealing, picture perfect forms on your computer. Since forms are stored electronically, you now have more freedom to change your forms as your needs change.

If you want to design paper forms to be filled out by hand or with a typewriter, then Informed Designer is all you need. You can design forms for in-house printing, or you can prepare camera-ready artwork for your commercial printer.

When you're ready to move from paper forms to an electronic forms solution, Informed Designer provides all the features you need to design intelligent form templates, ready to be filled out electronically with Informed Filler. Use Informed Designer's powerful tools and functions to make your forms automatically calculate, format, lookup, and check information for the person using Informed Filler. You can also configure forms for electronic signing by using digital signatures.

Although Informed Filler is required to fill out and save forms electronically, you can test your forms using Informed Designer's Test mode. This allows you to make sure that your formatting, calculations, and other intelligent features work properly without having to switch to a different application.

### The Informed Designer Manual Set

The Informed Designer manual set is designed to provide you with a complete reference to the features and capabilities of Informed Designer. The manuals combine text and graphics to thoroughly document every aspect of the software. In addition to your Informed Designer Forms Automation manual, the set also contains the following:

- The *Informed Designer Getting Started Guide* provides you with information on installing and registering Informed Designer, and also describes the minimum hardware and software configurations required to use the Informed Designer application.

- The *Informed Designer Design and Graphics* manual provides a complete reference to Informed Designer's design and graphics features. Instructions are given for every step in the form design process, including preparing the drawing area of the template, using the drawing tools, adding graphics, and printing. You also learn how to mail form templates using an electronic mail system.

## About This Manual

This manual includes a complete reference to Informed Designer's forms automation features. General topics are organized in the following chapters:

- Chapter 1, "Adding Intelligence To Your Forms," describes Informed Designer's data handling capabilities. You'll learn how to add "intelligent" features to your form templates.
- Chapter 2, "Using Digital Signatures," explains Informed's digital signature capabilities, and describes how to configure a template for electronic signing.
- Chapter 3, "Customizing Menus," provides information on how to configure a form's menus to create a simpler and more familiar environment for the Informed Filler user. You'll also learn about the different menu item types and their uses.
- Chapter 4, "Using Buttons," describes Informed Designer's Button tool and how it can be used to draw and configure buttons on a form to perform specific actions
- Chapter 5, "Routing," explains how to configure suggested routes for your templates to aid the Informed Filler user in addressing and mailing completed forms.
- Chapter 6, "Form Tracking," introduces Informed's form tracking feature and explains how to configure forms for tracking throughout your organization.
- Chapter 7, "Authorizing Templates," explains how to protect the integrity of electronic forms by authorizing templates for use in your organization.
- Chapter 8, "Form Template Distribution and Revision," describes Informed's built-in template distribution features and provides guidelines for distributing both new form templates and new revisions of templates.
- Chapter 9, "Using Formulas," teaches you how to create formulas and describes how they are used to calculate and check information, and provide dynamic tabbing on a form.
- Chapter 10, "Using Functions," describes each of Informed's powerful functions and how they help you create sophisticated formulas.
- Chapter 11, "Using Informed Number Server," teaches you how to configure Informed Number Server for use with templates filled out with Informed Filler.
- Chapter 12, "Using AppleScript," provides an overview of Informed's AppleScript scripting capabilities and describes in detail how you can customize forms using AppleScript. You'll

learn how to use Informed Designer's Scripts command to attach scripts to templates so that they can be configured to run when the Informed Filler user invokes certain actions.

- Chapter 13, "Informed 4D Externals," explains how to configure lookups and form submission with any 4th DIMENSION database.

## Conventions Used in This Manual

This section describes the conventions used in this manual to ensure that you can easily find and understand the information you need to perform specific tasks with Informed Designer.

### Finding Information

In addition to the table of contents at the beginning of this manual, you'll also find a table of contents at the beginning of each chapter, listing the main sections in that chapter. The example below shows the table of contents for Chapter 5, "Routing."

- Suggested Routes 5-3
- Suggested Routes for Multiple Platforms 5-3
- Adding, Changing, and Removing Suggested Routes 5-4
- Controlling the Data Format 5-8
- Using Mail Cells 5-9

Inside each chapter, the main topics are highlighted in a gray bar like the one at the beginning of this section, making it easy for you to quickly scan a page to find the topic of your choice. Subsections for each topic are highlighted with a large, bold font.

### Notes

Throughout this manual, you'll see paragraphs of text highlighted in gray boxes with the label "Note" in the left margin. These notes contain important information such as warnings, reminders, and specific conditions to be aware of. The example below shows a typical note.

**Note** Important information about Informed Designer appears in highlighted gray boxes like this one.

### Commands and Control Names

When specific instructions on how to perform a certain task are given in this manual, commands are shown in a different typeface from the rest of the text. The name of the menu where the command is found is also given in each instance. For example, when learning how to use Informed Designer's Cell palette, you'll read the following text:

“You can show the Cell palette by choosing **Cell Palette** from the Show submenu under Layout. To hide the Cell palette, click its close box or choose **Cell Palette** again.”

The names of controls such as buttons and settings on dialog boxes are always shown in single quotes. For lengthy control names, this helps to differentiate the control name from the text that surrounds it. For example:

“When setting preferences for Informed Designer’s spell checking feature, you can select the ‘Always provide alternative spellings’ checkbox.”

## Cross-platform Issues

Although this manual has tried to be platform neutral, the cross-platform nature of Informed Designer requires that special care be taken when documenting certain features of Informed Designer.

Throughout this manual you’ll see screens of dialog boxes and windows. Some of the screens show Windows dialog boxes and windows, others are from the Mac OS. In cases where a dialog box or window is substantially different between the two platforms, both versions are shown.

When a specific feature of Informed Designer is only applicable to one platform (Windows or Mac OS), an icon depicting either the Windows or Mac OS platform is displayed in the left margin next to the description of the feature as shown below.



The Windows Metafile format is not supported on the Mac OS.





# 1

## Adding Intelligence To Your Forms

In this chapter:

- Overview 1-2
- Cells 1-4
- Type Options 1-6
- Entry Options 1-7
- Tab Order 1-9
- Cell Types 1-15
- Indexes 1-33
- Calculations 1-35
- Default Values 1-37
- Auto-incrementing Numbers 1-39
- Using Lookups 1-47
- Data Verification 1-63
- Choices 1-68
- Cell Help 1-72
- Form Submission 1-73
- Using the Cell Palette 1-86
- Cell Report 1-90
- Testing Your Form Template 1-91



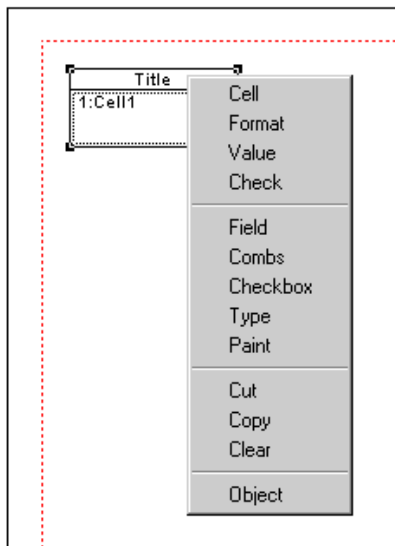
You add intelligent features to your form by using the Cell, Format, Value, Check, Conditional Tabbing, Lookup, and Help Message commands. These commands are found in the Settings menu.

#### Data Intelligence Commands

Command	Description
Cell	Use the Cell command to set the name and tab position of a cell. You can also configure type options, data entry options, and attach choice lists to make it easier for Informed Filler users to enter the cell's value.
Format	Use the Format command to choose the type of information that a cell will hold. The nine allowable cell types are text, character, number, name, date, time, boolean, picture, and signature. Each cell type offers different formatting options for controlling the exact format.
Value	You can set up formulas so that certain cells are automatically calculated or filled in with default values. Informed Filler can automatically increment a number each time the user fills out a new record. This is useful for numbers such as statement or invoice numbers.
Check	Use the Check command to enter error checking criteria so that Informed Filler can automatically check for errors when the user fills out a form. For example, you might want to restrict the discount amount on a sales slip to no more than ten dollars.
Conditional Tabbing	Not only can you customize the tabbing order of cells on a form, you can also configure dynamic tabbing so that the tab order changes depending on different conditions.
Lookup	With lookups, you can configure a form to read information from other forms and other information systems such as SQL databases. This powerful feature allows you to integrate a set of related forms and link them to other information services in your organization. For example, you could read inventory or customer information into your invoice form.
Help Message	You can provide a custom help message for each cell on your form. Help messages can provide useful information to the person who fills out the form.

The commands described in this chapter are accessible from menus and are selected in the normal way: By using the mouse or by typing keyboard equivalents where they exist.

On both Windows and Mac OS compatible computers, clicking the mouse button provides a shortcut to selecting various commands. With any tool selected, position the pointer over an object and click the right mouse button (Windows) or click the mouse button while pressing the Ctrl key (Mac OS) to display a pop-up menu. The pop-up menu contains the settings commands that are applicable to the type of object. For example, if you position the pointer over a field object and click the right mouse button, you'll see a pop-up menu containing the Cell, Format, Value, Check, Field, Combs, Checkbox, Type, Paint, Cut, Copy, Clear, and Object commands.



To select a command, drag the pointer to highlight the command, then click or release the mouse button.

The Cell palette also provides shortcuts to many commands. For a detailed description of the Cell Palette, see “Using the Cell Palette.”

## Cells

You create a cell each time you draw a field or place a new column in a table. Field and table cells are identical in all respects except one: a field cell holds a single value, whereas a table cell can hold multiple values, one for each row in the table. Remember though, you can’t name or format the individual rows in a table cell differently; all rows in a table cell have the same name, type, and formatting options.

You can display or hide the cells in the drawing window. When a cell is showing, its name and tab position (see next sections) are displayed with the type attributes set for that cell. The figure below illustrates a field with and without its cell showing.



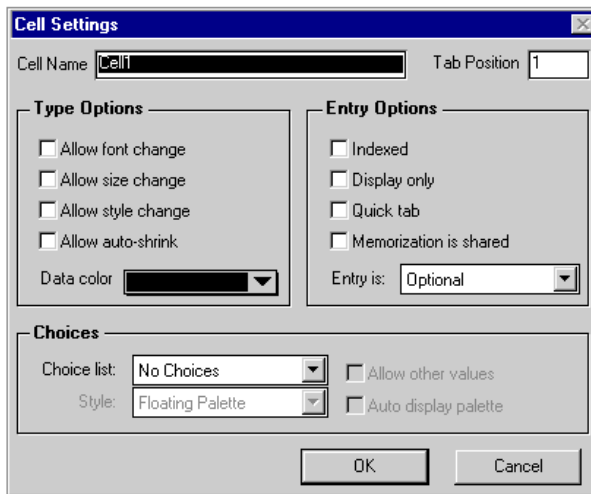
To display the cells in the drawing window, choose **Cell Names** from the Show submenu under Layout. When the cells are displayed, you’ll see a checkmark beside Cell Names indicating that the cells are showing. To hide the cells, choose **Cell Names** again to remove the checkmark.

For information about the appearance of fields and tables, see “The Field Tool” and “The Table Tool,” in Chapter 6 of your *Informed Designer Design and Graphics* manual.

## Cell Names

Each cell on your form must be named uniquely. When you create a new cell, Informed Designer automatically names the cell ‘CellX,’ where X is the next available number. It’s easier to recognize a cell if you give it a descriptive name (a cell called ‘Discount’ is more recognizable than one called ‘Cell31’).

You can change a cell’s name using the Cell command. To change a cell’s name, select the cell that you want to rename, then choose **Cell...** from the Settings menu. This dialog box will appear:



Type the cell’s new name in the ‘Cell name’ text box and click ‘OK.’ When you click ‘OK,’ Informed Designer will check that the new name is valid according to these rules:

- A cell name must begin with a letter (a-z, or A-Z).
- Each character in a cell name must be a letter, number, space, or underscore character (\_).
- A cell name can be no longer than 255 characters.
- A cell name cannot contain any reserved words.

If Informed Designer detects an error in the cell name, or if another cell already has that name, you’ll be alerted with a message.

Informed Designer’s Cell palette provides an alternate, and often more convenient, method of changing a cell’s name. See “Using the Cell Palette” later in this chapter for more information.

As explained in Chapter 6 of your *Informed Designer Design and Graphics* manual, each field and column cell has a title as part of the structure of fields and tables. When you draw a new field or table, titles are initially set to “Title.” When you change a cell’s name, Informed Designer will automatically change the field or column title to match the cell title, if you have never explicitly changed the title yourself. If you rename a cell that’s used in a formula, Informed Designer will automatically change the cell’s name in the formula too.

## Type Options

You can set type options for each cell on your form. By setting a cell’s type options, you determine factors such as the color of the data in a cell, whether or not the person filling out the form can make changes to the appearance of the data in a cell, and whether or not the type size of the data will shrink to fit the cell if the user enters more information than will fit in the cell area.

### Font, Size, and Type Style

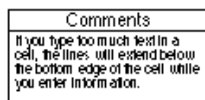
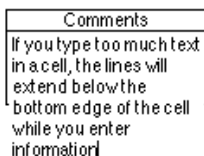
When the Informed Filler user fills out a form, the information they type into each cell is displayed using the cell’s type attributes that you’ve chosen (see “The Cell Section” and “The Column Sections” in Chapter 5 of your *Informed Designer Design and Graphics* manual). These attributes include the font, font size, type style, alignment, and leading.

For each cell, you can control whether or not the person filling out the form can change the font, font size, and type style of the information being entered. For example, you might want to allow style changes so that words or letters can be underlined. To allow such options, click any of the ‘Allow font change,’ ‘Allow size change,’ or ‘Allow style change’ checkboxes on the Cell Settings dialog box. These options apply to all cell types except Signature and Picture.

### Auto-Shrink

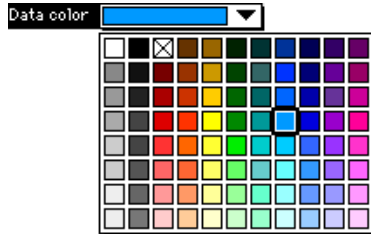
The ‘Allow auto-shrink’ checkbox controls another type option. When you enter information into a cell, you can enter more information than will actually fit in the cell area. Normally, when you press Tab to move to the next cell, the information that doesn’t fit is hidden.

If you check the ‘Allow auto-shrink’ option, Informed will shrink the type size of the information so that it fits entirely in the cell area when you print your form.



## Data Color

You can choose a color with which to display cell values. The data color setting applies to all cell types with the exception of Picture. Pictures always appear in their original color. To select a color, click the 'Data color' drop-down list on the Cell Settings dialog box, and hold down the mouse button.



Position the pointer over the desired color on the color palette and release the mouse button.

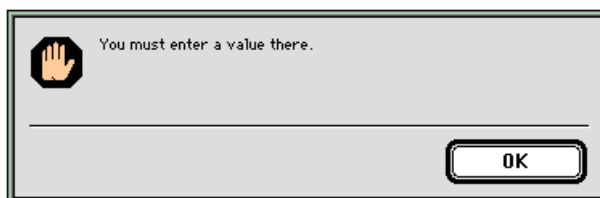
## Entry Options

Informed Designer's data entry options allow you to determine how information is entered when the Informed Filler users fills out a form. You can specify cell attributes such as whether or not entering a cell value is optional, recommended, or required, and whether or not a cell can share its memorized values with similar cells in other forms. You can also set a cell to hold only calculated or default values, instead of values entered by the user.

## Entry Status

Check formulas, as explained in "Data Verification," allow you to specify sophisticated formulas that can test for a variety of errors as the Informed Filler user fills out a form. For simple, more common error conditions, the entry status feature allows you to check for blank values where values are either required or recommended.

If a particular cell value is mandatory, select 'Required' from the 'Entry is' drop-down list. If a cell value is not mandatory but you'd like to suggest to the user that one be entered, select the 'Recommended' option instead. If the Informed Filler user neglects to enter a value in the cell, a message will appear in a dialog box indicating that a value is required or recommended.



The message is displayed when the user tabs to a different cell or when the record is accepted. Once the message appears for the first time (for a particular cell), it will not appear again unless the user later changes the value again or, for required values, when the record is accepted. Informed Filler will not allow the user to accept a record if a required cell value has not been entered.

As a shortcut, you can select the ‘Required’ entry status by clicking the corresponding button on the Cell palette. For more information, see “Using the Cell Palette” later in this chapter.

## Display Only

Informed Designer allows you to decide whether or not a cell’s value can be changed by the Informed Filler user. You might want some cells to hold only their calculated or default values, while you might allow other cells to hold values entered by the user.

If you check the ‘Display only’ feature for a cell, Informed Filler won’t let the user change that cell’s value when forms are filled out. The cell will be excluded from the tab order and if the user tries to type in the cell, a beep will sound.

You can also change the ‘Display only’ setting for a cell using the Value command and the Cell Palette. Visually, Informed Designer shows you which cells have the ‘Display only’ option selected by displaying a cell’s frame in red.

## Shared Memorization

“Default Values” later in this chapter, explains how you can specify a default value for any cell. This feature is useful if a cell’s default value is known in advance and is the same for all users.

Many forms contain information that is specific to the person filling out the form. For any one person, this information is usually the same for each form they fill out. Informed Filler provides a feature that allows the person filling out a form to specify a cell’s default value. Chapter 3, of the *Informed Filler User’s Manual* explains how the user can choose the Memorize command to set the default value for a cell.

Often different types of forms will contain some common information. For example, both purchase requisition and travel expense forms contain cells for employee information. For a particular employee, this information is the same on every form, both for purchase requisitions and travel expense forms.

The ‘Memorization is shared’ option allows you to specify that a cell is to share its memorized value with similar cells on other forms. This means that the Informed Filler user can memorize a cell’s value once and have the memorized value automatically take effect for the same cell on other forms. A cell’s memorized value is shared only if the ‘Memorization is shared’ option is selected. The memorized value is shared only with cells on other forms that have the same cell name and the ‘Memorization is shared’ option selected.



## Tab Order

Each time you create a new cell, Informed Designer assigns the next available *tab position* to that cell. On your form, the tab position of all the cells together determines the form's *tab order*; that is, the order that you tab from one cell to the next when you fill out or edit a form. The cell with tab position 1 is entered first, then the cell with tab position 2, and so on.

ABC Company 12233-44 Ave. New York, NY 98765  <b>INVOICE</b>	<b>Sold To</b> ①		<b>Ship To</b> ②	
	Date ③	Terms ④	PO Number ⑤	Ship Via ⑥
	City ⑦	No.	Description	Price
	Signature _____			Shipping Total ⑧ ⑨

The circled numbers indicate the order in which tabbing will occur.

You can change a form's tab order with the Cell command, the Change Tab Order command, or by using the Tab tool.

Use the Cell command to reposition a cell in the tab order. To do this, select the cell and choose **Cell...** from the Settings menu. When the Cell dialog box appears, type the new tab position in the 'Tab position' text box.

<b>Cell Settings</b>	
Tab Position	2

Change a cell's tab position by typing here.

After you enter the new tab position, click 'OK' to change the selected cell. If you enter an invalid tab position, Informed Designer will alert you with a message.

Changing the tab position of one cell automatically changes the tab position of other cells as well. It's like removing the cell from the tab order list, then re-inserting it back in a new position.

## Tabbing in Tables

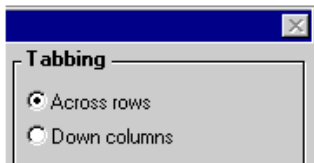
The tab order for tables is handled differently than for fields. Although a table is made up of one or more column cells, all the column cells share a common tab position. For example, if you draw a table with three columns as the first object on your form, all three columns will have tab position 1.

Although the columns in a table all share the same tab position, you can control the direction of the tabbing. The tab order within a table can be either across the rows or down the columns. Being able to tab in different directions makes it easier to fill out certain types of forms. For example, a table on an invoice form usually has columns such as ‘Quantity,’ ‘Part Number,’ and ‘Price.’ The logical order for filling out the form would be to tab *across* the rows and enter the quantity, part number, and price for each item being sold.

Tab across the rows →

Qty	Part No.	Price
10	20-0000	99.00
5	10-0000	195.00

To set a table’s tab direction to go across the rows, choose **Table...** from the Settings menu, then click the ‘Across rows’ radio button on the Table Settings dialog box.



Other types of forms are filled out easier by tabbing *down* each column rather than across the rows. For example, travel expense forms sometimes use columns for each day of the week, with the name of each expense listed beside the rows on the table. Instead of tabbing across the rows and filling out the ‘Breakfast’ expense for each day, it’s more convenient for the user to tab down one entire column and fill out all the expenses for that day, and then go on to the next column.

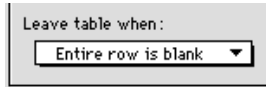
Tab down the columns ↓

	Mon	Tue	Wed
Breakfast	5.95		
Lunch	10.00		
Dinner	15.95		
Hotel	100.00		
Cab	10.00		
Entertainment			

To set a table’s tab direction to go down the columns, click the ‘Down columns’ radio button on the Table Settings dialog box.

If the tab direction for a table is set to go across the rows, the Informed Filler user can override this by pressing the Enter (Windows) or Return (Mac OS) key instead of the Tab key. Pressing Enter/Return moves down the columns rather than across the rows.

Tables on forms usually have multiple rows. However, it's not always the case that every row in a table is used when a form is filled out, so Informed allows the user to leave a table without having to tab through excess empty rows. You use the 'Leave table when' option on the Table Settings dialog box to specify when tabbing will leave a table.



By default, the 'Leave table when' option is set to 'Entire row is blank.' This means that tabbing leaves a table after tabbing through one empty row. To change this option, click the 'Leave table when' drop-down list and select the 'First column is blank' item. This means that tabbing leaves a table after tabbing out of the first column of an empty row. The 'Leave table when' option is only available when the tabbing direction is set to 'Across rows.'

## The Tab Tool



Informed Designer's tool palette contains the Tab tool. This tool allows you to change the tab order of the cells on your form by simply clicking and dragging the pointer from one cell to another.

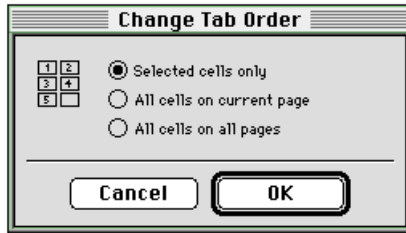
Suppose you have three cells on your form: Name (tab position 1), Fax (tab position 2), and Phone (tab position 3). You might want to change the Phone cell so that it is in tab position 2.

With the Tab tool selected, position the pointer over the Name cell (tab position 1). The pointer changes into a hand. Click the cell and hold the mouse button down while dragging the hand to the Phone cell. A gray line follows the hand to indicate which cell you are moving from. When a high-lighted border appears inside the Phone cell, release the mouse button. The Phone cell is now in tab position 2 and the Fax cell has changed to tab position 3.

## The Change Tab Order Command

The tab order of a form is commonly based on the top-left through bottom-right positioning of each cell; that is, cells are generally filled out from left to right, starting at the top of a form and working downwards. Informed Designer provides a command that lets you easily reorder the tab position of cells in this manner.

The Change Tab Order command changes the tab position of cells to match their relative position to each other. You can reorder all cells or only selected ones on a single page or on all pages of your form. To use this command, choose **Change Tab Order...** from the Arrange menu or simply double-click the Tab tool. The Change Tab Order dialog box will appear.

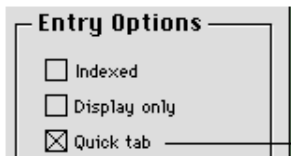


To reorder only the selected cells, choose the ‘Selected cells only’ option. The other two options allow you to reorder all cells on the current page, or all cells on all pages. After choosing an option, click ‘OK’ to perform the command. To cancel the Change Tab Order command, click ‘Cancel’ instead.

## Quick-Tabs

When the Informed Filler user fills out a form, pressing the Tab key moves from one cell to the next. Pressing the Shift-Tab key moves them in the opposite direction. Often the user might want to move directly to a particular cell on a form without having to pass through each individual cell to get there. The Quick-Tab feature allows the user to bypass the normal tab order of a form.

By pressing the F2 key (Windows) or Command-Tab keys (Mac OS), the user moves directly to the next Quick-Tab cell. Pressing Shift-F2 (Windows) or Command-Shift-Tab (Mac OS) moves them to the previous Quick-Tab cell instead. To make a Quick-Tab cell, select the cell and choose **Cell...** from the Settings menu.



Select this option on the Cell Settings dialog box to make the selected cell a Quick Tab.

Click the ‘Quick tab’ checkbox, then click ‘OK’ to dismiss the dialog box.

Use the Quick-Tab feature when your form is divided into sections. By making the first cell in each section a Quick-Tab cell, the user can easily move from one section to the next by pressing F2 (Windows) or Command-Tab (Mac OS) when the form is filled out.

In all tables, the first column is automatically a Quick-Tab cell, regardless of whether or not you select the Quick-Tab option. If the Quick-Tab option is selected for a column cell, F2/Command-Tab will tab from row to row. The first field cell following a table is also automatically a Quick-Tab cell.

The Cell palette provides an alternative, and often more convenient method of setting the Quick-Tab attribute for a cell. For more information, see “Using the Cell Palette” later in this chapter.

## Master Page Cells

When you fill out a multi-page form, Informed will automatically change pages for you when you tab between cells on different pages. Like all cells on your form, the cells that appear on the master page also have a tab position. However, since the items on the master page automatically appear on all pages of your form, there’s no need to change pages when you tab to a master page cell. Instead you’ll remain on the current page.

For more information about the master page, see “The Master Page” section in Chapter 4 of the *Informed Designer Design and Graphics* manual.

## Conditional Tabbing

By configuring a cell to have conditional tabbing, you allow the Informed Filler user to tab past sections of the form that are not relevant to the information that they are entering. For example, you could specify a tabbing condition for ‘Local’ and ‘Out of Town’ checkboxes on a travel expense form. If the user selects the ‘Local’ checkbox and then tabs from that cell, they would tab past sections for claiming expenses such as ‘Hotel Accommodation’ and ‘Airline Travel,’ and go directly to sections for claiming expenses such as ‘Parking’ and ‘Fuel.’

You specify various tabbing conditions by writing a tab formula. A tab formula applies to a particular cell and determines where tabbing should move when the Informed Filler user tabs from that cell. The result of the formula can be either the name or the tab position of the cell to which tabbing should move.

A tab formula can make use of Informed’s powerful formulas and functions capabilities. For a detailed description of these features, please see Chapters 9 and 10 of this manual.

The example formula below tests whether or not the checkbox cell “Married” is checked and, if so, returns “Spouse Name,” the name of the cell in which the user enters his or her spouse’s name. If Married is unchecked (that is, false), the formula returns “Employer,” the name of the first cell following the section for married applicants.

```
If Married then
  "Spouse Name"
Else
  "Employer"
End
```

The purpose of the above tab formula is to automatically tab past a section on the form that applies only to married applicants when the ‘Married’ checkbox cell is not checked. If a tab formula does

not return a result, then tabbing will move to the next cell in the tab order. If Spouse Name were the cell following Married (in tab order), then the following formula would work just as well.

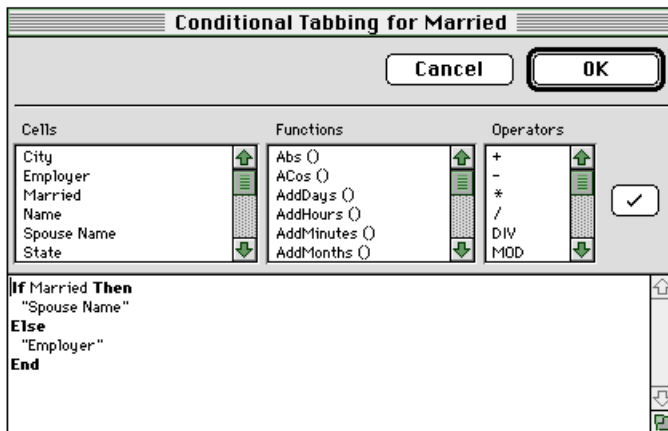
```
If Not Married then
    "Employer"
End
```

### Note

In a conditional tabbing formula, the name of the cell to move to must be in quotes. Otherwise, the value of that cell will be used as the name.

The above examples demonstrate how a tab formula can return a cell's name to identify the name of the cell to move to. A tab condition formula can also return the tab position of the cell instead.

To specify the tab formula for a particular cell, select the cell, then choose **Conditional Tabbing...** from the Settings menu. The Conditional Tabbing dialog box appears.



You enter the formula by typing in the large text box. You can resize the dialog box to show more or less of the tab conditions formula. Informed Designer makes it easy to enter complex, error-free formulas. Instead of typing cell names, functions, and operators, you can double-click any entry in any of the corresponding scrolling lists. The entry is inserted into the formula at the current insertion point. You can move between the lists on the dialog box by pressing Tab. When you tab into a list, a bold frame appears around it to indicate that it's selected.



If you double-click to enter a function that has one or more parameters, Informed Designer will automatically position the insertion point at the first parameter. If you double-click a function while holding down the Alt (Windows) or Option (Mac OS) key, the parameter names are included within parentheses.

If you click the checkmark button while entering a formula, or if you click 'OK' to dismiss the dialog box, Informed Designer will check to make sure that the formula is valid. The formula is formatted properly, and if any errors are detected, a message appears describing the nature of the error.

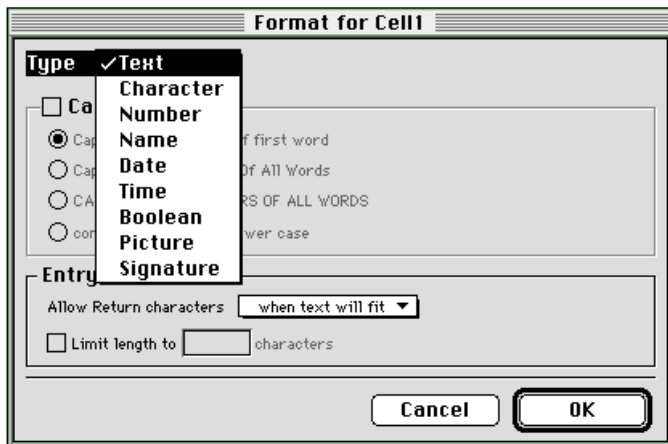
## Cell Types

This section describes Informed's cell types and the options associated with each one. Altogether there are nine different cell types. When you create a cell, you should set its type so that it matches the information that the cell is intended to hold. For example, if a cell holds a number, then its type should be number. That way, you can use the cell in arithmetic calculations, and you could have Informed Filler validate numbers that are entered into the cell when the user fills out the form.

The following table lists the nine cell types with examples.

Cell Types		
Cell Type	Examples	
Text	Business form #29	12345 - 123 Street, A Big City
Character	(555) 555-1212	02983-1283
Number	101	\$12,550.75
Name	Mr. John Smith	Jones, Mr. Tom F.
Date	10/25/89	Wednesday, November 8, 1989
Time	14:20	03:15:04 PM
Boolean	Yes	
Picture		
Signature	 Mary Ann Hancock	

Use the Format command to change the type and formatting options of a cell. To set a cell's type, select the cell, then choose **Format...** from the Settings menu. The Format dialog box appears, allowing you to choose a type from the 'Type' drop-down list.



When you select a different type, the Format dialog box changes to show the available options for that type. After you select the cell type and any options, click 'OK' to change the selected cell. To cancel the Format command, click 'Cancel' instead.

You can change the type and format of two or more cells at the same time. Simply select each cell before choosing the Format command. For more information, see the "Changing Multiple Objects" section, in Chapter 7 of your *Informed Designer Design and Graphics* manual.

Using the Format command, you can change the default cell type and format for field and table cells (the default settings determine the type and formatting options of new cells). To change the default settings, deselect all objects and choose the Format command. Any changes that you make on the Format dialog box will be set for any cells that you draw thereafter. If the Pointer tool is selected, the default settings are changed for both the Field and Table tools. If only one of the Field or Table tools is selected, the defaults are changed for that tool only. For more information, see the "Changing Default Settings" section, in Chapter 7 of your *Informed Designer Design and Graphics* manual.

As a shortcut, you can select the Format command as well as various cell types and formatting options by clicking different buttons on the Cell palette. For more information, see "Using the Cell Palette" later in this chapter.

## Text

Use the Text cell type for cells that hold textual information such as an address, a comment, or a memo. The Text cell type allows the Informed Filler user to enter any letter, number, or symbol from the keyboard into a cell.



The image shows a dialog box titled "Format for Cell1". At the top, there is a "Type" dropdown menu currently set to "Text". Below this, there are two main sections: "Case Options" and "Entry Options".

The "Case Options" section is currently unchecked. It contains four radio button options:
 

- Capitalize first letter of first word
- Capitalize First Letter Of All Words
- CAPITALIZE ALL LETTERS OF ALL WORDS
- convert all letters to lower case

The "Entry Options" section contains:
 

- An "Allow Return characters" dropdown menu set to "when text will fit".
- A checkbox for "Limit length to" which is unchecked, followed by a text input field and the word "characters".

At the bottom of the dialog box are two buttons: "OK" and "Cancel".

**Note**

Although numbers can be entered in a text cell, these values are still treated as text. If you intend to store numbers in a cell and use them in arithmetic formulas, use the Number cell type instead.

**Case Options**

A variety of options control the case of letters in a text cell. Use the case options to convert words or letters to upper or lower case. To use a case option, click the 'Case options' checkbox then choose one of the four possible options by clicking the appropriate radio button.

The first three options convert the first letter only, the first letter of all words, or all letters of all words to upper case. The last option converts all letters of all words to lower case.

**Entry Options**

Entry options for the Text cell type allow you to control whether or not the Informed Filler user can press the Enter (Windows) or Return (Mac OS) key to start a new line. You can also limit the length of a text cell to a specific number of characters.

For new cells, the 'Allow Return characters' option is set to 'when text will fit.' With this setting, the Informed Filler user can press the Enter (Windows) or Return (Mac OS) key to start a new line only if there is enough room in the cell for another line. To allow the use of the Enter/Return key for new lines regardless of the size and content of the cell, select the 'always' option instead. To prevent this use of the Enter/Return key, select the 'never' option. When the Informed Filler user presses the Enter/Return key and the use of this key for adding new lines is not permitted (because of the 'Allow Return characters' setting), Informed Filler will act as though the user pressed the Tab key instead and tab to the next cell in tab order.

To limit the length of a text cell to a specific number of characters, select the ‘Limit length to’ checkbox and enter a number in the text box provided. If the Informed Filler user attempts to enter more characters than specified by this limit, a beep will sound. If a second attempt is made, a message is displayed in a dialog box indicating the length limit.

## Character

The Character cell type also stores textual values. However, unlike the Text cell type, character values must match a specific format that you define. Use the Character cell type to store telephone numbers, zip codes, or any values that are always formatted exactly the same way.

The screenshot shows a dialog box titled "Format for Phone Number". It has a "Type" dropdown menu set to "Character". Below this, there are two text input fields: "Character Format" containing "(###) ###-####" and "Default Format" containing "(000) 000-0000". There are two radio buttons for "Match from": "Left" and "Right", with "Right" selected. Below these is a "Test Value" text box. To the right of the "Character Format" field is a list box titled "Common Formats" containing several format strings, with "(###) ###-####" selected. Below the list box is a "Formatted Value" text box. At the bottom of the dialog are "Cancel" and "OK" buttons.

### Note

The character format is used only to format values that the Informed Filler user enters or changes. If Informed Filler displays a character value that was formatted differently than the current character format for the cell, the value will display in its original format. For example, you could make a change to a “Phone number” cell on a form so that its character format is set to include the area code instead of the phone number only. If the new form is used to view data entered using the original form, the “Phone number” cell will still display the phone number without the area code, until the user explicitly changes the value in that cell.

## Character Format

A character format is a sequence of characters that rigidly defines the length and format of a valid cell value. Each character can be either a data character or a literal. Each *data character* represents a character position where a value must be supplied. A *literal* is a format character such as a dash or a parenthesis. When entering data into a character cell, the Informed Filler user doesn’t have to type literal characters; Informed Filler inserts them automatically.

In the example telephone number format shown in the previous figure, the character format consists of ten data characters (the digits), and four literals (the space, dash, and parentheses). When the

Informed Filler users enter a telephone number, they need only supply the digits. The space, dash, and parentheses are inserted automatically. Informed Filler will also ensure that what is typed correctly matches the format of a telephone number.

Each data character in a character format defines the valid set of characters for that position in a cell value. For example, since the second character in the telephone number character format is a number sign (#), then only the digits '0' through '9' are allowed in that character position. There are four predefined data characters symbols. They're shown in the table below.

#### Data Characters

Data Character	Allowable Characters (character set)
A	ABCDEFGHIJKLMNOPQRSTUVWXYZ
a	abcdefghijklmnopqrstuvwxyz
#	0123456789
?	Any character

If the predefined data characters aren't suitable, then you can define your own data character by enclosing the set of allowable characters within the '<' and '>' delimiters. For example, suppose that only the characters '1' through '9' were allowed as the first digit in a telephone area code. You couldn't use the number sign (#) data character because it allows the digit '0' as well. The character format below uses a custom defined data character.

```
(<123456789>##) ###-####
```

You can also group and repeat data characters or literals. If a number between 2 and 99 follows a data character or literal, that character (or literal) is repeated that many times in the format. For example, if '#3' appears in a format, this means that the number sign (#) is repeated three times.

```
(#3) #3-#4
```

To group two or more characters in a character format, enclose the characters within the '{' and '}' delimiters. You can repeat a group of characters the same way you repeat a data character or literal—by following the group with a number between 2 and 99. In the example below, the last four digits of the telephone number character format are represented by a group of two digits repeated twice.

```
(###) ###-{##}2
```

The characters 'A,' 'a,' '#,' '?,' '<,' '>,' '{,' '}',' and 'O' through '9' are called special characters. They're special because, as described above, they each have a predefined meaning when used in a character format. If you want to include a special character as a literal in a character format, you must precede it with the *escape character*. Informed Designer's escape character is the backslash symbol (\).

Suppose that you want to use brackets ('{...}') instead of parentheses to surround the area code of a telephone number. You would use the escape character in the character format (because brackets are special characters).

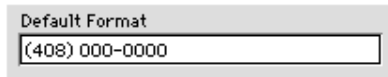
```
\{###\} ###-####
```

Since the escape character precedes each of the bracket symbols, the brackets are interpreted as literals and not as group delimiters.

Informed Designer provides a list of common character formats. You can choose a common format by double-clicking an entry in the scrolling list of choices. Or you can create your own format by typing directly in the 'Character format' text box.

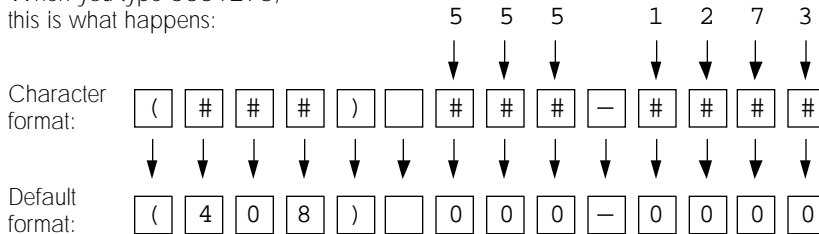
### Default Format

Each character cell can have a default format. The default format doesn't represent the default value of the cell (see *Calculations and defaults*). Instead, it's used when someone filling out your form types an incomplete value. If the user enters an incomplete value, Informed Filler will fill in the missing characters with those found in the default format.



In the example telephone number format, the default format is used to supply a default area code. If the Informed Filler user types the value '5551273,' Informed Filler will fill in the area code and display '(408) 555-1273.' The digits typed are matched with the characters in the character format, starting at the right end and working to the left. Since only seven of the ten required data characters are entered, the remaining three are obtained from the default format.

When you type 5551273,  
this is what happens:



The result: (408) 555-1273

The 'Match from' option controls which direction the characters you type are matched with the characters in the character format.



It's important to choose the proper direction when you supply a default character format. Otherwise, Informed Filler will use the wrong default characters to complete a typed value. For example, the value '5551273' would be displayed as '(555) 127-3000' if you changed the matching direction from right to left.

You enter a default format by typing the value in the 'Default format' text box. The value you type must match the character format. If you type an invalid default format, Informed Designer will alert you with a message. To choose the match direction, click the appropriate radio button.

## Testing Your Character Format

To confirm that you've entered the correct format values and options, Informed Designer allows you to test your format before dismissing the Format dialog box. After you enter the character and default formats, and choose the match direction, you can enter a sample value in the 'Test value' text box. The converted value is shown under the Formatted Value heading.

The following table gives a few examples. If the value under the 'Entry' column doesn't match the format, you'll see the word 'Error' (and a message) under the 'Informed Filler Displays' column.

Example Formats

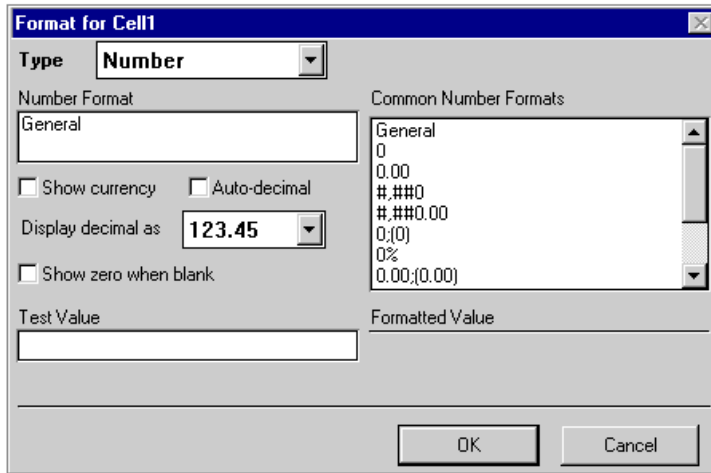
Character Format	Default Format	Match	Entry	Informed Displays
(###) ###-####		Left	4154561234	(415) 456-1234
(###) ###-####		Right	4561234	Error - not enough characters
(###) ###-####	(415) 000-0000	Right	1234	(415) 000-1234
A# #A#		Left	t5t4r4	T5T 4R4
####	0000	Right	12	0012
####	0000	Right	12a	Error - 'a' doesn't match
###?	0000	Right	12a	012a
###?	0000	Left	12a	Error - 'a' doesn't match
{#3-}2#3*		Left	123456789	123-456-789
AA-#3	XL-000	Right	534	XL-534
AA-#3	XL-000	Right	XM534	XM-534
AA-#3	XL-000	Right	X8534	Error - '8' doesn't match
The \answer is ##.		Right	23	The answer is 23.

\* Is equivalent to '###-###-###.'

If you enter an invalid character format, default format, or test value, the formatted value will be blank. When you click 'OK' to dismiss the Format dialog box, Informed Designer will check to make sure that the character format and default format are valid. If an error is detected, you'll be warned with an alert.

## Number

Use the Number cell type to store number values. You can display numbers with up to 18 digits of precision. The largest number that you can store accurately is 999,999,999,999,999. The smallest is -999,999,999,999,999.



### Number Format

Like the Character cell type, you describe the format of a number cell using a sequence of symbols. You can choose from a list of common number formats, or you can create your own.

When you create a number format, you type special symbols in a form that represents how you want the number to look. Each symbol has a defined meaning. For example, the symbol zero ('0') is used as a digit placeholder. The number format '000' formats any number value to produce an equivalent number that's at least three digits long.

#### Common Number Formats

Number Format	Entry	Informed Displays
General	123	123
	123.309	123.309
	-1523.001	-1523.001
000	0	000
	12	012
	1234	1234
###.##	1.2	1.2
	53.215	53.22
###.00	1.2	1.20
	1500.379	1500.38
#,###.00	1500.379	1,500.38
	5	5.00

The ‘General’ number format is a special format that makes use of no formatting options. Numbers entered using this format are displayed as typed with a maximum of nine decimal places of accuracy. Other formats consist of special characters and symbols, each of which has its own meaning, as described below.

#### Symbol Meaning

Symbol	Meaning
0	A digit placeholder. If a number has fewer digits than zeros in the format, Informed Filler inserts extra zeros. If the number has more digits to the right of the decimal place than zeros in the format, Informed Filler rounds the number’s fractional part to the number of decimal places in the format.
#	A digit placeholder. This symbol follows the same rules as zero above, except that extra zeros aren’t displayed if the number has fewer digits on either side of the decimal than there are number signs (#) in the format.
*	Like the zero symbol, the star (*) is a digit placeholder. However, if a digit is not supplied for the corresponding character position in the number, Informed displays a star (*) instead of a zero.
.	The decimal point. The position of this symbol in the number format determines how many digits are displayed to the left and right of the decimal point.
,	Thousands separator. If this symbol appears on either side of the decimal point, Informed inserts thousands separators on that side in the formatted number.

In addition to these symbols, you can also include literal characters before and after the digit symbols in a number format. A literal character appears unchanged in a formatted number. For example, the number format ‘Acc. No. 0000’ would always format to the characters ‘Acc. No.’ followed by a four digit number.

Informed Designer allows you to enter a different number format for the positive, negative, and zero forms of a number. This feature allows you to use custom negation indicators such as parentheses or the letters ‘Dr.’ Simply separate the individual formats—in the order positive, negative, then zero—with the semi-colon symbol (;). The table below shows a variety of number formats with examples.

## Number Formats

Number Format	Entry	Informed Displays
#,##0.00Cr;#,##0.00Dr;zero	123.45	123.45Cr
	-6251.32	6,251.32Dr
	0	zero
#,##0.00;(#,##0.00);0	-12345	(12,345.00)
	0.00	0
Acc. No. 0000	15	Acc. No. 0015
Balance due is #,##0.00;Credit is #,##0.00;Nil balance	15.75	Balance due is 15.75
	-1500	Credit is 1,500.00
	0	Nil balance

To enter a number format, type directly in the 'Number format' text box, or double-click an entry in the scrolling list of common number formats.

## Currency

If you want Informed to automatically add a currency indicator to a formatted number, check the 'Show currency' checkbox. The currency symbol appears immediately to the left or right of the digits in the number depending on the standard used in your country. Informed knows which symbol to use.

## Auto-decimal

For floating point numbers (numbers with at least one decimal place of accuracy), you can enter the decimal point yourself, or Informed can insert it for you. This option is often found on electronic calculators. It's commonly used for entering currency values.

## Using Auto-decimal

Number Format	Auto-decimal	Entry	Informed Displays
#,##0.00	No	12345	12,345.00
	Yes	12345	123.45
	Yes	10000	100.00
	Yes	1234.5	1,234.50

To use the auto-decimal option, click the 'Auto-decimal' checkbox. The person who fills out your form can always override this feature by entering a decimal point when a number is typed.

## Decimal style

You can display the decimal point of a number as the decimal point symbol, or as a vertical bar that extends from the top edge to the bottom edge of the cell. Although the decimal point is the most common choice, the vertical decimal bar is often used on table columns to line up numbers on adjacent rows.



123.00	123	00
45.35	45	35
32.99	32	99
25.95	25	95
15.89	15	89

To choose a decimal style, select your choice from the 'Display decimal as' drop-down list.

When you choose the vertical decimal bar option, Informed Designer automatically positions the line according to the current type attributes of the cell, and the number of decimal places in the cell's number format. If you change any of these settings, Informed Designer will reposition the line as necessary.

#### Note

When you use the vertical bar option, number values are always right aligned, even if you select center or left alignment.

You can select a cell's vertical decimal bar and change its appearance. Like line objects, a vertical decimal bar has pen shade, line width, and line style attributes. To select a vertical decimal bar, click the line with the Pointer tool. The line will shimmer when it's selected. To change an attribute, choose a new setting from a Style submenu, or use the Paint command in the Style menu. For more information about paint attributes and how to change them, see the "Paint Settings" section, in your *Informed Designer Design and Graphics* manual.

## Displaying Zero Values

When a form is filled out with Informed Filler, values appear only when the user types in the blanks, or when cells are calculated or filled in with default values. This is analogous to filling out a form by hand or with a typewriter.

Sometimes you want to display the number zero in a cell even if a value hasn't been entered or calculated. Maybe you want to show a zero amount in the extension cell even if the quantity and price values are blank. If you check the 'Show zero when blank' checkbox, Informed Filler will display the zero value when the cell is empty.

## Testing Your Number Format

You can test your number format before you dismiss the Format dialog box. After you enter the number format and choose your options, enter a sample number in the 'Test value' text box. The formatted number is shown under the 'Formatted value' heading.

If you enter an invalid number format or test value, Informed Designer will clear the formatted value. If you click 'OK' to dismiss the Format dialog box and an error is detected in the number format, you'll be alerted with a message.

## Name

If a cell will contain a person's name, use the Name cell type. The Name cell type displays names using a format that you define.

A name has up to five parts: a prefix (such as Mister or Professor), a first name, a middle name, a last name, and a suffix (such as Junior). Multiple prefixes, middle names, and suffixes are allowed. You can display a part in long or abbreviated form, or you can hide a part altogether. You can also display the surname before all other parts, or in its usual position between the middle name and suffix.

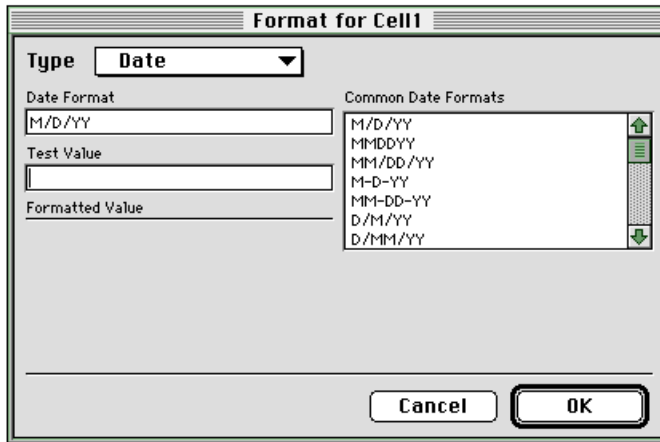
The screenshot shows the 'Format for Cell1' dialog box. The 'Type' dropdown is set to 'Name'. Under 'Name Parts', there are five checkboxes: Prefix (unchecked), First (checked), Middle (unchecked), Surname (checked), and Suffix (unchecked). Below each checkbox are two radio buttons for 'Long' and 'Abbr.' forms. The 'Surname' section has two radio buttons: 'Last' (selected) and 'First'. The 'Sample' field displays 'John Smith'. At the bottom are 'OK' and 'Cancel' buttons.

Check the name parts that you want to include in the name format. For those parts, select either the long or abbreviated form by clicking the appropriate radio button. Then choose the surname position by clicking either of the 'First' or 'Last' radio buttons. The sample name changes to reflect the format that you choose.

When the users fill out a name cell, Informed Filler will match the parts that they type with the parts in the name format. If the user types a name in a format that's different from the format of the cell, Informed Filler will automatically change what they type to match the correct format. In order to correctly identify the parts of a name, Informed Filler refers to a list of prefixes and suffixes. This list can be found in Appendix A.

## Date

Use the Date cell type to store date values. You can store any date between roughly 30,000 BC and 30,000 AD. Date cells can be used in formulas and in functions that manipulate dates. For example, the AddDays function adds a certain number of days to an old date to calculate a new date. For information about formulas and functions, see Chapters 9 and 10.



You can display date values in any format you like. You describe the format of a date by typing special symbols in a form that represents how you want the date to look. For example, the symbol 'MONTH' represents the month spelled in capital letters (for example 'JANUARY').

### Common Date Formats

Date Format	Entry	Informed Displays
M/D/YY	Jan 14 96	1/14/96
	15/08/96	8/15/96
	1*	1/1/96
MM/DD/YYYY	3/9/89	03/09/1989
MON-D-YY	2/17/92	FEB-17-92
	June 3 90	JUN-3-90
Month D, YYYY	3/15/88	March 15, 1988
Mon. D/YY	9/21/91	Sep. 21/91
Dy, Mon D, YYYY	111891	Sun, Nov 18, 1991
MON-D-YYYY AD	2/14/520 bc	FEB-14-520 BC
MONTH, YY	7/78	July, 78
MMDDYY	10/2/90	100290

\* Missing parts are filled in using today's date.

A date has four components: the day of week, the day of month, the month, and the year. You create a date format by combining symbols that represent these components in any order and format. The following table describes each symbol.

#### Symbol Meaning

Symbol	Meaning
D	The day of month without a leading zero (1 - 31).
DD or 0D	The day of month with a leading zero (01 - 31).
M	The month of year without a leading zero (1 - 12).
MM or 0M	The month of year with a leading zero (01 - 12).
Month	The month of year spelled with the first letter capitalized (January - December).
MONTH	The month of year spelled in capital letters (JANUARY - DECEMBER).
month	The month of year spelled in small letters (january - december).
Mon	The abbreviated month of year with the first letter capitalized (Jan - Dec).
MON	The abbreviated month of year in capital letters (JAN - DEC).
mon	The abbreviated month of year in small letters (jan - dec).
YY	The year displayed as a two-digit number.
YYYY	The year displayed as a four-digit number.
Day	The day of week spelled with the first letter capitalized (Sunday - Saturday).
DAY	The day of week spelled in capital letters (SUNDAY - SATURDAY).
day	The day of week spelled in small letters (sunday - saturday).
Dy	The abbreviated day of week with the first letter capitalized (Sun - Sat).
DY	The abbreviated day of week in capital letters (SUN - SAT).
dy	The abbreviated day of week in small letters (sun - sat).
AD or BC	The abbreviated era in capital letters.
ad or bc	The abbreviated era in small letters.

When you combine two date components, you separate them with a separator character. You can use the slash (/), comma (,), space ( ), decimal (.), or dash (-) separator characters.

When the user types a date value, Informed Filler interprets the different date components and displays them using the format of the date cell. If you don't type a component that's part of the date format, Informed Filler inserts the corresponding component of today's date. For example, if today's date is February 15th, 1996, and you type the value '1' using the date format 'M/D/YY,' Informed Filler will display '2/1/96.' The current month and year are inserted to complete the date value.

When you use a date format with a four-digit year (YYYY), Informed Filler will always fill in the current century if you enter a two digit year.

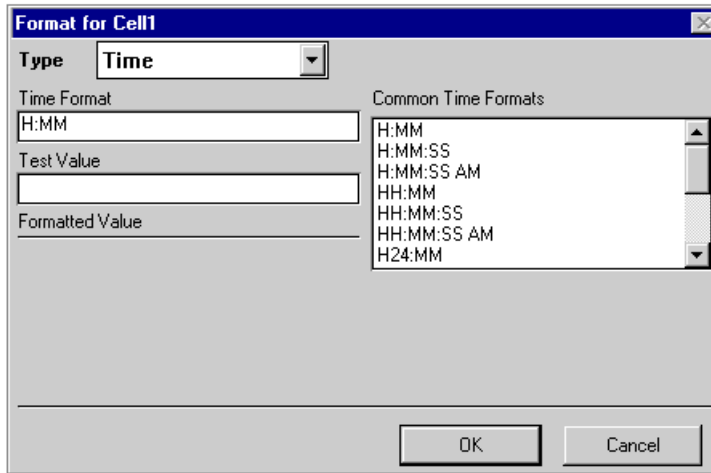
You can choose one of the common date formats by double-clicking an entry in the scrolling list, or you can create your own format by typing in the 'Date format' text box. If you enter an invalid date format, Informed Designer will alert you with a message.

## Testing Your Date Format

You can test your date format before you dismiss the Format dialog box. After you type the date format that you want, enter a sample date value in the ‘Test value’ text box. The formatted date is shown under the ‘Formatted value’ heading. If you enter an either an invalid date format or an invalid sample value, Informed Designer will clear the formatted value.

## Time

Use the Time cell type to store time values. A time value can range from 0:00:00 to 23:59:59; it represents the time of day. Time cells can be used in formulas and with functions that manipulate time values. For example, the function ADDMINUTE adds a certain number of minutes to an old time to calculate a new time. For information about formulas and functions, see Chapters 9 and 10.



You can display time values in any format you like. You describe the format of a time by typing special symbols in a form that represents how you want the time to look. For example, the symbol ‘HH’ represents the hour of day in 12 hour format. The following table shows some common time formats.

## Common Time Formats

Time Format	Entry	Informed Displays
HH:MM	5 34	05:34
	5:34:15	05:34
	5*	05:00
H:MM	5:34	5:34
HH MM SS	17:3:23	5 03 23
H24:MM	3:14	3:14
	3:14 PM	15:14
H:MM:SS PM	9 45	9:45:00 AM
	3:10 AM	3:10:00 AM
	14:50	2:50:00 PM
M:SS	09:40	9:40

\* Missing parts are filled in with zero.

A time value has four components: the hour, the minute, the second, and the AM/PM indicator. You create a time format by combining symbols that represent these components in any order and format. The table below describes each symbol.

## Symbol Meaning

Symbol	Meaning
H	The hour in 12 hour form without a leading zero (1 - 12).
HH or 0H	The hour in 12 hour form with a leading zero (01 - 12).
H24	The hour in 24 hour form without a leading zero (1 - 23).
HH24 or 0H24	The hour in 24 hour form with a leading zero (01 - 23).
M	The minute without a leading zero (1 - 59).
MM or 0M	The minute with a leading zero (01 - 59).
S	The second without a leading zero (1 - 59).
SS or 0S	The second with a leading zero (01 - 59).
AM or PM	Include the AM/PM indicator in capital letters.
am or pm	Include the AM/PM indicator in small letters.

When you combine two time components, you must separate them with a separator character. You can use the colon (:), decimal (.), or space ( ) separator characters.

When the Informed Filler user types a time value, Informed Filler interprets the different time components and displays them using the format of the time cell. If the user doesn't type a component that's part of the time format, Informed Filler inserts a zero for them. For example, if they type '8' using the time format 'HH:MM,' Informed Filler will display '08:00.'

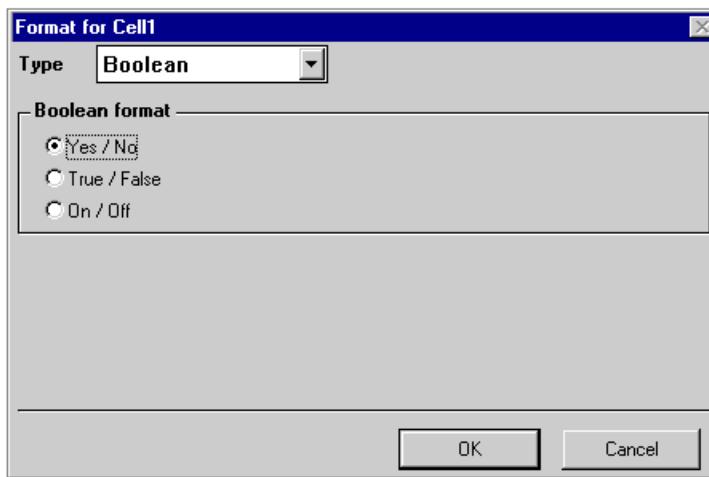
You can choose one of the common time formats by double-clicking an entry in the scrolling list, or you can create your own format by typing in the 'Time format' text box. If you enter an invalid time format, Informed Designer will alert you with a message.

## Testing Your Time Format

You can test your time format before you dismiss the Format dialog box. After you type the time format that you want, enter a sample time value in the ‘Test value’ text box. The formatted time is shown under the Formatted Value heading. If you enter either an invalid time format or an invalid sample value, Informed Designer will clear the formatted value.

## Boolean

The Boolean cell type stores values that are True or False. You can display the value of a boolean cell as ‘Yes’ or ‘No,’ ‘True’ or ‘False,’ or ‘On’ or ‘Off.’



With the Boolean cell type, Informed Filler will ensure that the person filling out the form enters only the values that are appropriate for the selected style. If the user types only part of a value, Informed Filler will convert it to its full form. For example, if they type the letter ‘n’ while using the Yes/No style, Informed Filler will convert the value and display ‘No.’

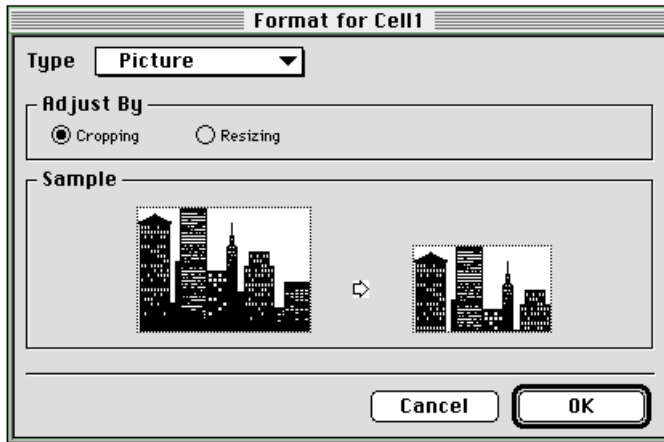
To choose a boolean format, click one of the radio buttons under the ‘Boolean format’ heading.

### Note

The Boolean cell type is appropriate for use with Informed’s checkbox feature since, like Boolean cells, a checkbox can represent one of two values. For information on checkboxes, see “Checkboxes” in Chapter 7 of your *Informed Designer Design and Graphics* manual.

## Picture

The Picture cell type allows you to reserve space on your form for pictures. A picture can be any image created with virtually any drawing program. When filling out a form, instead of typing a value in a picture cell, the user pastes an image using the Paste command from the Edit menu, or imports a picture using the Insert File command from the Cell menu.



Informed Filler supports the following picture formats in picture cells: Windows Bitmap (.BMP), Windows Metafile (.WMF), Macintosh PICT (.PCT), and Encapsulated PostScript (.EPS).



The Windows Metafile format is not supported on the Mac OS.

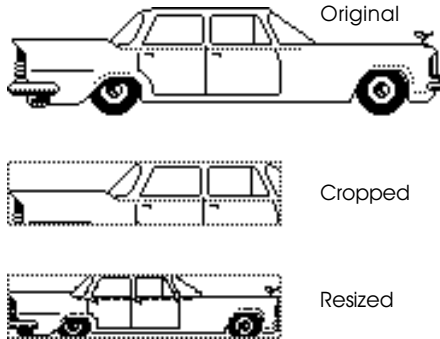
By using a picture cell, the person filling out your form can enter a different picture on each new completed form. For example, on your inventory form you could store a picture of each inventory item.

### Note

Unlike cells of other types, you can't index picture cells. See "Indexing Cells" later in this chapter for more information.

Often the size of the picture that's pasted into a picture cell is larger than the size of the cell itself. Informed Designer allows you to resize or crop the image so that it fits in the cell. When an image is cropped, the area that doesn't fit in the cell is hidden. If you choose the resize option instead, the image is reduced proportionally to a size that fits completely in the cell.





When a form is filled out with Informed Filler, pressing the Tab key moves the user from one cell to the next. When they move to a picture cell, the frame of the cell flashes to indicate that the cell is currently *active*. When the Informed Filler user chooses the Insert File command, the standard Open dialog box appears, prompting to select a file. As a shortcut to choosing the Insert File command, the user can select the picture cell and press Enter (Windows) or Return (Mac OS).

While a picture cell is active, the user can clear it by pressing the Backspace or Delete key, or by choosing the Clear command in the Edit menu.

## Signature

The Signature cell type allows you to create a cell that can be used to sign the data on forms electronically with digital signatures. Each signature cell can be configured to sign the entire form, or parts of the form. Although the signature cell is created in Informed Designer, you actually sign the form in Informed Filler. For a detailed discussion of the signature capabilities of Informed, please see Chapter 2, “Using Digital Signatures” and Chapter 7 “Authorizing Templates.”

## Indexes

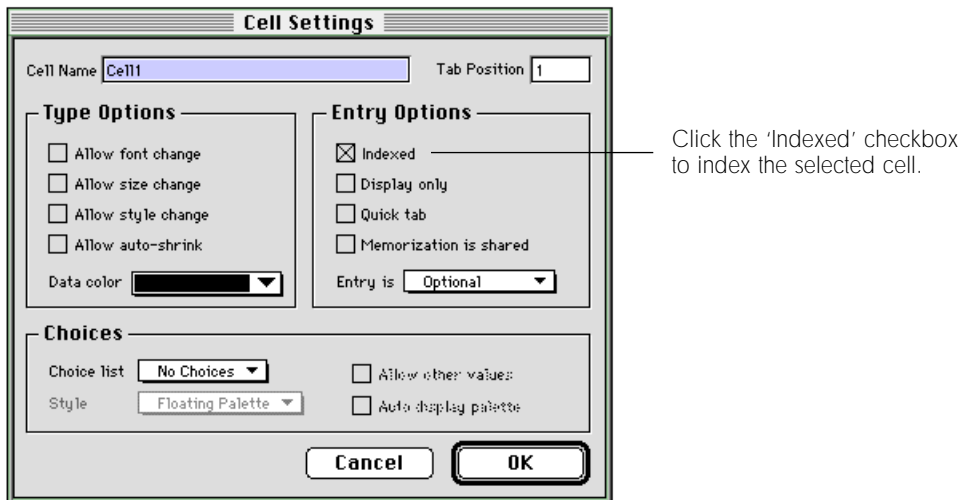
Each time a form is filled out with Informed Filler, the information that the user types is added to the data document’s database of records. A data document can contain potentially thousands of records. Informed Filler’s Find command lets you search through the database to find and display particular records.

Each cell that you create on a form (by drawing fields and tables) can be indexed with the exception of signature cells and picture cells. An *index* is a pre-sorted list of cell values that Informed Filler maintains automatically as you add, remove, and change records. Although you never actually see an index, you can certainly notice its effect when you use Informed Filler’s Find command to find forms.

If a cell is indexed, Informed Filler can search quickly through thousands of records to find a matching value. Depending on the speed of your computer and the number of records in your data document, searching can be as fast as one or two seconds. If a cell is not indexed, Informed Filler has to examine the cell value on each record individually in order to find those that match. Searching can take considerably longer if the cell is not indexed.

## Indexing a Cell

You index a cell by clicking the 'Indexed' checkbox on the Cell dialog box. Select the cell (or cells) that you want to index, then choose the **Cell...** from the Settings menu. The Cell dialog box appears.



Click the 'Indexed' checkbox, then click 'OK' to dismiss the Cell dialog box.

With the exception of picture and signature cells, you can index any cell on your form. The following table describes how values of each different cell type are indexed.

### Indexing Cells

Cell Type	Index Method
text	Each word is indexed separately
character	The entire character value is indexed
name	Each name part is indexed
number	The numeric value is indexed
date or time	The date or time value is indexed
boolean	The boolean value is indexed

It's important that you choose the correct cell type for the indexed cells on your form. A cell's type determines not only the type of information that the cell can store, but also the order that cell values are sorted. For example, suppose that a cell called 'Date' stores the ship date on an invoice form. If

the cell's type is Text, cell values would be sorted alphabetically. By using a date cell instead, the cell would be sorted chronologically rather than alphabetically.

When you create a form, care must be taken to properly select which cells are indexed. Since each index adds to the size of your document and affects how quickly Informed Filler can add, remove, and change records, you shouldn't overuse indexes. As a general rule, you should index only those cells that are intended to be used commonly for searching. They usually include cells such as a person's name, or the identification number of a form (the invoice number, for example).

## Calculations

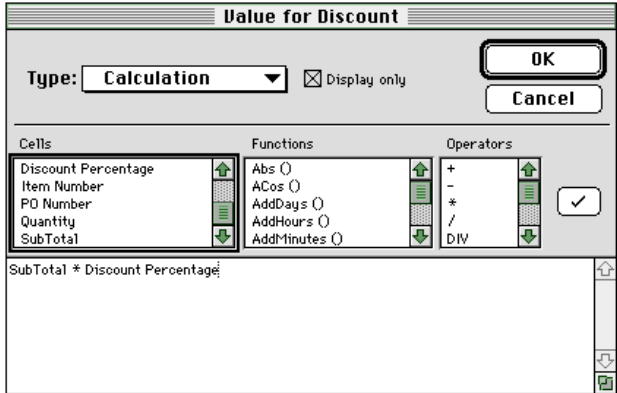
Often a cell gets its value by manipulating other information on a form. For example, the discount amount on a sales slip is calculated as the discount rate times the total purchase amount. You can use a calculation so that the value is filled in automatically for the Informed Filler user.

	<b>Sub Total</b>	<b>754.90</b>
<b>Discount Rate</b>	<input type="text" value=".07"/>	<b>Discount</b>
		<b>52.84</b>
	<b>Total</b>	<b>702.06</b>

Calculate the discount amount as:  
'Sub Total \* Discount Rate'

In addition to mathematical calculations, you can also manipulate other types of information to produce a calculated result. Informed Designer provides a comprehensive set of operators and functions that make it easy to create sophisticated calculations. You can even use if-then-else logic to calculate different results under certain conditions. For detailed information on formulas and functions, see Chapters 9 and 10 respectively.

To create a calculation, first select the cell that you want to calculate, then choose **Value...** from the Settings menu. The Value dialog appears.



You create a calculation by typing a formula in the formula text box. First select the calculation option by choosing ‘Calculation’ from the ‘Type’ drop-down list. As a shortcut, Informed Designer will automatically select the ‘Calculation’ type when you type the first character of a formula.

A formula is like a mathematical equation. You combine operators and functions with cell names and constants to produce a new result. For example, to create a calculation that multiplies the cells ‘Sub Total’ and ‘Discount Percentage,’ you would enter this formula:

```
Sub Total * Discount Percentage
```

This formula uses the multiplication operator (\*) to return the product of the two values. Other formulas could use additional operators and any of Informed’s powerful functions.

Suppose that instead of entering the discount rate, you would like to calculate its value according to the total purchase amount. If the total purchase amount is less than \$50, the discount rate should be 0.05 (5%). Otherwise, the rate should be 0.15 (15%). The formula below uses the IF statement to calculate the correct value.

```
If Sub Total < 50 Then
  0.05
Else
  0.15
End
```

The first line of the formula checks the value of the ‘Sub Total’ cell. If its value is less than 50, the result of the formula is 0.05. If its value is not less than 50 (that is, greater than or equal to 50), the result of the formula is 0.15 instead. For a complete discussion about formulas and functions, see Chapters 9 and 10.

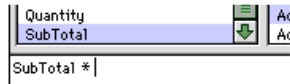
The result of a calculation formula should match the type of the cell that it sets. For example, if a cell’s calculation formula adds two numbers, the cell’s type should be number. If the resulting type of a calculation is different than the cell’s type, Informed will try to automatically convert the result to the cell’s type. For more information, see the “Type Compatibility” section in Chapter 9, “Using Formulas.”

You can use the ‘Display only’ feature to prevent someone filling out your form from changing the value of a calculated cell. This feature is also available on the Cell dialog box, and is described in detail in the “Entry Options” section of this chapter.

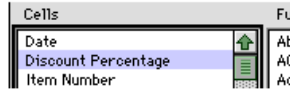
## Entering a Calculation Formula

As described earlier in this section, you create a calculation by typing a formula in the formula text box on the Value dialog box. Informed Designer makes it easy to enter complex, error-free formulas. Instead of typing cell names, functions, and operators, you can double-click any entry in any of the corresponding scrolling lists on the Value dialog box. The entry is inserted in the formula at the

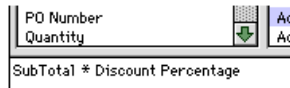
current insertion point. You can move between the scrolling lists by pressing Tab. When you tab into a list, a bold frame appears around it to show that it's selected.



Place the insertion point at the proper location...



...double click an entry in one of the scrolling lists...



...the entry appears in the formula.

If you double-click to enter a function that has one or more parameters, Informed Designer will automatically position the insertion point at the first parameter. If you double-click a function while holding down the Alt (Windows) or Option (Mac OS) key, the parameter names are included within parentheses.

You can also enter a cell's name by clicking the cell in the drawing window. This is useful if you don't know the name of the cell, but you can see it in the drawing window.

If you click the checkmark button while entering a formula, or if you click 'OK' to dismiss the dialog box, Informed Designer will check to make sure that the formula is valid. The formula is formatted properly, and if any errors are detected, a message appears describing the nature of the error.

## Default Values

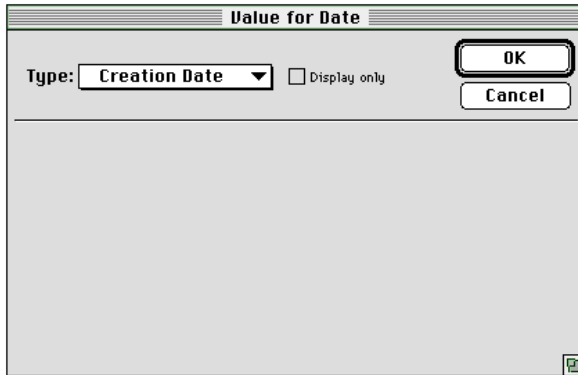
A default value is a value that Informed Filler automatically fills in each time the user fills out a new form. However, unlike calculations, a default value doesn't change unless the user types a different value. Use a default value whenever a cell often has the same value. For example, the default value for the date cell on an invoice could be today's date.

There are three different types of default values. They are:

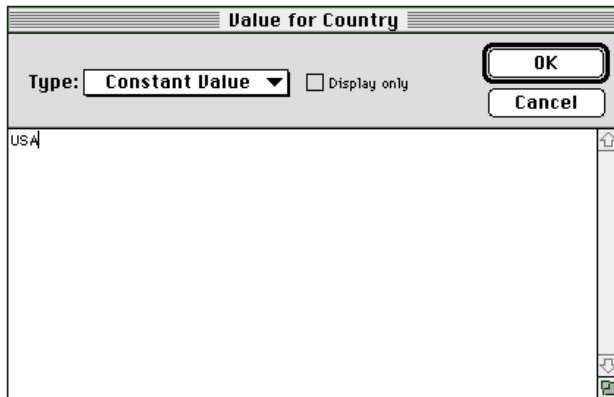
- creation date
- creation time
- constant value

'Creation date' and 'Creation time' default value types are used for automatic entry of the current date or time when the Informed Filler user fills out a new form. The 'Constant value' default type requires that you specify the default value itself.

To create a default value, first select the cell that you want to have the default value, then choose **Value...** from the Settings menu. On the Value dialog box, select 'Creation date,' 'Creation time,' or 'Constant value' from the 'Type' drop-down list.



When you select the "Constant value" type, a text box appears allowing you to enter the default value.



The default value that you enter should be appropriate for the type of cell to which it applies. For example, if the default value applies to a number cell, be sure to enter a number as the default value.

**Note**

The values of cells that are checkboxes automatically default to unchecked when a new form is filled out. To specify the "checked" default value, enter "True," "Yes," or "On."

After selecting the default type and, for constant values, entering the default value, click 'OK' on the Value dialog box. To cancel the Value command, click 'Cancel' instead.

## Auto-incrementing Numbers

Forms such as invoices, time sheets, and purchase orders are often numbered uniquely for identification purposes. Each time a new form is filled out, a new number is assigned. Informed Designer provides a number of ways to automatically generate these numbers.

Other sections of this chapter explain how you can specify calculations and default values for any cell using Informed Designer's Value command. By selecting 'Auto-increment' from the 'Type' drop-down list on the Value dialog box, you can configure a cell to be assigned a new number each time the Informed Filler user fills out a new form.

The screenshot shows a dialog box titled "Value for Invoice Number". It has a "Type" dropdown menu set to "Auto-increment" and a checked "Display only" checkbox. Below this, there are two more dropdown menus: "Get next value when" set to "New record is added" and "Assign next value from" set to "This template". There are also two text input fields: "Next value:" which is empty, and "Increment by:" which contains the number "1". At the top right of the dialog are "OK" and "Cancel" buttons.

There are several different methods with which Informed Filler can obtain new numbers. The next available number can be stored in the form template itself, or it can be obtained from another application or data source. You choose a method by selecting a choice from the 'Assign next value from' drop-down list.

The screenshot shows a dropdown menu for "Assign next value from:". The menu is open, showing several options. The first option, "This template", is selected with a checkmark. The other options are "Apple event application", "DAL", "ODBC", "Oracle", and "Sybase".

The first two options, 'This template' and 'Apple event application,' are built into Informed Designer and Informed Filler. The 'Apple event' option is available only on the Mac OS. All other options correspond to the data access plug-ins that you have installed in your Informed plug-ins folder.

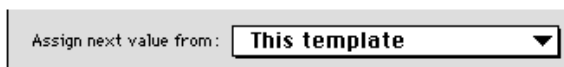
Informed Filler's Cell menu contains the Assign Next Value command. The Informed Filler user can select an auto-incrementing cell and choose this command to manually obtain a new number and enter it in the cell. If you choose the 'New record is added' option from the 'Get next value when' drop-down list on Informed Designer's Value dialog box, Informed Filler will automatically enter the next available number when the user adds a new record.

**Note**

If the user of your form is often mobile and not connected to the application or database that supplies new numbers, the 'Manually only' setting might be more appropriate. That way, the Informed Filler user can choose to assign numbers manually using the Assign Next Value command, and do so only when a connection is active.

## Storing the Number in the Form Template

With the 'Assign next value from' drop-down list set to 'This template,' Informed Designer stores the next available number in the form template document itself.



Each time Informed Filler assigns a new number to a cell, the next available number is read from the form template document and entered in the cell. The number is then incremented and stored back in the form document.

To set the next available number, enter a value in the 'Next value' text box. This value can be a number or an alpha-numeric value. By default, Informed Filler will increment the next value by 1 each time a number is assigned. You can change the increment amount by typing a different value in the 'Increment' text box.

If your form will be used by more than one person, you might want to prevent the same number from being assigned to two different forms filled out by two different users. One way to do this is to link the auto-incrementing cell to a database or application that can act as a central distribution point for new numbers. Alternatively, you might consider combining the number with other values on the form (the user's employee number, for example), to ensure uniqueness among different users.

## Linking to Apple Event Applications

An Apple event aware application is an application that can send and receive messages using the Apple event capability of the Mac OS operating system. Two Apple event aware applications can communicate with each other using this method.

Informed Filler can use Apple events to communicate with another application in order to request and obtain new values for auto-incrementing cells. However, only certain applications, including



Informed Number Server and 4th DIMENSION by ACIUS (using the Informed 4D External) can understand the specific Apple events that Informed Filler uses to communicate. Furthermore, this method of linking is available only on Mac OS compatible computers using system software version 7.0 or later. For detailed information about Informed Number Server, please see Chapter 11, “Using Informed Number Server.”

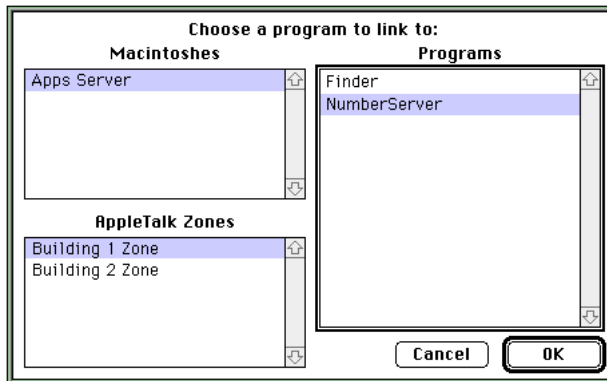
With the ‘Assign next value from’ option set to ‘Apple event application,’ you can choose an application to link the selected cell to.

Assign next value from: **Apple event application** ▼

### Note

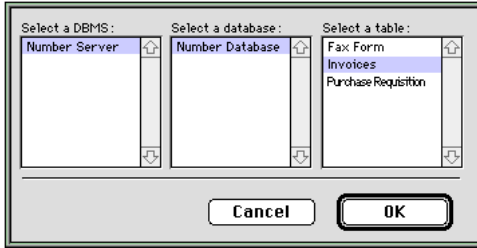
Before you can link an auto-incrementing cell to an Apple event aware application, you must run the application and open the correct database or data file.

To select an application, click the ‘Choose Application’ button. Informed Designer displays the Program Linking dialog box.



Choose the computer that’s running the application, then select the application itself and click ‘OK.’ The name of the application will appear next to the ‘Choose Application’ button.

Since a single application—such as Informed Number Server—can generate several different sequences of numbers (for example, one for invoices, one for purchase orders, and so on), you must specify which particular number to link the auto-incrementing cell to. To do this, click the ‘Choose Table’ button. (In database terminology, a file of information is often referred to as a table.)



Most applications—like Informed Number Server—act like a single DBMS and offer access to one or more databases. When you select a database, the third list shows the available tables. Each table delivers a corresponding sequence of numbers. Select the appropriate table (or number) name then click ‘OK.’ The names of the DBMS, the database, and the table appear next to the ‘Choose Table’ button.

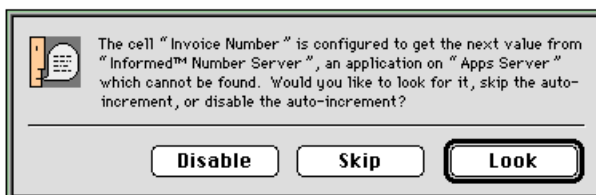
## Errors While Testing or Filling Out Forms

Although Informed Designer can check to make sure that you don’t make mistakes when you link an auto-incrementing cell to an Apple event aware application, there are several reasons why an auto-increment can fail to work properly. For example, suppose that an auto-increment is linked to a particular application, and that application is not running on the correct computer. Or maybe the application doesn’t have the correct database or data file open.

Errors can occur when you test a form with Informed Designer or when forms are filled out with Informed Filler. There are three basic types of errors.

- The application to which an auto-incrementing cell is linked cannot be found.
- The application to which an auto-incrementing cell is linked is available, but an error occurs while requesting the next value.
- System 7 (or later) is not running but is required in order for the auto-increment to work.

It’s normal to expect the ‘application not found’ message in certain situations. Since Informed uses the name of the computer to find the application to which an auto-incrementing cell is linked, this error will occur if you change the computer’s name, or if you move the application to a different computer with a different name. You’ll see an error message the first time Informed attempts to obtain the next value for the auto-incrementing cell.



The message dialog box contains the 'Look,' 'Skip,' and 'Disable' buttons. You can try to find the required application by clicking the 'Look' button. You'll see the Program Linking dialog box shown earlier. If you successfully find the correct application, Informed will automatically re-link the auto-incrementing cell before continuing. If you click 'Skip' or 'Disable,' the link will be ignored and no attempt will be made to find the missing application. 'Skip' ignores the link that time only which means the error message will appear the next time a new number is requested. Clicking 'Disable' ignores the link until the next time the form template document is opened.

The second type of error can occur when the required application is available but an error from that application is detected. The application may provide information about the error that has occurred. For example, if you've configured an auto-incrementing cell to obtain the next invoice number from Informed Number Server, but Informed Number Server is unable to find the requested number, you'll see this error message:



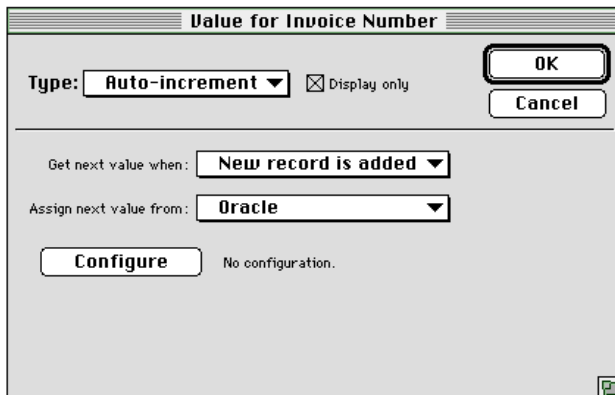
If you click the 'Retry' button, Informed Filler will attempt to request the next value again. This option is useful if the application is running on a different computer since you might be able to fix the problem there and then continue. As explained earlier, the 'Skip' and 'Disable' options allow you to ignore the error. Clicking 'Skip' ignores the error one time only, whereas the 'Disable' option ignores the error until the next time the form document is opened.

The third type of error will occur if you're not running System 7. Auto-incrementing cells that are linked to Apple event aware applications work only if you're using system software version 7 or later, of the Mac OS operating system.

## Linking to Other Data Sources

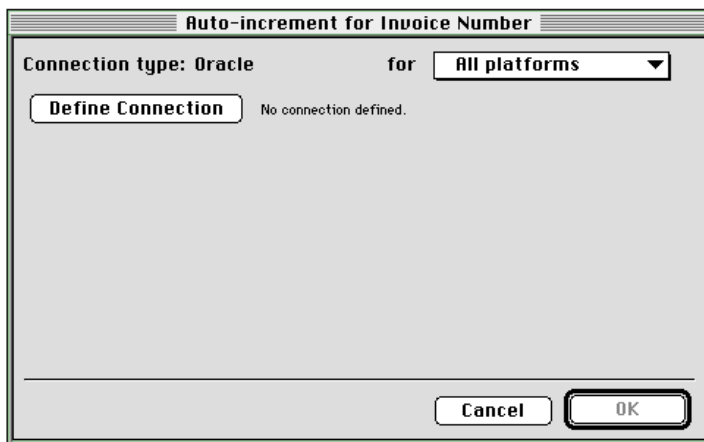
Informed's data access plug-ins are designed to allow access to a wide range of databases and data sources. They include support for many of the standard desktop database formats as well as common SQL databases such as Oracle and Sybase. Informed's plug-in architecture allows Shana to continually develop new plug-ins and update existing plug-ins to support new databases and new standards in data access.

When you choose an item from the 'Assign next value from' drop-down list that corresponds to a data access plug-in, a 'Configure' button appears on the Value dialog box.



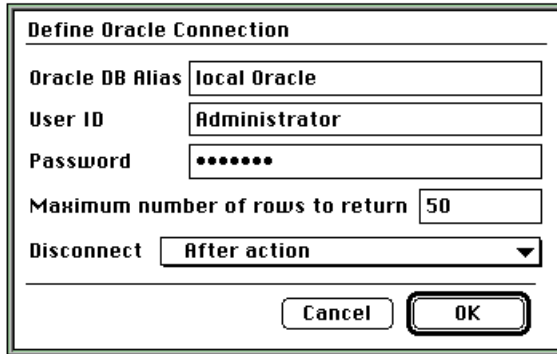
The screenshot shows a dialog box titled "Value for Invoice Number". At the top, there is a "Type:" dropdown menu set to "Auto-increment" and a checked checkbox labeled "Display only". To the right are "OK" and "Cancel" buttons. Below this, there are two more dropdown menus: "Get next value when:" set to "New record is added" and "Assign next value from:" set to "Oracle". At the bottom left is a "Configure" button, and to its right is the text "No configuration." A small help icon is in the bottom right corner.

Clicking 'Configure' displays a configuration dialog box that allows you to specify the connection information and data source-specific instructions for the selected data source. The configuration dialog box for the Oracle data access plug-in is shown below.



The screenshot shows a configuration dialog box titled "Auto-increment for Invoice Number". It displays "Connection type: Oracle" and "for All platforms" with a dropdown arrow. Below this is a "Define Connection" button and the text "No connection defined." At the bottom are "Cancel" and "OK" buttons.

With most databases, it is necessary to provide connection information in addition to the specific instructions that are to be carried out by the data source. Connection parameters usually consist of a user ID, a password, and information that identifies the data source or server. This information is specific to the data source that you're linking to and is specified by clicking the 'Define Connection' button.



The image shows a dialog box titled "Define Oracle Connection". It contains several input fields and a dropdown menu. The fields are: "Oracle DB Alias" with the value "local Oracle", "User ID" with the value "Administrator", "Password" with a masked value of seven dots, "Maximum number of rows to return" with the value "50", and "Disconnect" with a dropdown menu showing "After action". At the bottom of the dialog are two buttons: "Cancel" and "OK".

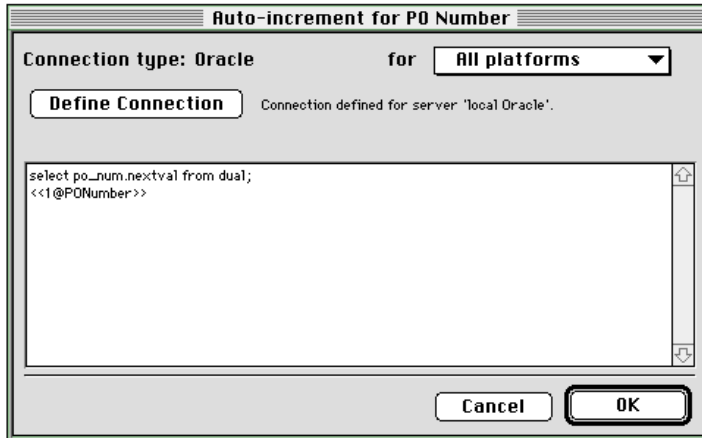
If you enter all necessary connection parameters here, Informed Filler will automatically connect for the user when a new value is requested. You can, however, leave optional parameters blank. Leaving a parameter, such as the user ID or password, blank means that the Informed Filler user will be requested to enter this information when a new value is requested.

The details of connecting to a data source are the same regardless of whether the link is configured for an auto-incrementing cell, a lookup, or for submitting a completed form. The details of the Define Connection dialog box as well as other relevant data source-specific information can be found in the on-line document "DGRPLG.PDF" (Windows) or "Informed Designer Plug-ins" (Mac OS). This document is automatically installed when you install Informed Designer and is viewed using Acrobat Reader (also included with Informed Designer).

**Note**

Before you can configure an auto-incrementing cell to an external data source, the data source (a dBase file, for example) must already exist. Informed Designer will not create the database or data source for you.

Once you've defined the connection to the data source, a large text box appears allowing you to enter the appropriate instructions for obtaining and incrementing the next available number for the auto-incrementing cell.



The text that you enter is specific to the type of data source that you are connecting to. For example, if you're connecting to an Oracle database, you'll enter SQL statements that conform to Oracle's syntax.

Regardless of which type of data source you're connecting to, the statements or instructions that you enter should instruct the data source to return and increment a single value, that being the next available number for the auto-incrementing cell. That value is then placed in the cell by including a return locator following the data source-specific instructions. A return locator is a means of identifying one of potentially many return values, along with the cell on the form into which the value should be placed.

A return locator is specified as the name of a cell enclosed within double less-than and double greater-than characters. The cell name is preceded by a number and the "@" symbol. The number identifies the return value and the cell name identifies the cell into which the value should be placed. If, for example, the data source returns three values, you would use the numbers 1, 2, and 3, respectively, to identify those values. Since the instructions for an auto-increment configuration are intended to return a single value, the number preceding the "@" symbol will most often be 1. For example, the statements below instruct an Oracle server to return and increment the value of an Oracle sequence number named 'po\_num.'

```
select po_num.nextval from dual;
<<1@PONumber>>
```

The 'select' statement returns a single result. The return locator "<<1@PONumber>>" places the return value in the cell named "PONumber." Although uncommon, the instructions for an auto-incrementing cell can return multiple values if desired. By using multiple return locators, multiple return values can be placed in multiple different cells on the form.

If you want to include a cell's value in the instructions, simply enter the cell's name within double less-than and double greater-than characters.

## Configuring for Multiple Platforms

Many of the databases and data sources that Informed can link with are accessible from both the Windows and Mac OS platforms. However, the details of accessing a database or data source from each of the platforms might be different. For example, suppose that you're linking an auto-incrementing cell to an Oracle database. For Mac OS users, you might be accessing the Oracle database using the Mac OS Oracle client software (SQL\*NET), whereas on Windows you might be using ODBC instead. The specific parameters needed to connect to the database, therefore, might be different depending on which platform the Informed Filler user is using.

Since the 'Apple event' option is available only on the Mac OS, the linking of an auto-incrementing cell to an Apple event-aware application takes effect only for Informed Filler users with Mac OS compatible computers. For accessing databases or data sources through data access plug-ins, the dialog box for configuring an auto-incrementing cell contains a drop-down list with the items 'This platform' and 'All platforms.'



For each different data access plug-in, Informed Designer knows if the configuration details are the same or different for the two platforms. If they're the same, the 'All platforms' option will be available and the auto-incrementing cell you configure on one platform will function on both.

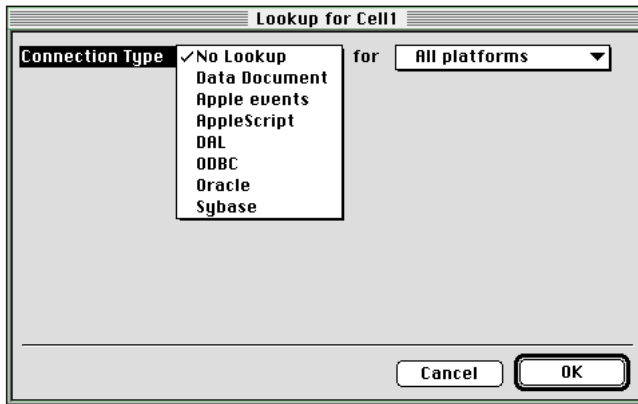
If the configuration details are different for each platform, 'This platform' will be the only choice available in the drop-down list. For accessing these types of databases, you have to configure the auto-incrementing cell on one platform, then move the form template to the other platform and repeat the configuration. Informed Designer stores the configuration for both platforms. Informed Filler uses the configuration that corresponds to the user's platform.

Although it may be necessary to configure an auto-incrementing cell twice, once on each platform, the resulting form template document is still a platform neutral document. That is, a single version of the template will work with Informed Filler on both platforms. Informed Filler automatically uses the configuration information that's appropriate for the user's platform.

## Using Lookups

Lookups are an important time saving feature. Information that exists on other forms, or in other databases or information systems can be looked up as forms are filled out. An information system can range anywhere from a small desktop database, to a high capacity, high performance SQL database running on a mainframe. By using lookups, you reduce the amount of data entry required to complete a form. This increases the productivity of Informed Filler users, reduces errors, and helps to ensure that the information entered is current.

There are several different methods with which Informed Filler can access and look up information. Informed Designer's Lookup dialog box contains a drop-down list that lists each of the different methods.



The 'Data Document' connection type allows you to look up information in other data documents containing data for other forms. The 'Apple events' and 'AppleScript' connection types are available only on the Mac OS. All other connection types correspond to the data access plug-ins that you have installed in your plug-ins folder.

## How it Works

Lookups are configured for individual cells that contain the information to be looked up. For example, if you want Informed Filler to look up inventory information when the user enters a part number on an invoice form, you'd configure the cell that contains the part number to perform the lookup. This cell is called the lookup cell. While filling out forms with Informed Filler, typing a part number would trigger the lookup and, in turn, fill in the related inventory information in the appropriate cells on the form.

Configuring a lookup is a three step procedure:

- Select the lookup cell and choose the Lookup command
- Choose the connection type
- Specify the configuration details

The Lookup dialog box contains controls for selecting the connection type and configuring the lookup. Select the lookup cell then choose the Lookup command to display the Lookup dialog box.

As a shortcut, you can also click the Lookup button on the Cell palette to display the Lookup dialog box. For more information about the Cell palette, see "Using the Cell Palette" later in this chapter.



The 'Connection type' drop-down list contains a list of all available connection types. This list will always include 'Data document' and, on the Mac OS, 'Apple events' and 'AppleScript.' The AppleScript option is available only if AppleScript is installed and active (AppleScript comes with System 7.5 or later).

Other connection types correspond to the data access plug-ins that you have installed in your plug-ins folder. They might include 'Oracle,' 'Sybase,' 'ODBC,' as well as others.

The first connection type is 'No Lookup.' To clear a previously configured lookup, select this item then click 'OK' on the Lookup dialog box.

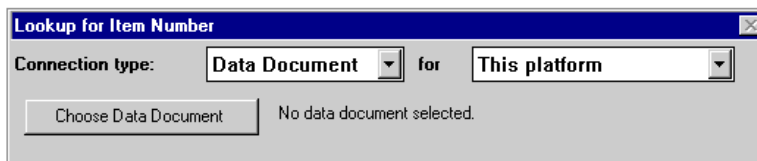
When you select a connection type, the Lookup dialog box changes to display the configuration controls and settings appropriate for that type of connection. For example, the 'Data document' connection type has controls for selecting a data document and linking cells, whereas the AppleScript option allows you to select a script.

Once you've selected the connection type and specified all configuration settings, click 'OK' to save the lookup settings. The following sections describe the procedures for configuring the different types of lookups.

## Data Document Lookups

Information can be looked up in data documents containing information for other forms. That way, information that has already been entered on a different form can be looked up to avoid retyping.

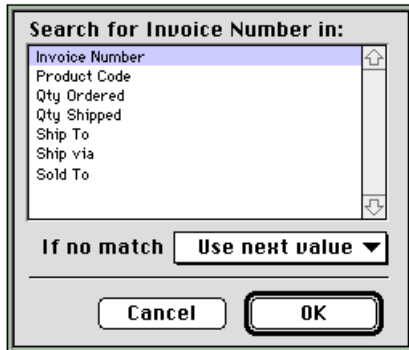
With the 'Data Document' connection type selected, the Lookup dialog box contains a button titled 'Choose Data Document.'



Clicking 'Choose Data Document' displays the standard Open dialog box allowing you to select a data document. The document that you select is the one in which information is looked up when the lookup is performed. Informed Designer reads the list of cells in the data document you select and displays them in a list labelled 'Remote data.' A second list contains the names of the cells on your form template. With a data document selected, the 'Choose Data Document' button changes to 'Clear Lookup.' You can click this button to clear the linking information.

When the user enters a value in the lookup cell, Informed Filler searches for that value in the match cell in the data document. For example, you might enter an invoice number in the lookup cell on a packing slip to look up common information in a corresponding invoice form and enter it on the packing slip. The invoice number cell in the invoice data document would be selected as the match

cell. Choose the match cell by first clicking the 'Choose Match Cell' button. A list containing the cells in the remote data document appears.



Select the appropriate cell in the list then click 'OK.' The name of the match cell will appear next to the 'Choose Match Cell' button on the Lookup dialog box.

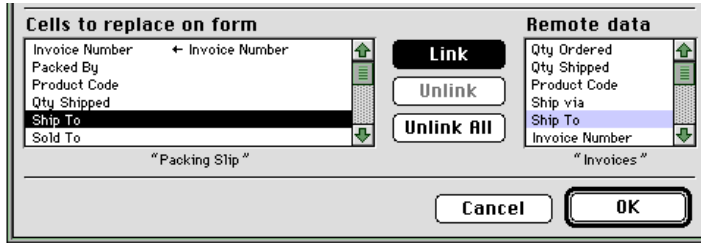
The match cell in the lookup data document must be a field cell. You cannot perform a lookup into a table cell. When a lookup is performed, Informed Filler searches through all records in the lookup data document comparing the value entered on the form with that of the match cell.

#### Note

The time that it takes Informed Filler to perform a lookup varies depending on how many records are in the lookup data document and whether or not the match cell is indexed. For optimal performance, be sure to index the match cell. See "Indexes" earlier in this chapter for more information.

The 'If no match' drop-down list contains two options. The option you choose determines how Informed Filler will function if the search for a lookup value fails. If you choose the 'Use next value' option, Informed Filler will use the next higher value in the lookup data document. If you choose the 'Do not lookup' option, Informed Filler won't copy any cell values from the data document and will instead clear the linked cells on the form and display a dialog box indicating that no match was found.

The lower section of the Lookup dialog box contains two lists separated by three buttons. You use these lists to link cells on the form (under 'Cells to replace on form') with cells in the data document in which the lookup is performed (under 'Remote data'). When the lookup is performed, the linked cells are filled with the corresponding values of the cells in the data document. You link two cells by clicking one in each of the two lists, then clicking the 'Link' button.



To indicate that two cells are linked, Informed Designer shows the remote cell in the left list with an arrow pointing to the cell on the form. To unlink a cell, select it in the left list, then click ‘Unlink.’ Clicking the ‘Unlink All’ button unlinks all cells.

## Apple Event Lookups

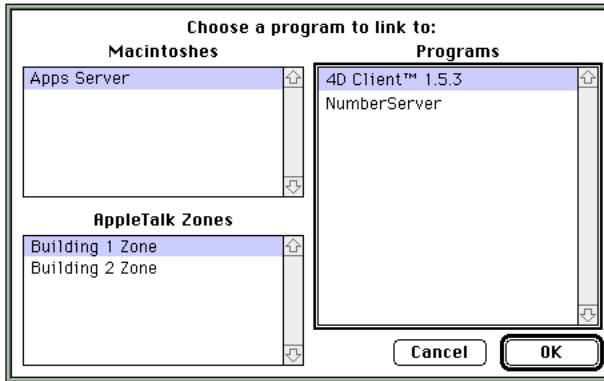
Apple events is an IAC (inter-application communications) capability that’s available only on computers running version 7.0 or later of the Mac OS. Apple events allows two different applications—either on the same computer or on two different computers connected to the same network—to communicate directly with each other.

The ‘Apple events’ connection type is used to link lookups to other Apple event aware applications. Linking, however, works only with applications that understand the specific type of Apple events that Informed uses to communicate. 4th DIMENSION by ACIUS (using the Informed 4D External) is one such application.

Configuring an Apple event connection involves selecting the application and database to connect to, and linking cells on your form with fields in the remote database. With the ‘Apple event’ connection type selected, a button titled ‘Choose Application’ appears on the Lookup dialog box.



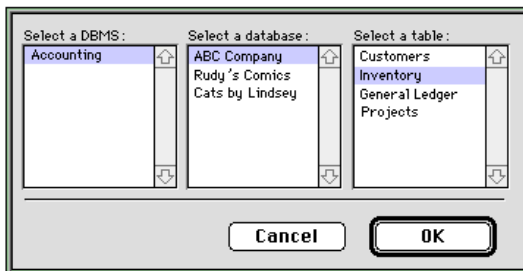
The Apple event application to which you’re linking a lookup must be running when you configure the lookup. Click the ‘Choose Application’ button to select this application. The Program Linking dialog box appears.



The application can be running on any Mac OS compatible computer connected to the network. If the application is running on a computer other than the one that you're running Informed Designer on, that computer must have a name, and it must have program linking turned on. On the Program Linking dialog box, choose the computer that's running the application, then select the application itself and click 'OK.'

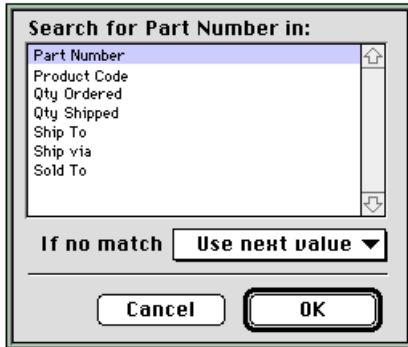
A database or accounting system might contain several different files or tables of information. For example, many accounting systems maintain customer, vendor, inventory, and invoice information in separate files. In database terminology, a file of information is commonly referred to as a table. The individual pieces of information contained in a table—such as a customer's name, address, and terms—are called fields.

Once you've selected an application, click the 'Choose Table' button to choose a particular table to look up into. You'll see a dialog box listing the available options.



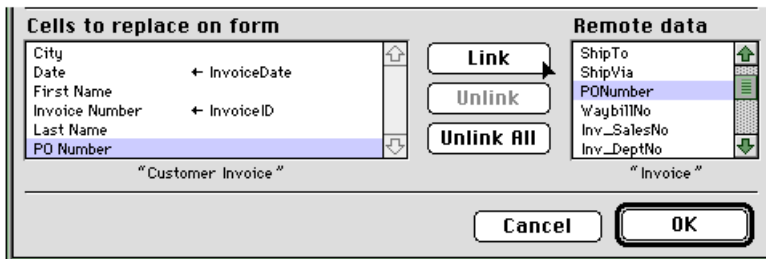
Depending on the application that you've selected, you may or may not see different options in each list. If you select a DBMS (database management system) in the leftmost list, you may see several databases to choose from. Many applications act like a single DBMS and offer access to one or more databases. When you select a database, the third list shows the available tables. Select the appropriate table, then click 'OK.' The name of the table will appear next to the 'Choose Table' button and the 'Remote data' list will contain the fields in that table.

Click the 'Choose Match Field' button to select the match field. This is the field that the application will search through to find a matching value when the lookup is performed. For example, suppose that you're setting up an inventory lookup to fill in the description and price of an item when a part number is typed. The match field would be the part number field in the inventory table of the accounting system or database.



Select the appropriate field in the list. As with data document lookups, the 'If no match' drop-down list controls the action Informed Filler takes if the requested lookup value is not found by the lookup application. If you choose the 'Use next value' option, Informed Filler will use the next higher value. If you choose the 'Do not lookup' option, Informed Filler won't copy any values and will instead clear the linked cells on the form and display a dialog box indicating that no match was found. After selecting the match field and 'If no match' option, click 'OK.' The name of the match field will appear next to the 'Choose Match Field' button on the Lookup dialog box.

To specify which cells on the form are filled in when the lookup is performed, you link cells in the 'Cells to replace on form' list with fields in the 'Remote data' list.



Clicking the 'Link' button links the selected cell in the left list with a selected field in the right list. Clicking 'Unlink' clears the link instead. To unlink all cells, click the 'Unlink All' button.

## AppleScript Lookups

AppleScript provides a flexible method of integrating different applications. AppleScript is the basis of the Mac OS open scripting environment. It provides the means of integrating different Mac OS applications at a very high level. This integration, however, works only with other Mac OS applications that also support AppleScript, such as Claris' FileMaker Pro.

You create an AppleScript lookup by writing a script. The script instructs an application to find specific information and copy it back onto the form. Since AppleScript is itself a scripting language, AppleScript lookups are very flexible, making it easy to customize forms for specific needs. Like any lookup, an AppleScript lookup is triggered whenever the user enters or changes a value in the lookup cell.

The example script below searches in a FileMaker Pro database for the customer number entered in a cell called 'CustNo.' The customer's address and phone number are copied from fields in the FileMaker Pro database back into cells on the form.

```
-- initialize some AppleScript variables
copy "" to theName
copy "" to theAddress
copy "" to thePhone

-- copy the customer number entered on the form to a variable
tell application "Informed Filler"
    copy Cell "CustNo" of Window "Invoice" to theCustNo
end tell

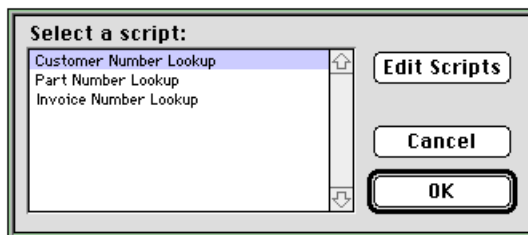
-- search in the FileMaker Pro database for the customer
tell application "FileMaker Pro" of machine "Accounting"
    show (every Record whose Cell "Customer Number" = theCustNo) ↵
        of Window "Customer Database"
    if (Count class Record) of Window "Customer Database" > 0 then
        copy Cell "Name" of Window "Customer Database" to theName
        copy Cell "Address" of Window "Customer Database" to theAddress
        copy Cell "Phone" of Window "Customer Database" to thePhone
    end if
end tell

-- copy the results back onto the form
tell application "Informed Filler"
    copy theName to Cell "Name" of Window "Invoice"
    copy theAddress to Cell "Address" of Window "Invoice"
    copy thePhone to Cell "Phone" of Window "Invoice"
end tell
```

The AppleScript connection type is available only if you're using a Mac OS compatible computer with AppleScript installed. With the AppleScript connection type selected, the Lookup dialog box contains a button titled 'Choose Script.'



Rather than entering an AppleScript script on the Lookup dialog box, you do so using the Scripts command in the Configure submenu of Informed Designer's Settings menu. This command allows you to add, name, remove, and edit scripts. When you click 'Choose Script' on the Lookup dialog box, the list of scripts that have been added to the form template are presented in another dialog box.



You can select a script, or, as a convenience, you can click 'Edit Scripts' to add a new script. Clicking 'Edit Scripts' is a shortcut to choosing the Scripts command from the Configure submenu. For detailed information on adding, naming, editing, and removing scripts, see Chapter 12, "Using AppleScript." For a comprehensive description of the AppleScript language and important background information, please see the *AppleScript Language Guide* (from Apple Computer Inc., available separately).

After selecting a script, click 'OK' on the dialog box. The script name appears to the right of the 'Choose Script' button.

**Note** Since only Informed Filler understands AppleScript scripts, AppleScript lookups cannot be tested using Informed Designer's Test mode. You must use Informed Filler on a Mac OS compatible computer to test an AppleScript lookup.

## Linking Through Data Access Plug-Ins

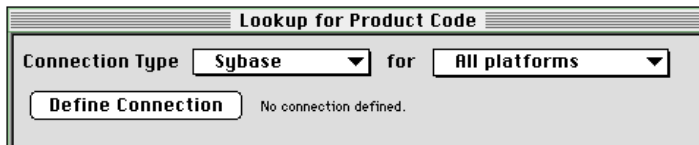
Informed's data access plug-ins are designed to provide access to a wide range of databases and data sources. They include support for many of the standard desktop database formats as well as common SQL databases such as Oracle and Sybase. Informed's plug-in architecture allows Shana to continually develop new plug-ins and update existing plug-ins to support new databases and new standards in data access.

When you choose a connection type that corresponds to a data access plug-in, the Lookup dialog box changes to reflect the specific configuration details for that type. For many connection types there are two methods of configuration: “Easy” and “Custom.” Some connection types provide only one of these methods.

The “Easy” method is intended to provide an easy-to-use method of configuration. Although less flexible, it usually requires very little knowledge of the technical details of the data source. For example, no knowledge of Sybase’s SQL language is necessary when configuring a Sybase lookup using the Easy configuration method.

The “Custom” method of configuration provides much more flexibility, but often requires more knowledge of the data source. Configuring a custom Sybase lookup, for example, requires that you enter an actual SQL query.

After choosing a connection type, a single button appears near the top of the Lookup dialog box. The title of this button is specific to the type of connection you choose. For many connection types, the button title is ‘Define Connection.’



With most data sources, it is necessary to provide connection information in addition to the specific instructions that are to be carried out by the data source. Connection parameters usually consist of a user ID, a password, and information that identifies the data source or server. This information is specific to the data source that you’re linking to. The Define Connection dialog box for Sybase is shown below.

If you enter all necessary connection parameters here, Informed Filler will automatically connect for the user when the lookup is performed. You can, however, leave optional parameters blank. Leaving a parameter, such as the user ID or password, blank means that the Informed Filler user will be requested to enter this information when the lookup is performed.

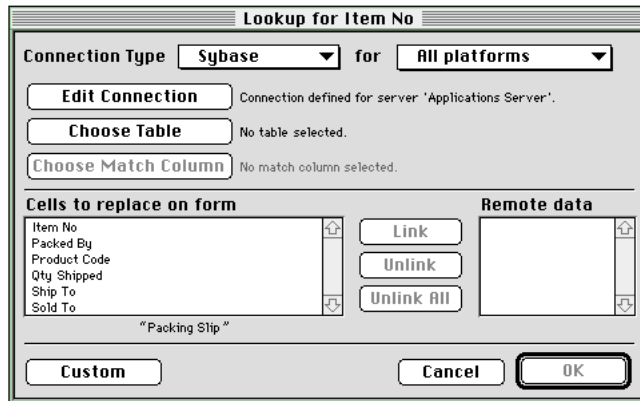


The details of connecting to a particular type of data source are the same regardless of whether the link is configured for a lookup, an auto-incrementing cell, or for submitting a completed form. The details of the Define Connection dialog box as well as other relevant data source-specific information can be found in the on-line document “DGRPLG.PDF” (Windows) or “Informed Designer Plug-ins” (Mac OS). This document is automatically installed when you install Informed Designer and is viewed using Acrobat Reader (also included with Informed Designer).

#### Note

Before you can link a lookup to an external data source, the data source (a dBase file, for example) must already exist. Informed Designer will not create the database or data source for you.

If additional configuration information is needed (which is the case for most data sources), once you’ve defined the connection, the Lookup dialog box will change to show the controls and settings for either of the Easy or Custom configuration methods. If both the Easy and Custom configuration methods are supported by the selected connection type, you’ll see the Easy configuration screen with a button near the bottom-left of the dialog box titled ‘Custom.’



Clicking the ‘Custom’ button switches the Lookup dialog box to show the Custom configuration screen. The button title changes to ‘Easy.’ Clicking ‘Easy’ switches you back to the Easy configuration screen.

## Easy Configuration

If a plug-in supports the Easy configuration method, after you’ve defined the connection, the Lookup dialog box will change to show additional buttons and two scrolling lists. These controls make it very easy to configure the lookup. Once you’ve specified the necessary parameters and links, Informed Designer automatically generates the instructions required by the data source to perform the lookup.

Depending on which connection type you’ve selected, the titles of buttons and the dialog boxes that appear when you click them may vary. These details can be found in the on-line document “DGR-PLG.PDF” (Windows) or “Informed Designer Plug-ins” (Mac OS). This document is automatically installed when you install Informed Designer and is viewed using Acrobat Reader (also included

with Informed Designer). The examples shown in this section correspond to the Sybase connection type.

For many types of databases and data sources, information is stored in “tables.” Each table stores information about a particular type of object or entity. For example, an accounting database may have separate tables for customers, vendors, inventory items, and the chart of accounts. Each table contains columns of information. Each column stores one value of information. A customer table, for example, might have separate columns for the customer number, name, address, and balance.

When a lookup is performed, the value that the Informed Filler user types is looked up in a column of a table. If a match is found, the values of other columns are returned and entered in other cells on the form. To select the lookup table, click the ‘Choose Table’ button.

### Note

In order to choose a table, it is necessary that Informed Designer connect to the data source to obtain the list of available tables. Be sure that the connection has been properly defined and that the database or data source is available before you click ‘Choose Table.’

For Sybase lookups, multiple dialog boxes will appear when you click ‘Choose Table,’ one to select a database and one to select a table. Once you’ve selected a table, the name of the table will appear next to the ‘Choose Table’ button, and the columns in that table will be listed in the ‘Remote data’ scrolling list.

**Lookup for Last Name**

Connection Type: **Sybase** for **All platforms**

**Define Connection** Connection defined for server 'SYBASE'.

**Choose Table** Selected table is 'customers'.

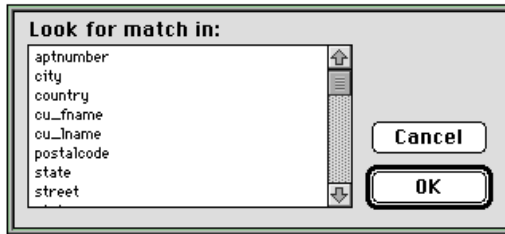
**Choose Match Column** No match column selected.

Cells to replace on form	Remote data
City	city
First Name	country
Last Name	cu_fname
State	cu_lname
Street	state
Zip	street

Buttons: Link, Unlink, Unlink All

Bottom buttons: Custom, Cancel, OK

Click ‘Choose Match Column’ to select the match column. This is the column in the table that the database will search through to find a matching value when the lookup is performed.

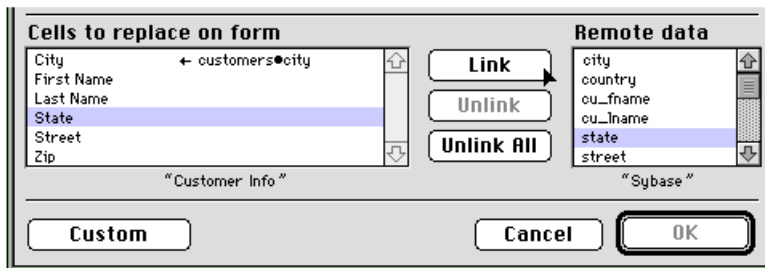


Select the match column, then click 'OK.' The name of the match column appears beside the 'Choose Match Column' button on the Lookup dialog box.

### Note

If the structure of the tables or columns or other elements of the data source changes after you've chosen a table, be sure to choose the table again. In doing so, Informed Designer will connect to the data source and obtain the current database structure.

The scrolling lists titled 'Cells to replace on form' and 'Remote data' are used together to specify which cells on the form are filled in with information returned from the data source. The 'Remote data' list contains the names of the columns that are available from the data source. The 'Cells to replace on form' list contains all cells on the form template. To specify that a cell value is to be replaced with a return value, simply select the cell in the left list and the return value in the right list, then click the Link button. The name of the return value will appear in the 'Cells to replace on form' list with an arrow pointing towards the cell name.



To unlink one cell, select the cell then click 'Unlink.' To unlink all cells, click 'Unlink All.'

## Custom Configuration

Some data sources support only the Custom configuration method. The Custom configuration method is much more flexible compared to the Easy method, but it requires more knowledge of the database or data source. For data sources that support both Easy and Custom configuration, you switch between the two methods by clicking the 'Custom' / 'Easy' button located near the bottom-left of the Lookup dialog box. With the Custom configuration method selected, the Lookup dialog box changes to contain a large text box (in addition to the 'Define Connection' button). For many types of data sources, you'll also see a button titled 'Auto-Generate.'

**Lookup for Item No**

Connection Type **Sybase** for **All platforms**

**Define Connection** Connection defined for server 'Applications Server'.

**Auto-Generate**

**Easy** **Cancel** **OK**

The Custom configuration method requires that you enter text that instructs the data source to perform the lookup. The text that you enter is specific to the type of data source that you are connecting to. For example, if you're connecting to a Sybase database, you'll enter SQL statements that conform to Sybase's syntax. The example shown below queries a Sybase database to lookup an employee's number and return her name, department, and extension.

**Lookup for Emp No**

Connection Type **Sybase** for **All platforms**

**Define Connection** Connection defined for server 'Applications Server'.

**Auto-Generate**

```
use personnel
go
select emp_no, emp_name, emp_dep, emp_ext
from employees
where convert(varchar, emp_no) like '<<EmpNumber>>%'
<<1@EmpNumber>>
<<2@EmpName>>
<<3@EmpDepartment>>
<<4@EmpExtension>>
```

**Easy** **Cancel** **OK**

Regardless of the type of data source you're connecting to, the statements or instructions that you enter should instruct the data source to return one or more values. To include a cell's value in the instructions, simply enclose the cell name within double-less than and double-greater than characters. In the example shown above, the select statement includes the value of the cell "EmpNumber."

The values returned from the data source are entered in other cells on the form by including return locators following the data source-specific instructions. A return locator is a means of identifying a return value, along with the cell on the form into which the value should be entered.

A return locator is specified as the name of a cell enclosed within double less-than and double greater-than characters. The cell name is preceded by a number and the “@” symbol. The number identifies the return value and the cell name identifies the cell into which the value should be placed. If, for example, the data source returns three values, you would use the numbers 1, 2, and 3, respectively, to identify those values.

Data sources that support both Easy and Custom configuration will contain an ‘Auto-Generate’ button on the Custom Lookup dialog box. Clicking this button will automatically generate the appropriate instructions according to the current Easy configuration settings. This feature is useful if you want to change—or customize—the effect of the Easy configuration in a small way.

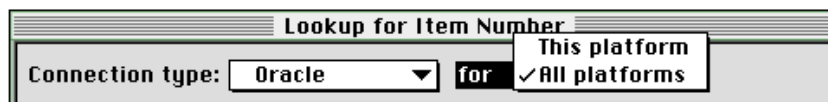
After completing the Easy configuration, switch to the Custom method by clicking the ‘Custom’ button at the bottom-left of the Lookup dialog box. Then click the ‘Auto-Generate’ button. Informed Designer will examine the current Easy configuration and generate the corresponding instructions. These instructions will appear in the large text box as though you had typed them yourself. If the text box already contains instructions, they will be replaced with those automatically generated.

Data sources that do not support Easy configuration can also include the ‘Auto-Generate’ button. The action taken depends on the particular connection type that you’ve selected. These details can be found in the on-line document “DGRPLG.PDF” (Windows) or “Informed Designer Plug-ins” (Mac OS). This document is automatically installed when you install Informed Designer and is viewed using Acrobat Reader (also included with Informed Designer).

## Configuring for Multiple Platforms

Many of the databases and data sources that Informed can link with are accessible from both the Windows and Mac OS platforms. However, the details of accessing a database or data source from each of the platforms might be different. For example, suppose that you’re linking a lookup to an Oracle database. For Mac OS users, you might be accessing the Oracle database using the Mac OS Oracle client software (SQL\*NET), whereas on Windows you might be using ODBC instead. The specific parameters needed to connect to the database, therefore, might be different depending on which platform the Informed Filler user is using.

The Lookup dialog box contains a drop-down list with the items ‘This platform’ and ‘All platforms.’



For each different connection type, Informed Designer knows if the configuration details are the same or different for the two platforms. If the connection type is supported on both platforms and the configuration details are the same on both, the ‘All platforms’ option will be available. The lookup you configure on one platform will function on both.

If the configuration details are different for each platform, or if the connection type is available only on the platform you're using, 'This platform' will be the only choice available in the drop-down list. For accessing these types of databases and data sources, you have to configure the lookup on one platform, then move the form template to the other platform and repeat the configuration. Informed Designer stores the configuration for both platforms. Informed Filler uses the configuration that corresponds to the user's platform.

Although it may be necessary to configure a lookup twice, once on each platform, the resulting form template document is still a platform neutral document. That is, a single version of the template will work with Informed Filler on both platforms. Informed Filler automatically uses the configuration information that is appropriate for the user's platform.

## Lookup Errors

Although Informed Designer can check to make sure that you don't make mistakes when you configure a lookup, there are several reasons why a lookup can fail to work properly. For example, suppose that a lookup is linked to a particular data document, and the document is accidentally deleted. Or maybe the application or database required by a lookup is not running.

Errors can occur while you test a lookup with Informed Designer, or as you fill out forms with Informed Filler. For the three connection types that are built into Informed (Data document, Apple event, and AppleScript), there are five basic types of errors. They are:

- The lookup data document or application cannot be found.
- The data document or application is available, but an error occurs while configuring or performing the lookup.
- System 7 (or later) is not running and is required for Apple event lookups.
- AppleScript is not installed and is required for AppleScript lookups.
- There's an error in your AppleScript script.

For Apple event lookups, Informed uses the name of the computer to find the lookup application. An error will occur if you change the computer's name, or if you move the application to a different computer with a different name. Similarly, an error will occur if the lookup data document required by a lookup cannot be found. You'll see an error message when Informed Filler attempts to perform the lookup for the first time.

The dialog box that appears contains the buttons 'Look,' 'Skip,' and 'Disable.' You can try to find the required lookup data document or application by clicking the 'Look' button. For data document lookups, you'll see the standard Open dialog box. For Apple event lookups, you'll see the Program Linking dialog box instead. If you successfully find the correct lookup data document or application, Informed will automatically re-link the lookup before continuing. If you click 'Skip' or 'Disable,' the error will be ignored and no attempt will be made to find the missing document or application. 'Skip' ignores the error that time only which means the error message will appear the next time the lookup is attempted. The 'Disable' option ignores the error until the next time the form document is opened.

When you choose the Lookup command to change the configuration of an existing lookup, the lookup data document or application must be available. If it's missing, an error message will appear with options to look for the document or application, clear the lookup and continue, or cancel the Lookup command.

The second type of error can occur when the lookup data document or application is available but an error of some sort is detected. For Apple event lookups, the lookup application may provide information about the particular error that has occurred. For example, if you've configured an Apple event lookup to read inventory information from an accounting system, but the required accounting database is not open, you'll see an error message indicating so.

If you click the 'Retry' button, Informed will attempt to perform the lookup again. This option is useful if the lookup application is running on a different computer (from the one that you're running Informed Designer or Informed Filler on), since you might be able to fix the problem there and then continue. As explained earlier, the 'Skip' and 'Disable' options allow you to ignore the error. Clicking 'Skip' ignores the error one time only, whereas the 'Disable' option ignores the error until the next time the form document is opened.

The third and fourth types of errors will occur if you're not running System 7 (or later) or AppleScript when Informed Filler attempts to perform an Apple event or AppleScript lookup, respectively. An error will also occur if there are mistakes in your AppleScript lookup script.

## Data Verification

In "Cell Types" earlier in this chapter, you learned about cell types and formatting options. By choosing the correct cell type, you can restrict a cell's values to those that match the cell type. For example, date cells accept only valid date entries.

Informed Designer allows you to further restrict the allowable values for any cell. Often the value in a cell must be within a certain range or it must conform to specific data entry rules. For example, on a sales slip you might want to restrict the value entered in the discount amount cell to no more than ten percent of the total sale.

By using Informed Designer's Check command, you can create a *check formula* for any cell on your form. A check formula tests for error or warning conditions. The result of a check formula—which must be True or False—indicates whether or not a cell's entry is valid. You can even use alert dialogs and help messages to describe error conditions.

## Check Formulas

Like calculation formulas, you create a check formula by combining operators and functions with cell names and constants to produce a new result. The resulting type of a check formula must be boolean. That is, a check formula must return True or False. A True result indicates a valid entry, whereas a False result represents an invalid value.

Suppose that the shipping charge cell on an invoice should accept only numbers greater than 2 dollars. Assuming that the shipping charge cell is called 'Shipping,' you could use the check formula shown below.

```
Shipping >= 2
```

This formula uses the 'greater than or equal to' comparison operator (>=) to test if the value of 'Shipping' is at least 2. When a person enters or changes the 'Shipping' value, the formula is evaluated using the new value. If the value is greater than or equal to 2, the result of the formula will be True and the value will be accepted. Otherwise, the formula will return False and a beep will sound to indicate that an error has been detected.

You can use Informed's logical operators to combine more than one test condition. For example, suppose that the shipping charge must also be less than or equal to 10 dollars. Using the 'AND' operator, you could combine the two test conditions into one formula.

```
Shipping >= 2 AND Shipping <= 10
```

Only values that are greater than or equal to 2 AND less than or equal to 10 would be accepted.

It's often easier to understand the effect of a check formula if you change the format to use the IF statement. With the IF statement, you can test for different conditions and explicitly return a True or False result. For example, the check formula below is equivalent to the example which allows a shipping charge between 2 and 10 dollars.

```
If Shipping >= 2 AND Shipping <= 10 Then
  Return True
Else
  Return False
End
```

If the condition between the words 'If' and 'Then' is true, then the formula returns the result that follows the word 'Then.' If the condition is false, then the result that follows the word 'Else' is returned instead. The word 'Return' is optional. It can precede the result of a formula.

Often you'd like to test a cell using other cell values on your form. For example, maybe the shipping charge applies only to customers outside of New York state. The following formula uses the IF statement to test the value of the 'State' cell.



```

If State <> 'NY' Then
    Return Shipping >= 2 AND Shipping <= 10
Else
    Return Shipping = 0
End

```

If the value of 'State' is not equal to (<>) the value 'NY,' then the formula proceeds to test if the shipping amount is between 2 and 10. Otherwise, the formula tests for a shipping amount of 0.

If a check formula doesn't return a result, the formula's result is assumed to be True. For example, if the value of 'State' equals 'NY,' then the check formula below would return the True value.

```

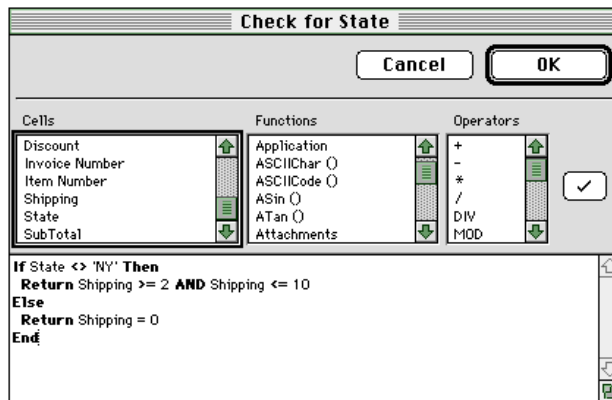
If State <> 'NY' Then
    Shipping >= 2 AND Shipping <= 10
End

```

You can use check formulas to test other types of information too. You can compare date, time, name, and boolean values, and use any of Informed's powerful functions to manipulate and test cell values. For a complete discussion of formulas and functions, see Chapters 9 and 10.

## Entering a Check Formula

To create a check formula, first select the cell that you want to check, then choose **Check...** from the Settings menu. The Check dialog box appears.



You enter a check formula by typing in the formula text box. You can resize the dialog box to show more or less of the check formula.

Informed Designer makes it easy to enter complex, error-free formulas. Instead of typing cell names, functions, and operators, you can double-click any entry in any of the corresponding scrolling lists on the Check dialog box. The entry is inserted into the formula at the current insertion point. You can move between the lists on the dialog box by pressing Tab. When you tab into a list, a bold frame appears around it to indicate that it's selected.

If you double-click to enter a function that has one or more parameters, Informed Designer will automatically position the insertion point at the first parameter. If you double-click a function while holding down the Alt (Windows) or Option (Mac OS) key, the parameter names are included within parentheses.

You can also enter a cell's name by clicking the cell in the drawing window. This is useful if you don't know the name of the cell, but you can see it in the drawing window.

If you click the checkmark button while entering a formula, or if you click 'OK' to dismiss the dialog box, Informed Designer will check to make sure that the formula is valid. The formula is formatted properly, and if any errors are detected, a message appears describing the nature of the error.

## Alert Dialogs and Help Messages

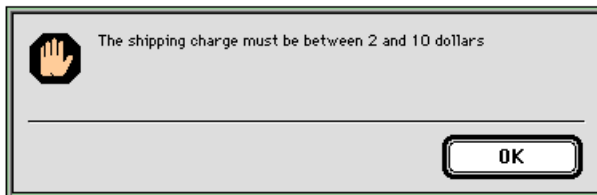
Informed Designer allows you to supply optional messages in any check formula. By using alert dialogs and help messages, you can explain to someone filling out your form why an error occurred, or you can provide helpful reminders of important instructions that must be followed when certain information is entered.

### Alert Dialogs

An alert is a dialog box that contains a text message. If you include the words 'with Alert' and a message within quotation marks following the result of a check formula, Informed will conduct the alert when the cell is checked (see "Evaluating Check Formulas"). For example, the check formula below uses an alert dialog to report an error when an invalid shipping amount is entered.

```
If Shipping >= 2 AND Shipping <= 10 Then
  Return True
Else
  Return False with Alert 'The shipping charge
  must be between 2 and 10 dollars.'
End
```

When the formula is evaluated, the result will be True if the value of 'Shipping' is within the correct range. If the value is out of range, the formula will return False and you'll see this dialog box:



By using the IF statement, you can use different alerts under different conditions. In the above example, the alert is shown only when the formula detects an incorrect shipping amount. Below is a more complicated example.

Suppose that the discount amount on an invoice can be no higher than 15% for purchases under \$5,000, and no higher than 20% for purchases \$5,000 and over. Suppose also that purchases with a discount higher than 15% must be accompanied with a supervisor's signature. Consider the check formula shown below.

```

If TotalSale < 5000 Then
  If Discount Rate > 0.15 Then
    Return False with Alert 'The discount rate
    cannot exceed 15%.'
  End
Else
  If Discount Rate > 0.20 Then
    Return False with Alert 'The discount rate
    cannot exceed 20%.'
  ElseIf Discount Rate > 0.15 Then
    Return True with Alert 'This purchase requires a
    supervisor's signature.'
  End
End

```

The first IF statement tests whether or not the total sale amount (in the cell called 'TotalSale') is less than \$5,000. For sales under \$5,000, an alert is used only if the discount rate is found to be greater than 15%. Since this is an error condition, the formula returns a False result.

If the total sale amount is greater than or equal to \$5,000, the formula tests for a discount rate higher than either 20% or 15%. If the discount rate is higher than 20%, an alert reports the error and the formula returns False. If the discount rate is greater than 15% but less than 20%, an alert reminds the person filling out the form that the purchase must be approved by a supervisor. Since this is not an error condition, the formula returns True and not False.

## Help Messages

In the section "Cell Help," you'll learn how to create a custom help message for any cell on your form. While entering data in a cell on a form, the help message for that cell can be displayed by choosing Informed Filler's Help command. This command is available in Informed Designer's Form menu while you test your form.

When a check formula causes an alert to be displayed, Informed Filler will automatically add the alert text to the custom help message for the cell. Therefore, you can always see a cell's most recent alert message by displaying the help message.

If you replace the word 'Alert' with the word 'Help' in a check formula (as described above), the alert won't be displayed when the check formula is evaluated.

```

If Shipping >= 2 AND Shipping <= 10 Then
  Return True
Else
  Return False with Help 'The shipping charge
  must be between 2 and 10 dollars.'
End

```

If an invalid shipping amount is entered, the quoted message will be added to the help message for the cell being checked.

For more information about help messages and the Help command, see “Cell Help.” For information about testing your form template, see “Testing Your Form.”

## Evaluating Check Formulas

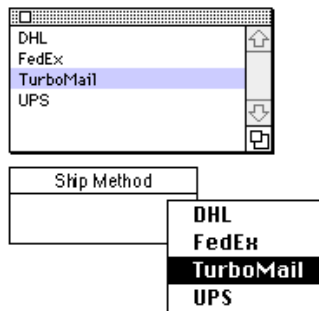
When a form is filled out with Informed Filler, pressing the Tab key moves the user from one cell to the next. Each time they enter a different cell value, Informed Filler checks the value by evaluating the cell’s check formula. If the check formula returns False, a beep will sound and the users will remain at the current cell with the incorrect value selected.

Once an error condition has been detected (as described above), Informed Filler will allow users to proceed to other cells, even if they don’t enter a correct value first. Therefore, the user isn’t forced to correct the mistake before moving to other cells. However, Informed Filler won’t allow accepting, printing, or mailing a form with incorrect values.

## Choices

Often a cell will take on a variety of common values. For example, the shipping method on an order form might be Mail, UPS, or Federal Express. Instead of typing the shipping method each time, the Informed Filler user can select an entry from a list of common choices.

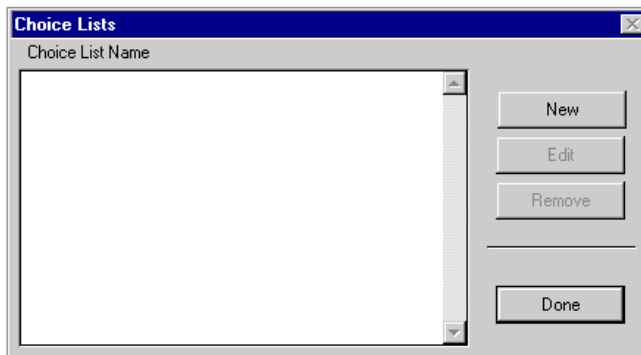
Informed Filler can present a list of choices using either of two methods: a floating palette, or a drop-down list. Both allow the Informed Filler user to pick a choice directly from the list, or type the first few characters of the desired choice.



With Informed Designer, creating and using a choice list is a two step process. First, you name the choice list and specify each of the choices. Then you specify which cells the choice list is to be used with. By creating a choice list as a separate step, a single choice list can be used among several different cells. That way, when you need to change the items in a choice list, you can do so once and have the change take effect for multiple cells.

## Editing Choice Lists

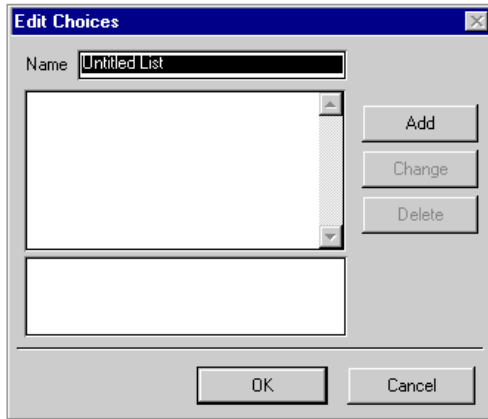
To create or delete a choice list, and to add, remove, or change items in a choice list, select **Choices...** from the Configure submenu under Informed Designer's Form menu. The Choice Lists dialog box appears.



The Choice Lists dialog box contains various controls to add, remove or edit choice lists. It also contains a scrolling list that displays the names of any choice lists that you've already configured for the current form. To add a new choice list, click 'New.' To edit an existing choice list, click it in the scrolling list, then click 'Edit,' or simply double-click it in the scrolling list. When you click 'New' to add a new choice list, or 'Edit' to change an existing choice list, the Edit Choices dialog box appears (see "Editing Choices" below). To remove a choice list, click it in the scrolling list, then click 'Remove.'

## Editing Choices

When you create a new choice list or edit an existing list, you do so using the Edit Choices dialog box. With this dialog box you can add new choices, or delete or change existing choices.



The current choices appear in the scrolling list. To add a new choice, type the entry in the text box below the scrolling list and click ‘New.’ The new choice is added to the list in sorted order. Although there’s no limit to the number of choices a cell can have, we recommend that you enter no more than 50.

To change or remove an existing choice, first select the choice by clicking it in the scrolling list; the selected choice appears in the text box. To remove the selected choice, click ‘Delete.’ To change the selected choice, type the new value then click ‘Change.’

After you’ve entered all of the choices, click ‘OK’ to close the Choices dialog box. To discard any changes, click ‘Cancel’ instead.

## Choice Items

Each item in a choice list can be an actual choice value or a formula that returns one or more values. If you know each choice item in advance, simply add them one at a time on the Edit Choices dialog box.

You can add descriptive text to a cell’s list of choices that won’t appear in the cell once a choice is made. You do this by separating the descriptive text from the choice by two vertical bars (||). For example, if you want the code 0001 to appear in the shipping cell each time you select Airborne Express, enter the choice “Airborne Express || 0001.” When a user selects the choice, only the text to the right of the vertical bars, in this example “0001”, will appear in the cell. In the choice list itself, the actual choice will appear within parentheses to the right of the descriptive text.

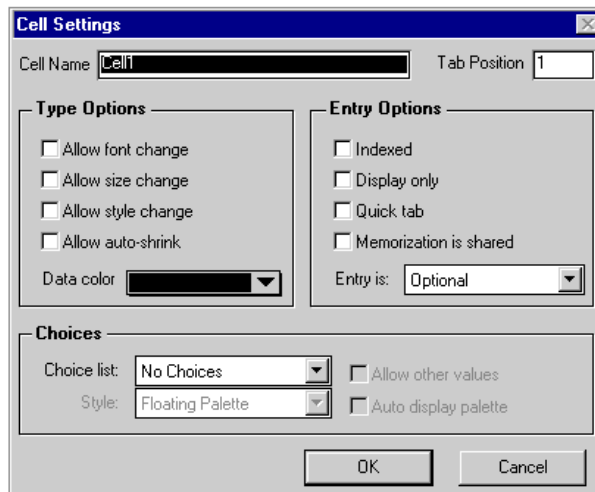
By using formulas, you can create dynamic choice lists that change based on other information on the form. For example, suppose that the available shipping methods for invoiced goods are different depending on whether the goods are shipped to a destination in the United States or in Canada. You might enter a formula that returns a different list of ship method choices depending on the value of the Country cell. When you enter a formula, you must place it within double less-than and double greater-than characters as shown in the following example.

```
<<If Country="USA" Then Return {"FedEx","UPS","Purolator","Mail"} ElseIf
Country="Canada" Then Return {"FedEx","Mail"} End>>
```

The above formula returns the four ship methods “FedEx,” “UPS,” “Purolator,” and “Mail” if the value of the Country cell is “USA.” If Country is “Canada,” the formula returns only “FedEx” and “Mail.” When using formulas for choice items, you enter the formula in the same place you enter choice values—in the text box on the Edit Choices dialog box.

## Configuring Choices for a Cell

Simply defining a choice list does not activate it for a particular cell. To configure a cell to use a choice list, select the cell then choose **Cell...** from the Settings menu. The Cell dialog box appears.



The ‘Choice list’ drop-down list contains the items ‘No choices,’ ‘New List...,’ and the names of any choice lists that you have created. To configure a cell to use a particular choice list, simply select the name of the list from the ‘Choice list’ drop-down list. As a convenience, you can define a new choice list by selecting the ‘New List...’ item. Doing so displays the Edit Choices dialog box with which you can name the choice list and add, remove, or edit choice items. For detailed information on editing a choice list, see “Editing Choices” earlier in this chapter.

When you select a choice list, other options become available. The ‘Style’ drop-down list allows you to pick from two different methods of presenting choices to the Informed Filler user. They include ‘Floating Palette’ and ‘Drop-down List.’

Both styles allow the Informed Filler user to pick a choice either by selecting one directly from the list, or by typing the first few characters of the choice item. To select a choice directly, the user simply clicks the item on the floating palette or selects the item from the drop-down list. Alternatively, as the user types a value in the cell, Informed Filler will automatically enter the choice item that

matches most closely. For the floating palette option, this effect of typing occurs regardless of whether or not the palette is visible.

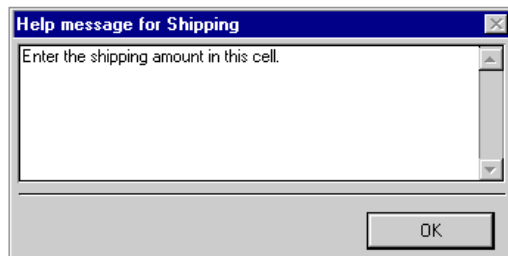
The 'Auto display palette' option is available only if the 'Style' option is set to 'Floating palette.' If you select this option, Informed Filler will automatically display the palette of choices when the user tabs to or selects the cell.

The 'Allow other values' option controls whether or not Informed Filler will allow the user to enter values other than those in the choice list. If you leave this option unchecked, Informed Filler will restrict entries to one of those in the choice list. To allow other values, select the 'Allow other values' option.

## Cell Help

Informed Designer allows you to create a custom help message for any cell on your form template. Use a help message whenever the information in a cell requires explanation or special instructions. Help messages are displayed on dialog boxes or, if you are using a Mac OS compatible computer, they also appear in balloons.

To enter a help message for a cell, select the cell that you want to change, then choose **Help Message...** from the Settings menu. The Help Message dialog box will appear.



A help message can be as long as you like. If you enter more lines than the height of the text box, you can use the scroll bar to scroll the message text up or down.

### Note

Balloon help on the Mac OS sets limitations on the amount of text that can be displayed. If Balloon help does not display the full amount of text in your help message it will still be available on the Help dialog box.

The help messages you create are available to users in Informed Filler, and in Informed Designer's Test mode. When you test a form in Informed Designer you can display the help message by selecting the cell and choosing the Help command in the Form menu.

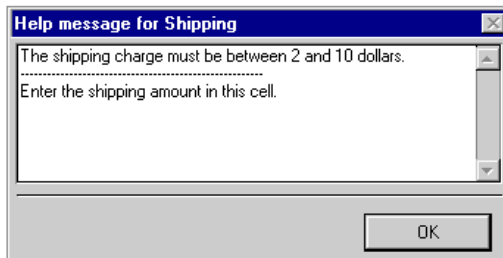


If you're using a Mac OS compatible computer with System 7 or later, you can also display a help message by choosing the Show Balloons command from the Help menu and pointing at the cell. A balloon will appear showing the cell's help message.

## Check Formulas and the Help Dialog

As discussed in "Alert Dialogs and Help Messages," by creating a check formula, you can have Informed Filler automatically display a message or warning when different data entry conditions are met.

When a check formula uses an alert or help message, the message is added to the cell's custom help message. The check formula message is separated from the cell's help message by a dashed line on the Help dialog box.

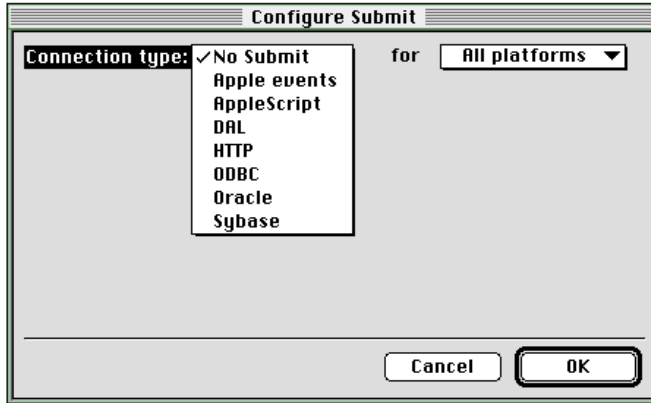


The check formula message will remain part of the help message until the person filling out the form changes the cell's value. For example, if the Informed Filler user enters a cell value that causes an alert message to display, the message will be added to the Help dialog box for that cell (as described above). If they change the value so that the cell's check formula no longer displays the alert, the alert message will be removed from the Help dialog box.

## Form Submission

Automated forms submission allows the Informed Filler user to easily transfer form data from a completed form—or record—into another information system without the need to rekey the data, and without having to follow cumbersome and often complex import and export procedures. Forms can be submitted by simply selecting Informed Filler's Submit command.

There are several methods with which Informed Filler can submit forms. Informed Designer's Configure Submit dialog box contains a drop-down list that lists each of the different methods.



The 'Apple events' and 'AppleScript' connection types are available only on the Mac OS. All other connection types correspond to the data access plug-ins that you have installed in your plug-ins folder.

## How it Works

Form submission requires configuration with Informed Designer. You configure forms submission using the Configure Submit command. You specify the type of connection with which to connect to the submission destination as well as various linking parameters. The linking parameters determine how information is transferred from the form to the submission destination.

Configuring a form for submission is a three step procedure:

- Choose the Submit command from the Configure submenu under Form
- Choose the connection type
- Specify the configuration details

The Configure Submit dialog box contains controls for selecting the connection type and configuring submission. If your computer uses the Mac OS, you'll always see the 'Apple events' and 'AppleScript' connection types. You'll also see the other connections types that correspond to the data access plug-ins you have installed in your plug-ins folder.

The first connection type is 'No Submit.' To clear a previously configured submission, select this item then click 'OK' on the Configure Submit dialog box.

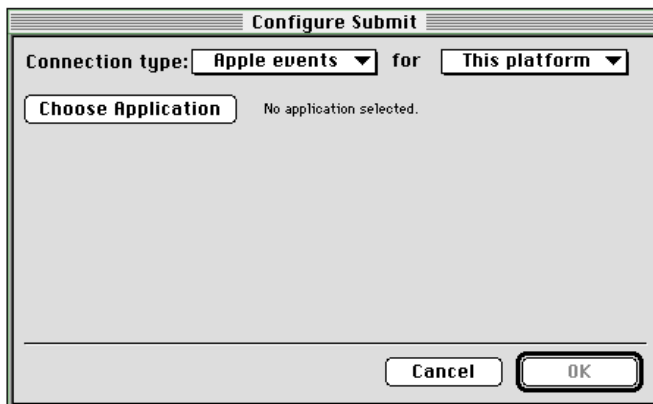
When you select a connection type, the Configure Submit dialog box changes to display the configuration controls and settings appropriate for that type of connection. For example, the Oracle connection type has controls for defining the connection and linking cells, whereas the AppleScript option allows you to select a script.

Once you've selected the connection type and specified all configuration settings, click 'OK' to save the configuration. The following sections describe the procedures for configuring the different types of form submission.

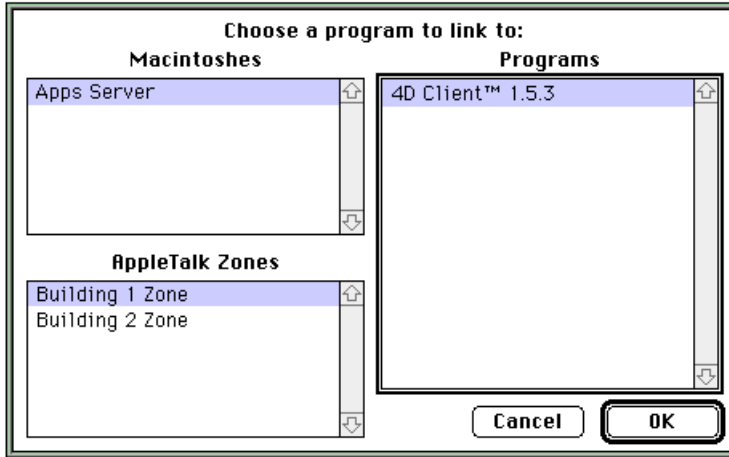
## Apple Event Submission

Informed allows you to submit completed forms to Apple event aware applications such as databases and accounting systems. This feature works only with computers using the Mac OS, and only with applications that understand the specific type of Apple events that Informed uses to communicate. 4th DIMENSION by ACIUS (with the Informed 4D external installed) is one such application.

Configuring form submission using the Apple event connection involves selecting the application and database to connect to, and linking cells on your form with fields in the remote database. With the 'Apple event' connection type selected, a button titled 'Choose Application' appears on the Configure Submit dialog box.



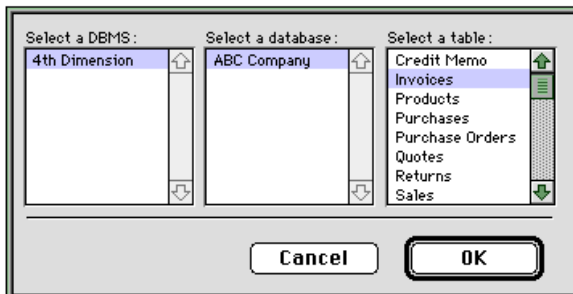
The application to which you link a form must be running when you configure the connection. Click the 'Choose Application' button to select this application. The Program Linking dialog box appears.



The application can be running on any Mac OS compatible computer connected to the network. If the application is running on a computer other than the one that you're running Informed Designer on, that computer must have a name and it must have program linking turned on. On the Program Linking dialog box, choose the computer that's running the application, then select the application itself and click 'OK.' When you click 'OK,' Informed Designer verifies that the application you selected supports the required Apple events to communicate with Informed. If it doesn't you'll see a message indicating so.

Many applications store different types of information in different files or tables. A file of information is commonly referred to as a table. The individual pieces of information contained in a table—such as the invoice number, date, and terms of an invoice—are called fields.

When you link a form to an application, you're required to specify which table the information should be stored in. For example, if you're linking an invoice form to an accounting system, you have to specify that the information will be stored in the invoices table in the accounting database. That way, the application will know how to store the information when a completed form is submitted with Informed Filler. You select a table by clicking the 'Choose Table' button. A dialog box appears listing the options available.



Most applications act like a single DBMS and offer access to one or more databases. When you select a database, the third list shows the available tables. Select the appropriate table, then click 'OK.' The name of the table will appear next to the 'Choose Table' button and the 'Links to remote data' list will show the fields in that table.

**Configure Submit**

Connection type: **Apple events** for **This platform**

**Change Application** "4D Client™ 1.5.3" on "Apps Server"

**Change Table** "4th Dimension:ABC Company :Contacts"

Cells on form	Links to remote data	Req.
City	Contact_No	
Company Name	Company_Name	
Contact ID	Last_Name	
Fax	First_Name	
First Name	Title	
Last Name	Address1	
"Customer Info"	"Contacts"	

**Link** **Unlink** **Unlink All** **Cancel** **OK**

You use the two scrolling lists to link the cells on your form with the fields in the table that you selected. When you use Informed Filler to send a completed form to the application, this linking information is used to ensure that the right values are stored in the right places in the table.

To link two items, select a cell in the left list and a field in the right list, then click the 'Link' button. When a field is linked, Informed Designer shows the cell in the 'Links to remote data' list with an arrow pointing towards the field.

**Cells on form** **Links to remote data** **Req.**

City	Contact_No	
Company Name	Company_Name	
Contact ID	Last_Name → Last_Name	
Fax	First_Name → First_Name	
First Name	Title	
Last Name	Address1	
"Customer Info"	"Contacts"	

**Link** **Unlink** **Unlink All** **Cancel** **OK**

To unlink a field, select it in the right list, then click the 'Unlink' button. To unlink all fields, click the 'Unlink All' button.

For most tables, certain information is required when you send a completed form with Informed Filler. For example, when you send an invoice to an application, it might be required that you include values for fields such as the customer number, invoice number, date, and terms. On the Configure Submit dialog box, a check mark under 'Req.' next to a field in the 'Links to remote fields' list indicates that a cell must be linked to that field.

## AppleScript Submission

Like AppleScript lookups, you configure form submission via the AppleScript connection type by writing a script. The script instructs an application to copy information from the form and insert it into another application. Since AppleScript is actually a scripting language, a script can be written to accomplish virtually any task. The AppleScript connection type is available only on Mac OS compatible computers with AppleScript installed.

The example below shows a script that submits a new customer into a FileMaker Pro database.

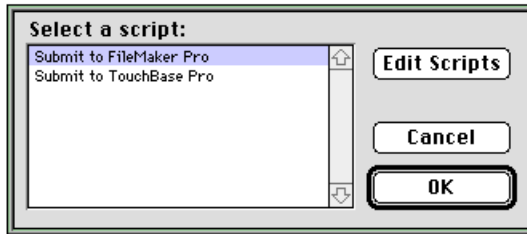
```
-- copy cells on form to AppleScript variables
tell application "Informed Filler™"
  copy cell "Company Name" to theCoName
  copy cell "Address" to theAddress
  copy cell "City" to theCity
  copy cell "State" to theState
  copy cell "ZIP" to theZIP
  copy cell "Phone" to thePhone
  copy cell "Contact" to theContact
end tell

-- create a new record in the FileMaker Pro database with data from form
tell application "FileMaker Pro v3.0v3"
  create new record at window "Customer Database" ↵
    with data {theCoName, theAddress, theCity, ↵
      theState, theZIP, thePhone, theContact}
end tell
```

With the AppleScript connection type selected, the Configure Submit dialog box contains a button titled 'Choose Script.'



Rather than entering an AppleScript script on the Configure Submit dialog box, you do so using the Scripts command in the Configure submenu of Informed Designer's Settings menu. This command allows you to add, name, remove, and edit scripts. When you click 'Choose Script' on the Configure Submit dialog box, the list of scripts that have been added to the form are presented in another dialog box.



You can select a script, or, as a convenience, you can click ‘Edit Scripts’ to add a new script. Clicking ‘Edit Scripts’ is a shortcut to choosing the Scripts command from the Configure submenu. For detailed information on adding, naming, editing, and removing scripts, see Chapter 12, “Using AppleScript.” For a comprehensive description of the AppleScript language and useful background information, please see the *AppleScript Language Guide* (from Apple Computer Inc, available separately).

After selecting a script, click ‘OK’ on the dialog box. The script name appears next to the ‘Choose Script’ button on the Configure Submit dialog box.

## Submission Through Data Access Plug-Ins

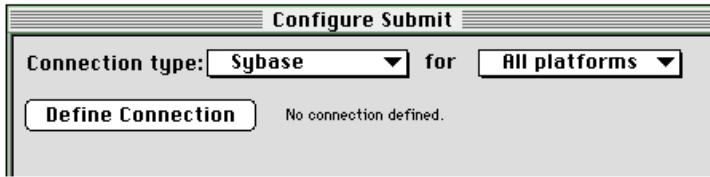
Informed’s data access plug-ins are designed to provide access to a wide range of databases and data sources. They include support for many of the standard desktop database formats as well as common SQL databases such as Oracle and Sybase. Informed’s plug-in architecture allows Shana to continually develop new plug-ins and update existing plug-ins to support new databases and new standards in data access.

When you choose a connection type that corresponds to a data access plug-in, the Configure Submit dialog box changes to reflect the specific configuration details for that type. For many connection types there are two methods of configuration: “Easy” and “Custom.” Some connection types provide only one of these methods.

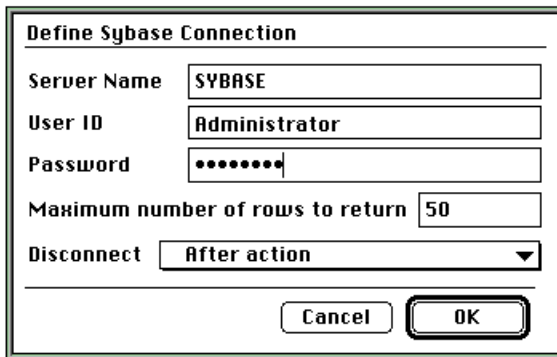
The “Easy” method is intended to provide an easy-to-use method of configuration. Although less flexible, it usually requires very little knowledge of the technical details of the data source. For example, no knowledge of Sybase’s SQL language is necessary when configuring submission to Sybase using the Easy configuration method.

The “Custom” method of configuration provides much more flexibility, but often requires more knowledge of the data source. When configuring custom submission to Sybase, for example, you are required to enter an actual Sybase SQL query.

After choosing a connection type, a single button appears near the top of the Configure Submit dialog box. The title of this button is specific to the type of connection you choose. For many connection types, the button title is ‘Define Connection.’



With most data sources, it is necessary to provide connection information in addition to the specific instructions that are to be carried out by the data source. Connection parameters usually consist of a user ID, a password, and information that identifies the data source or server. This information is specific to the data source that you're linking to. The Define Connection dialog box for Sybase is shown below.



If you enter all necessary connection parameters here, Informed Filler will automatically connect for the user when a form is submitted. You can, however, leave optional parameters blank. Leaving a parameter, such as the user ID or password, blank means that the Informed Filler user will be requested to enter this information when a form is submitted.

The details of connecting to a particular type of data source are the same regardless of whether the link is configured for a lookup, an auto-incrementing cell, or for submitting a completed form. The details of the Define Connection dialog box as well as other relevant data source-specific information can be found in the on-line document "DGRPLG.PDF" (Windows) or "Informed Designer Plug-ins" (Mac OS). This document is automatically installed when you install Informed Designer and can be viewed using Acrobat Reader (also included with Informed Designer).

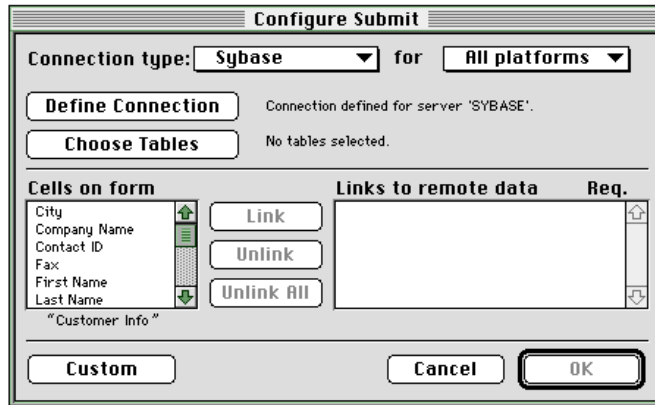
#### Note

Before you can configure form submission with an external data destination, the data destination (a dBase file, for example) must already exist. Informed Designer will not create the data destination for you.

If additional configuration information is needed (which is the case for most data sources), once you've defined the connection, the Configure Submit dialog box will change to show the controls and settings for either of the Easy or Custom configuration methods. If both the Easy and Custom



configuration methods are supported by the selected connection type, you'll see the Easy configuration screen with a button near the bottom-left of the dialog box titled 'Custom.'



Clicking the 'Custom' button switches the Configure Submit dialog box to show the Custom configuration screen. The button title changes to 'Easy.' Clicking 'Easy' switches you back to the Easy configuration screen.

## Easy Configuration

If a plug-in supports the Easy configuration method, after you've defined the connection, the Configure Submit dialog box will change to show additional buttons and two scrolling lists. These controls make it very easy to configure form submission. Once you've specified the necessary parameters and links, Informed Designer automatically generates the instructions required by the data source to perform the lookup.

Depending on which connection type you've selected, the titles of buttons and the dialog boxes that appear when you click them might vary. These details are found in the on-line document "DGR-PLG.PDF" (Windows) or "Informed Designer Plug-ins" (Mac OS). The examples shown in this section correspond to the Sybase connection type.

For many types of databases and data sources, information is stored in "tables." Each table stores information about a particular type of object or entity. For example, an accounting database may have separate tables for customers, vendors, inventory items, invoices, and the chart of accounts. Each table contains columns of information. Each column stores one value of information. An invoice header table, for example, might have separate columns for the invoice number, date, customer number, and shipping address.

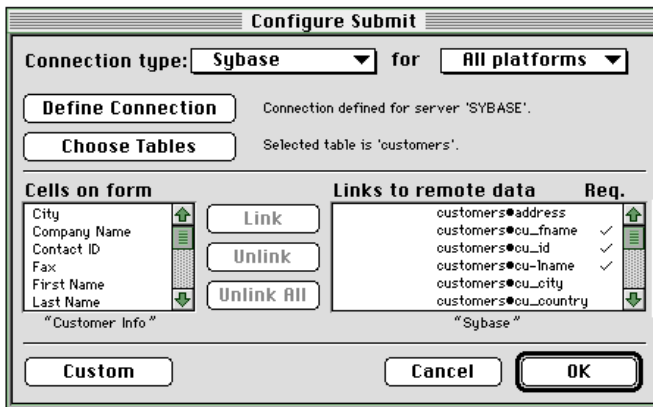
When a form is submitted, the information in cells is transferred into different columns in the data source. These columns do not all have to be from the same table. For example, the cells on an invoice form might be submitted into two tables, an invoice header table (containing the invoice number, customer number, date, and terms), and an invoice detail table (containing the invoice number, part number, quantity, and price for each item invoiced). Many databases offer one or more

tables into which form data can be submitted. You select which table—or tables— by clicking the ‘Choose Tables’ button.

**Note**

In order to choose a table, it is necessary that Informed Designer connect to the data destination to obtain the list of available tables. Be sure that the connection has been properly defined and that the database or data destination is available before you click ‘Choose Tables.’

For Sybase, multiple dialog boxes will appear when you click ‘Choose Tables,’ one to select a database and one to select one or more tables. Once you’ve selected one or more tables, the corresponding columns will be listed in the ‘Links to remote data’ scrolling list. Each column name will be prefixed with the name of the table to which it belongs.



Informed Designer automatically knows how to submit forms containing both fields (single value field cells) and tables (multi-value column cells). For many data destinations, including most SQL databases, Informed Filler will insert multiple rows into tables containing columns that are linked to column cells on the form. The number of rows inserted corresponds to the number of rows filled out in the table on the form being submitted. If both column cells and field cells are linked to the same table (which is common when submitting forms to relational databases), then the single value of a linked field cell is repeated for each row that is inserted.

The scrolling lists titled ‘Cells on form’ and ‘Links to remote data’ are used together to specify which cells on the form are entered into which columns in the data source. The ‘Cells on form’ list contains all cells on the form template. The ‘Links to remote data’ list contains the names of the columns in the data source. To specify that a cell value is to be entered into a column, simply select the cell in the left list and the column in the right list, then click the Link button. The name of the cell will appear in the ‘Links to remote data’ list with an arrow pointing towards the column name.

The screenshot shows a dialog box with two main sections: "Cells on form" and "Links to remote data".

Cells on form	Links to remote data	Req.
City	customers#address	✓
Company Name	customers#cu_fname	✓
Contact ID	customers#cu_id	✓
Fax		
First Name	customers#cu_fname	✓
Last Name	customers#cu_lname	✓
	customers#cu_city	
	customers#cu_country	

Buttons: Link, Unlink, Unlink All, Custom, Cancel, OK.

To unlink one cell, select the cell then click 'Unlink.' To unlink all cells, click 'Unlink All.'

## Custom Configuration

Some data destinations support only the Custom configuration method. The Custom configuration method is much more flexible compared to the Easy method, but it requires more knowledge of the database or data destination. For data destinations that support both Easy and Custom configuration, you switch between the two methods by clicking the 'Custom' / 'Easy' button located near the bottom-left of the Configure Submit dialog box. With the Custom configuration method selected, the Configure Submit dialog box changes to contain a large text box (in addition to the 'Define Connection' button). For many types of data destinations, you'll also see a button titled 'Auto-Generate.'

The screenshot shows the "Configure Submit" dialog box with the "Custom" configuration method selected. The "Connection type" is set to "Sybase" and "All platforms" is selected. The "Define Connection" button is active, and the "Auto-Generate" button is also visible. A large text box is present for entering the connection details. The "Easy" button is also visible at the bottom left.

The Custom configuration method requires that you enter text that instructs the data destination to perform the form submission. The text that you enter is specific to the type of data destination that you are connecting to. For example, if you're connecting to a Sybase database, you'll enter SQL statements that conform to Sybase's syntax, whereas if you're connecting to a web server, you'll enter an HTTP Post request. The following example submits an invoice into two tables of a Sybase database.

```

begin transaction
use accounting
go
insert into InvoiceHdr (inv_no, cust_no, inv_date, terms, ship_meth)
  values (convert(varchar, '<<Invoice Number>>'), convert(varchar, '<<Customer
Number>>'),
        convert(datetime, '<<Date>>'), convert(char, '<<Terms>>'),
        convert(varchar, '<<Ship Method>>'))
<<#LOOP>>
insert into InvoiceDtl (inv_no, part_no, quantity, price)
  values (convert(varchar, '<<Invoice Number>>'), convert(varchar,
'<<PartNumber>>'),
        convert(int, '<<Quantity>>'), convert(money, '<<Price>>'))
<<#ENDLOOP>>
commit transaction

```

Regardless of the type of data destination you're connecting to, the statements or instructions that you enter should instruct the data destination to accept one or more values. To include a cell's value in the instructions, simply enclose the cell name within double-less than and double-greater than characters. In the example shown above, the insert statements include the values of the cells "Invoice Number," "Customer Number," "Date," "Terms," "Ship Method," "Part Number," "Quantity," and "Price."

In order to insert the multiple values of a column cell, Informed provides a looping mechanism. Without this mechanism, the reference to a column cell will correspond to only the first row of that cell. For example, the example SQL statement below will insert one row into the table named "InvoiceDtl." The first value of the column cells "PartNumber," "Quantity," and "Price" would be inserted.

```

insert into InvoiceDtl (inv_no, part_no, quantity, price)
  values (convert(varchar, '<<Invoice Number>>'), convert(varchar,
'<<PartNumber>>'), convert(int, '<<Quantity>>'), convert(money, '<<Price>>'))

```

If you surround statements with the keywords "<<#LOOP>>" and "<<#ENDLOOP>>," Informed Filler will automatically repeat those statements once for each non-empty row of any column cells referenced.

```

<<#LOOP>>
insert into InvoiceDtl (inv_no, part_no, quantity, price)
  values (convert(varchar, '<<Invoice Number>>'), convert(varchar,
'<<PartNumber>>'), convert(int, '<<Quantity>>'), convert(money, '<<Price>>'))
<<#ENDLOOP>>

```

If the table containing the column cells "PartNumber," "Quantity," and "Price" contained four rows of information, the insert statement would be repeated four times.

Data destinations that support both Easy and Custom configuration will contain an 'Auto-Generate' button on the Configure Submit dialog box. Clicking this button will automatically generate the appropriate instructions according to the current Easy configuration settings. This feature is useful if you want to change—or customize—the effect of the Easy configuration in a small way.

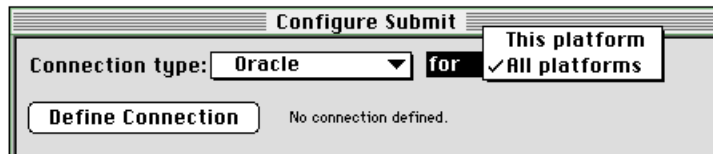
After completing the Easy configuration, switch to the Custom method by clicking the ‘Custom’ button at the bottom-left of the Configure Submit dialog box. Then click the ‘Auto-Generate’ button. Informed Designer will examine the current Easy configuration and generate the corresponding instructions. These instructions will appear in the large text box as though you had typed them yourself. If the text box already contains instructions, they will be replaced with those automatically generated.

Data sources that do not support Easy configuration can also include the ‘Auto-Generate’ button. The action taken depends on the particular connection type that you’ve selected. These details can be found in the on-line document “DGRPLG.PDF” (Windows) or “Informed Designer Plug-ins” (Mac OS).

## Configuring for Multiple Platforms

Many of the databases and data destinations that Informed can link with are accessible from both the Windows and Mac OS platforms. However, the details of accessing a database or data source from each of the platforms might be different. For example, suppose that you’re configuring form submission to an Oracle database. For Mac OS users, you might be accessing the Oracle database using the Macintosh Oracle client software (SQL\*NET), whereas on Windows you might be using ODBC instead. The specific parameters needed to connect to the database, therefore, might be different depending on which platform the Informed Filler user is using.

The Configure Submit dialog box contains a drop-down list with the items ‘This platform’ and ‘All platforms.’



For each different connection type, Informed Designer knows if the configuration details are the same or different for the two platforms. If the connection type is supported on both platforms and the configuration details are the same on both, the ‘All platforms’ option will be available. The linking you configure on one platform will function on both.

If the configuration details are different for each platform, or if the connection type is available only on the platform you’re using, ‘This platform’ will be the only choice available in the drop-down list. For accessing these types of databases and data destinations, you have to configure form submission on one platform, then move the form template to the other platform and repeat the configuration. Informed Designer stores the configuration for both platforms. Informed Filler uses the configuration that corresponds to the user’s platform.

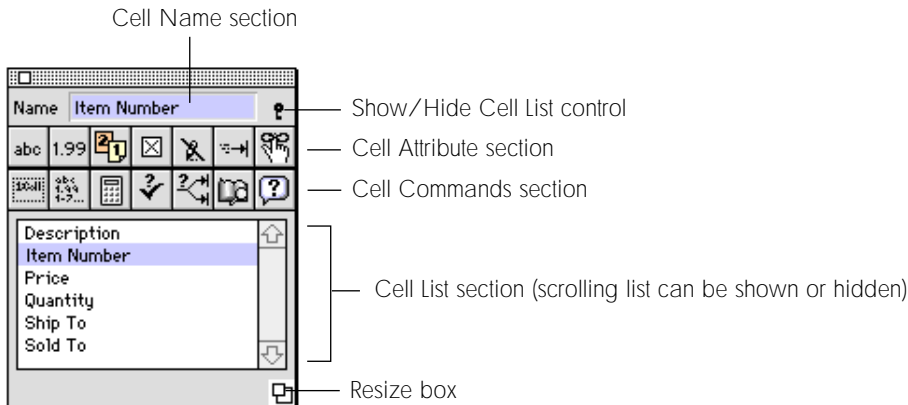
Although it may be necessary to configure form submission twice, once on each platform, the resulting form template document is still a platform neutral document. That is, a single version of the template will work with Informed Filler on both platforms. Informed Filler automatically uses the configuration information that is appropriate for the user's platform.

## Using the Cell Palette

The Cell palette is a convenient feature that provides quick access to cell settings. Using the Cell palette, you can quickly and easily perform the following tasks:

- change the name and title of a cell
- easily find a cell by name in the Cell List
- configure a selected set of attributes for a cell by clicking buttons on the palette
- access Informed Designer's cell settings commands with the click of a button

You can show the Cell palette by choosing **Cell Palette** from the Show submenu under Layout. To hide the Cell palette, click its close box or choose **Cell Palette** again. The following figure shows the parts of the Cell palette.



The top section of the Cell palette is called the Cell Name section. The Cell Name section displays the name of the currently selected cell, and can be used to change both the name and title of the cell. With no cells selected, the Cell Name section displays as a gray bar with the words "No cells selected." If a single cell is selected, the Cell Name section changes to display the word "Name" followed by a text box containing the name of the selected cell. With multiple cells selected, the Cell Name section shows as a gray bar with the words "Multiple cells selected."

Below the Cell Name section are two rows of buttons. The top row is called the Cell Attributes section. These buttons allow you to set specific attributes for selected cells. The bottom row is called the Cell Commands section. These buttons provide shortcuts to choosing Informed Designer's cell

settings commands. The Cell Attributes and Cell Commands sections are described in detail in “Cell Attributes” and “Cell Commands” later in this chapter.

The Cell palette also contains a control near the right edge of the Cell Name section. Clicking this control shows and hides the Cell List section, a scrolling list appended to the bottom of the Cell palette. See “Using the Cell List.”

## Activating the Cell Palette

Until you draw a new cell or select an existing cell, the Cell palette is inactive. That is, the buttons are unavailable, the Cell Name section doesn’t display a cell name, and no cells are selected in the Cell List section.

To make the buttons available, simply draw a new cell or select an existing cell on the form.

To activate the Cell Name section, you can:

- select a cell on the form and press F2 (Windows) or Command-Tab (Mac OS)
- select the ‘Name’ text box
- click a cell in the Cell List section and press Tab

To move off of the Cell palette and back to the form, press Enter (Windows) or Return (Mac OS), or simply click on an empty part of the drawing area, or on an object that does not contain any cells.

## Changing Cell Names

You can use the Cell palette to quickly change the name of any cell on your form. To do this, the Cell Name section on the Cell palette must be active.

To rename a single cell, select it, then press F2 (Windows) or Command-Tab (Mac OS). The ‘Name’ text box is highlighted. You can also activate the Cell Name section by selecting the text box. Type the new name in the text box and press Enter (Windows) or Return (Mac OS). Pressing Enter/Return leaves the Cell palette and returns to the form. The new cell name appears in the cell on the form and, if the Cell List section is visible, you’ll also see the new cell name updated and selected in the scrolling list.

### Note

If you haven’t already specified the title of a cell, changing the cell name also changes the title. However, once you manually change the title of a cell, changing the cell name leaves the title untouched.

You can move through the tab order of your form and rename several cells in succession by pressing F2 (Windows) or Command-Tab (Mac OS) after typing a new cell name. Pressing F2/Command-Tab selects the next cell in the tab order without leaving the Cell palette. For example, if you

select the cell with tab position 2 and change its name to 'Sold To,' pressing F2/Command-Tab updates the cell name and then automatically selects the cell with tab position 3.

## Using the Cell List

The Cell List section lists all cells on the form template, sorted by name. Clicking a cell's name in the list selects the cell (in the Cell List only), and deselects any selected objects on the form. Double-clicking a cell name in the list also selects the cell on the form and briefly displays a red arrow to indicate the cell's position. When visible, the Cell List section can be resized vertically by clicking and dragging the resize box at the bottom-right corner.

When the Cell List section is visible, you can select a cell in the scrolling list by using the mouse, by pressing the Up and Down Arrow keys, or by typing the first few characters of the cell's name. After selecting a cell, you can click any of the buttons on the Cell palette to set attributes or invoke commands.

You can use the Tab key to toggle between the Cell Name and Cell List sections. A highlighted text box indicates that the Cell Name section is active. A highlighted border around the scrolling list shows that the Cell List is the active section.

## Cell Attributes

The top row of buttons on the Cell palette is called the Cell Attribute section. These buttons allow you to set specific attributes for selected cells on your form without using commands or dialog boxes. These attributes include certain cell types, the checkbox style, and various data entry options.

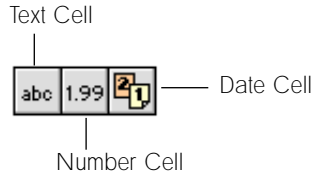
To use the Cell Attribute buttons, select one or more cells on your form, then click the appropriate button. The chosen attribute will be applied to all of the selected cells. The Cell Attribute buttons do not toggle on and off when you click them. To change an attribute that was set with the Cell attribute buttons, you must choose the appropriate command from the Settings menu (or click the corresponding Cell Commands button) and change the attribute using a dialog box. For example, if you click the 'Entry is Required' button to set a cell's data entry status, you must choose **Cell...** from the Settings menu (or click the Cell button) and change the entry status on the Cell Settings dialog box.

The following sections describe each button and its function.

### Cell Type Attributes

The Text Cell, Number Cell, and Date Cell buttons are used to set a cell's type attribute. A cell's type should match the information that it holds. For example, if a cell holds date information, its type should be date.





Clicking the Text Cell button sets the selected cell's type to Text with no special formatting (that is, no Case or Entry options). You can use this attribute for cells that hold textual information such as a comment or a memo.

Use the Number Cell button to set a cell's type to number with the following format: “#,##0.00.” With this format, the number 1500 is displayed as 1,500.00.

Clicking the Date Cell button makes the selected cell a date cell with the following format: “M/DD/YY.” Using this format, the date 07/26/96 is formatted as 7/26/96.

The Text, Number, and Date types are only three of the nine cell types that Informed supports. See “Cell Types” earlier in this chapter for detailed information about other cell types and formatting options.

## Checkbox Style Attribute

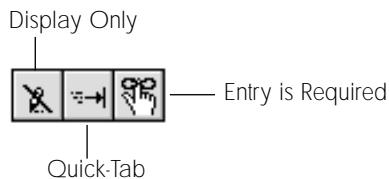


Use the Checkbox button to display a field or column cell as a checkbox with the simple ‘X’ checkbox style. For more information on checkboxes, see “Checkboxes” in Chapter 7 of your *Informed Designer Design and Graphics* manual.

## Data Entry Attributes

The Display Only, Quick-Tab, and Entry is Required buttons are used to set data entry options for selected cells. Data entry options allow you to determine how information is entered when the Informed Filler users fills out a form. By selecting a cell and clicking a button, you automatically set the corresponding data entry attribute for that cell.

The following figure shows the attribute associated with each button.

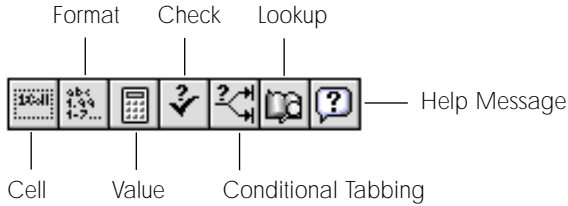


For a detailed description of the Display Only and Entry is Required attributes, see “Entry Options.” For information about Quick-Tabs, see “Quick-Tabs.”

## Cell Commands

The bottom row of buttons on the Cell palette are shortcuts to Informed Designer's cell settings commands. When you select a cell and click one of the buttons, Informed Designer displays the corresponding dialog box, allowing you to change the settings for the selected cell. For example, if you select a cell and then click the Cell button, Informed Designer displays the Cell Settings dialog box.

The following figure shows the cell settings command associated with each button.

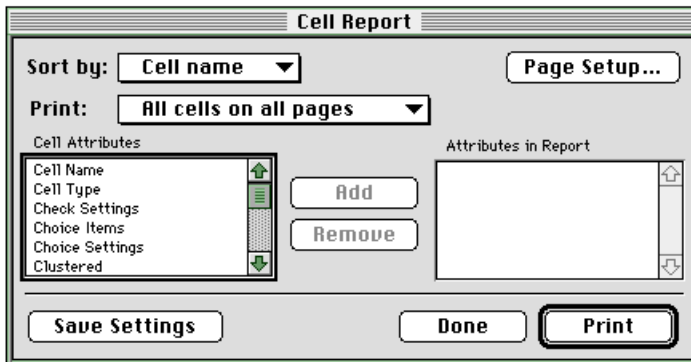


Informed Designer's cell settings commands can be found under the Settings menu. Information on all these commands is available in earlier sections of this chapter.

## Cell Report

Previous sections in this chapter explain how forms can be configured to automatically calculate, check, and look up information. Commands exist that allow you to specify cell help messages, display formats, and choice lists.

For large forms, the configuration of cells can be complex. Making even simple changes to a form can require first understanding how the different cells are related and depend on each other. The Cell Report command allows you to print a list of cells and their attributes. The report can be customized for your specific needs. Choosing the Cell Report command from the Form menu displays the Cell Report dialog box.



You customize the cell report by selecting which cell attributes to include as well as options that determine which cells to print and their order. You can print all cells, only those cells on the current page, or only the currently selected cells. Choose your setting from the 'Print' drop-down list. You can sort the report by either cell name or tab position by selecting an option from the 'Sort by' drop-down list.

The scrolling list labeled 'Cell Attributes' contains the different cell attributes that can appear on the cell report. To include an attribute in the report, select its name in the left list, then click 'Add.' The attribute name will appear in the 'Attributes in report' list. The attributes will be printed in columnar format in the order that they appear in this list. To remove an attribute from the report, select its name in the right list, then click 'Remove.'

You set the page setup of the cell report by clicking 'Page Setup' on the Cell Report dialog box. The standard Page Setup dialog box for the currently chosen printer will appear allowing you to change the page size and orientation, or select any printer specific options.

**Note**

Changing the page setup for the cell report does not affect the page setup for the form.

To print the cell report, click 'Print' on the Cell Report dialog box. The standard Print Job dialog box for the currently chosen printer will appear, allowing you to specify the number copies and other printing options.

Clicking 'Print' or 'OK' on the Print Job dialog box will begin printing the report. To cancel printing, click 'Cancel' instead.

To avoid having to re-configure the cell report each time you want to print a report, Informed Designer lets you save the settings on the Cell Report dialog box. Once you've selected the desired report options, click the 'Save Settings' button. The next time you choose the Cell Report command for the current document, Informed Designer will automatically restore the settings that were last saved.

When you've finished printing or configuring the cell report, click 'Done' to dismiss the Cell Report dialog box.

## Testing Your Form Template

The previous sections in this chapter describe Informed Designer's data intelligence commands. These commands are used to add intelligent features to your form templates.

Although you require Informed Filler to store, retrieve, and manipulate completed forms, Informed Designer allows you to *test* the intelligent features of your form during the design process. Informed Designer's Test mode simulates filling out a single form with Informed Filler. You can test calculations, formatting options, and other intelligent features of your form. You can fill in cell values to see the effect of different font and type style settings.

This section describes the commands available to you while testing a form. Informed Designer's data intelligence commands are not explained here. See the previous sections in this chapter for information about these commands and other data-related features.

To test your form, choose the Test command from the Form menu. Informed Designer hides any palettes, and disables any design related commands. To switch back to design mode, choose the Test command again.

## Entering Information

Switching to Test mode is like requesting a new record with Informed Filler. Default cell values are filled in for you and the cell with tab position '1' is selected.

Part #	Description	Qty	Price	Line Total
				<b>TOTAL</b>

If you've previously tested your form, information that you entered in cells will still be available. You can clear this information by choosing **Clear Record** from the Form menu.

You fill out a form by typing or pasting information into each cell. You can move from one cell to the next by pressing the Tab key or by clicking a different cell with the pointer. When you press Tab, Informed Designer will move you to the next cell in tab position order. If you hold down the Shift key while pressing Tab, the previous cell is selected instead. If you press F2 (Windows) or Command-Tab (Mac OS), you'll move directly to the next Quick-Tab cell. If you've configured any conditional tabs, you'll see their effect while testing your form.

When you move from one cell to another, Informed Designer checks the information you entered by evaluating the cell's check formula (if it contains one). If the check formula fails (returns a False result), Informed Designer will sound a beep and select the incorrect value. If the cell has no check formula, any entry is accepted.

## Inserting Files

While testing a form template, another way to fill in parts of a form is to use the Insert File command. This command allows you to import a text file into a text cell, or a picture into a picture cell. For the command to be available, the current cell must be a text or picture cell, and its 'Display only' option must be turned off.

To insert a file, select the cell, then choose **Insert File...** from the Form menu. For picture cells only, pressing the Enter (Windows) or Return (Mac OS) key is a shortcut for selecting the Insert File command. The standard Open dialog appears, allowing you to select a file. For text cells, the selected text file will be inserted into the cell at the current insertion point. For picture cells, the selected picture replaces the current picture in the cell.

## Entering Checkboxes

When you move to a checkbox cell, the frame flashes, indicating that the cell is selected. To change the value of a checkbox (from checked to unchecked, or vice versa), press any key or click the cell with the pointer.



If you check a checkbox that's clustered with other checkbox cells, Informed Designer will automatically turn off all other checkboxes that are part of the same cluster.

## Calculated Cells

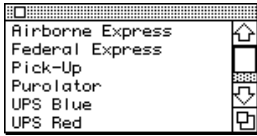
Informed automatically fills in calculated cells. A cell's calculation formula is evaluated whenever the value of any cell that participates in the formula changes.

If a cell's display only option is turned on, Informed Designer won't allow you to change the cell's calculated value. The cell will be excluded from the tab order and when you try to type in the cell, you'll hear a beep. If the display only option is turned off, you can change the cell's calculated value by selecting the cell and typing a different value.

Once you've typed to override the value of a calculated cell, you can easily change the value back to the cell's calculated value. Simply type a different value into one of the cells that participates in the calculation formula of the calculated cell—this triggers the calculation again—then re-type the appropriate value into that cell.

## Choices

For cells that have choices, you can enter information by selecting an entry from a choices palette or a drop-down list. For more information, see "Choices" earlier in this chapter. If you've chosen the 'Floating palette' and 'Auto display' options for a cell, Informed will automatically show and hide the Choices palette when you move to and from the cell.



You can also show the Choices palette by choosing **Show Choices** from the Form menu. When the Choices palette is displayed, a checkmark appears next to the Show Choices command. Choose the command again to hide the Choices palette.

## Lookups

In design mode, you can link your form with other information systems using the Lookup command. For example, you could have Informed Filler automatically ‘look up’ the description and price of a part in an inventory database when you type a part number on your invoice.

Only certain types of lookups function while you test a form. For a detailed explanation of Informed’s lookup capabilities, see “Lookups” earlier in this chapter.

## Signing Forms

You cannot test signing forms and verifying signatures with Informed Designer. However, while in Test mode you can see what a signed signature cell will look like. For more information, see “Testing Signature Cells” in Chapter 2.

## Buttons

Although you can click buttons while testing a form, doing so will not test their action. Buttons can only be tested using Informed Filler.

## Auto-incrementing Cells

When you choose the Test command to switch to Test mode, Informed Designer does not automatically enter the next available values for any auto-incrementing cells. To test an auto-incrementing cell while in Test mode, select the cell, then choose **Assign Next Value** from the Form menu.

## Changing the View Scale

When you fill out a form with Informed Filler, you can choose a view scale between 50%, 100%, and 200%. Reducing the view scale allows you to see more of the form in the form window. Enlarging the view scale can improve the readability of small type. While you test your form with Informed Designer, you can change the view scale by selecting a scale from the 'View' drop-down list at the bottom-left corner of the window.







## Using Digital Signatures

In this chapter:

- How Signatures Work 2-2
- Signing Plug-ins 2-3
- Signing With Informed Filler 2-3
- Signature Cells 2-4
- Important Precautions 2-7



## Using Digital Signatures

Approval is often a necessary step in the processing of a form. Traditionally, forms have been approved on paper by signing them with a pen. Today, technology allows us to sign forms electronically with digital signatures. Signing electronically reduces the need to print forms and, in some ways, offers more security than paper signatures.

You configure a template for signing by drawing one or more signature cells on the form using Informed Designer. Each signature cell can sign the entire form or only certain parts of the form. With a properly configured form template, Informed Filler users can easily sign completed forms and check the validity of signatures.

Informed supports the use of digital signature technology both for signing completed forms with Informed Filler, and for authorizing templates with Informed Designer. Authorizing a template adds an additional level of security to forms that are filled out and signed electronically. This security feature is described in detail in Chapter 7, “Authorizing Form Templates.”

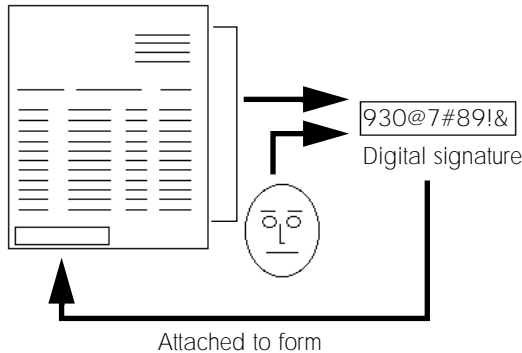
In this chapter, you’ll learn how digital signatures work and the steps necessary to configure a form template for electronic signing.

### How Signatures Work

Digital signature technology employs sophisticated encryption algorithms to provide reliable signer identification and fail safe tamper detection. This means that once someone has signed data electronically, the resulting digital signature can be used to:

- verify the identity of the person who signed the data
- detect whether or not the data has changed since it was signed

A digital signature is like a special number that’s derived from information about the person signing and the data being signed. This number can reliably identify the signer and detect any changes in the signed data. While the digital signature is stored with the signed data, the data itself is not altered in any way.



Once a digital signature is created, you can easily verify its validity. The verification process involves re-creating parts of the digital signature using current data, then comparing the results with the original signature. If they are not equal, then either the signed data or the digital signature itself has been changed or tampered with.

Before users can sign electronically, it is usually necessary that they obtain special files that act as their “electronic identity” for signing purposes. These files are often controlled and distributed by an organization’s security or administrative manager.

## Signing Plug-ins

Rather than having a single, built-in signing service, Informed relies on the digital signature services available on your computer. Nortel’s Entrust and Apple’s DigiSign are examples of such services.

Informed Designer and Informed Filler gain access to digital signature services through Informed signing plug-ins. A signing plug-in interacts directly with the available digital signature services and insulates Informed from the complexities of different signing technologies. That way we can introduce support for new or different signing technologies by simply developing new signing plug-ins. Like any Informed plug-in, signing plug-ins must be placed in your Informed plug-ins folder.

## Signing With Informed Filler

Most signing services provide ways of signing files. Signing this way signs an entire file, regardless of its content. Signing forms with Informed Filler offers significant advantages over signing the entire file. Informed Filler users can sign parts of forms or entire forms, and the way they sign forms is more similar to the way paper forms are signed.

When a form is signed with Informed Filler, the user is actually signing a single record. Each form filled out with Informed Filler is stored as a single record in a data document. Each digital signature

applies to some or all of the data for a single record, even if the data document contains multiple records. This means that the Informed Filler user can sign one form (or record), then change other records in the data document without affecting the digital signature on the signed form. To prevent the user from unintentionally changing signed data, Informed Filler locks cells once their contents have been signed.

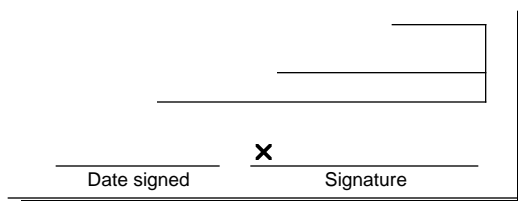
Informed Filler provides other features that make using digital signatures easy and efficient. They include the ability to:

- see exactly which cells a particular signature cell signs
- sign multiple records at the same time
- verify multiple digital signatures at the same time
- automatically verify digital signatures so that you don't have to.

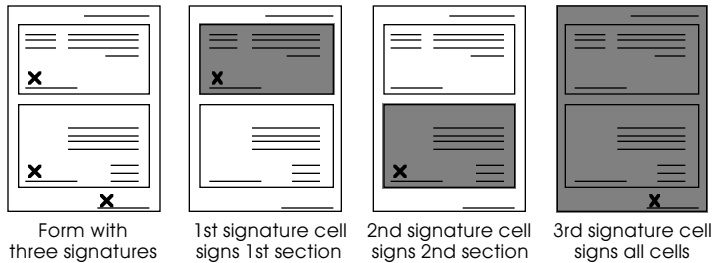
For a complete explanation of Informed Filler's digital signature features, please see Chapter 4, "Using Digital Signatures" in your *Informed Filler User's Manual*.

## Signature Cells

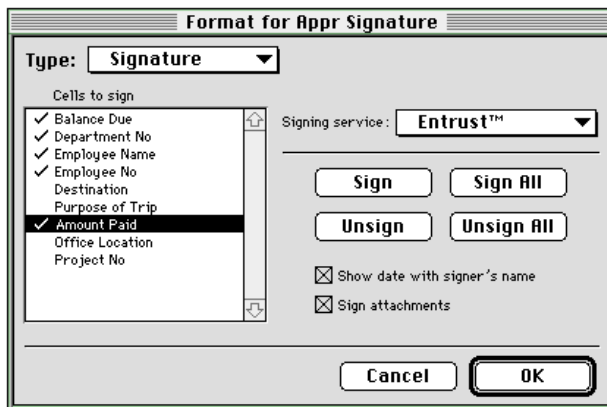
A signature cell is a cell that can store a digital signature. With Informed Designer, you draw a signature cell the same way you draw any other field cell—using the Field tool. With the flexibility that the Field tool offers, you can make a signature cell look just like the space for a signature on any paper form. For a complete explanation of Informed Designer's Field tool, see Chapter 6 of your *Informed Designer Design and Graphics* manual.



You can draw more than one signature cell on a single form. Each cell can be configured to sign different information. For example, a form that has two sections which are often filled out by two different people could have two signature cells. Each cell would sign only those cells in its respective section. You could even have a third signature cell that signs the entire form including the other two signatures.



You use Informed Designer's Format command to indicate which cells a signature cell signs. With a signature cell selected, choose **Format...** from the Settings menu to display the Format dialog box.



**Note** You cannot configure multiple signature cells at the same time. With two or more cells selected, the 'Signature' choice in the 'Type:' drop-down list on the Format dialog box is not available.

The scrolling list contains all cells on the form template with the exception of the signature cell itself. To indicate that a cell is to be signed, select its name in the list, then click 'Sign.' Click 'Unsign' to exclude a cell from being signed. As a shortcut, you can double-click a cell in the list to toggle its setting between signed and unsigned. To include or exclude all cells on the form, click either 'Sign All' or 'Unsign All.'

When configuring a signature cell, other signature cells appear in the scrolling list on the Format dialog box. A signature cell, therefore, can sign other signature cells. Signing a signature cell, however, does not automatically sign the cells that the other signature cell signs. For example, suppose that cell A signs five cells and cell B signs cell A. Signing the form in cell B signs cell A only, and not the five cells that cell A signs.

## The Signing Service

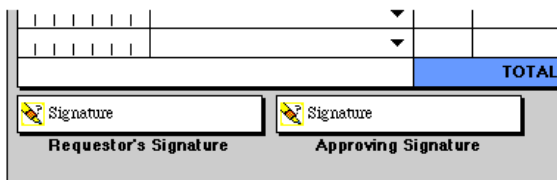
The ‘Signing service’ drop-down list contains the names of the signing plug-ins in your plug-ins folder. Normally you’ll see only one signing plug-in, such as Entrust, that corresponds to the digital signature technology used in your organization.

The ‘Signing service’ drop-down list also contains a ‘User’s choice’ item. When configuring a signature cell, you can specify that a particular signing service be used or you can allow the Informed Filler user to choose the service. If you select the ‘User’s choice’ option, the user will be asked to select which signing service to use when signing forms. This will happen only if there are two or more signing plug-ins installed.

Once you’ve indicated which cells are to be signed, and selected the correct signing service, click ‘OK’ on the Format dialog box to save the configuration. To cancel the Format command, click ‘Cancel’ instead.

## Display Options

With Informed Filler, the presence of a digital signature is indicated by a signature icon in the signature cell. The name of the person who signed the form appears to the right of the signature icon in the cell’s font, font size, style, and alignment. When testing a template, you’ll see the word “signature” instead.



Informed Filler can also display the date on which a record was signed in the signature cell. If you select the ‘Show date with signer’s name’ option, the date of signing will appear to the right of the signer’s name.

Like any cell, you can change the font, size, style, and alignment of a signature cell using the Type command or the Style submenus. When selecting the cell’s type settings, be sure that the name (and date, if selected) will fit in the dimensions of the cell. For more information about type settings, please see Chapter 7 of your *Informed Designer Design and Graphics* manual.

## Signing Attachments

Chapter 5, “Attachments,” in your *Informed Filler User’s Manual* explains how the Informed Filler user can attach files to records. This feature is useful if the user wants to attach associated files to completed forms. For example, an engineering organization may use an engineering change request form to authorize design changes. If the design documents were also stored electronically using,

say, a CAD program, the user could attach the relevant design documents to the corresponding engineering change request form. Informed Filler stores the attached file along with the record's cell data in a data document.

Like cell data, attached files can also be signed by a signature cell. By doing so, a signature can protect the integrity of attached files in addition to the information on the form itself. To configure a signature cell to sign attachments, select the 'Sign attachments' option on the Format dialog box.

## Testing Signature Cells

Informed Designer's Test mode allows you to fill out a single record to test your calculations, formatting, and other intelligent features.

When testing a form, you can select a signature cell like any other cell: by tabbing to the cell, or by clicking the cell with the mouse. When selected, a bold outline frames the cell's interior, and all cells that are signed by that cell are framed with a red box. You cannot, however, test signing a record or verifying a signature using Informed Designer's Test mode.

## Important Precautions

Care must be taken when planning the use of digital signatures. Once users have filled out and signed forms, changes to the template that might affect the signed data must be avoided. Otherwise you risk unintentionally invalidating existing digital signatures.

Changes that can invalidate existing digital signatures include:

- changing the cell type of a cell that contains signed data
- deleting a cell that contains signed data
- changing the configuration of a signature cell (changing which cells it signs).

If you need to change a form template (in the ways described above) that is associated with previously filled out and signed records, you should make the change in a new copy of the template to avoid affecting the existing data and digital signatures. This means that users would have two templates, one containing the unchanged design for use with previously completed and signed forms, and the other containing the revised design with which new records could be entered. This leaves the data of the previously completed forms unchanged, therefore preserving the integrity of any existing digital signatures.

For more information about revising form templates, see Chapter 8, "Form Template Distribution and Revision."





# 3 Customizing Menus

In this chapter:

- The Menu Bar 3-2
- Configuring a Menu 3-3
- Menu Item Types 3-4
- Help Menus 3-11
- Printing the Menu Configuration 3-12

## 3 Customizing Menus

The flexible nature of Informed’s features makes it suitable for designing forms that automate both general as well as very specific processes. To help make a form more “custom suited” to a particular purpose, Informed Designer allows you to customize the menus that are presented to the Informed Filler user. By customizing menus you can:

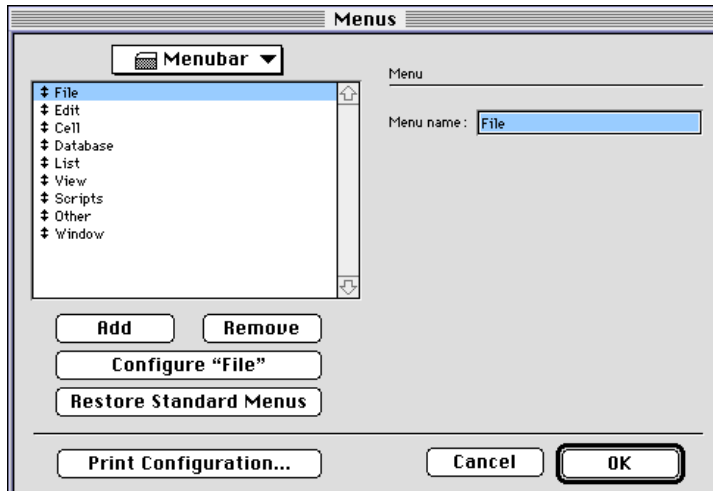
- remove menus and commands that are not important or relevant to your form template
- use terminology that is specific to the process for which a template is designed
- add commands that perform custom tasks

Customizing menus can provide a simpler and more familiar environment for the form filling user. This can result in higher productivity and fewer errors. In this chapter you’ll learn how to use Informed Designer’s Menu command to configure the menu bar, the individual menus, and the items that are contained in each menu. You’ll also learn about the different menu item types and their uses.

### The Menu Bar

The menu bar consists of one or more—usually several—menus aligned horizontally from left to right. Each menu contains one or more menu items, usually grouped by function. Each new template created with Informed Designer is automatically assigned a standard menu configuration.

To customize the menu bar, choose **Menus...** from the Configure submenu under Form. The Menu dialog box appears.



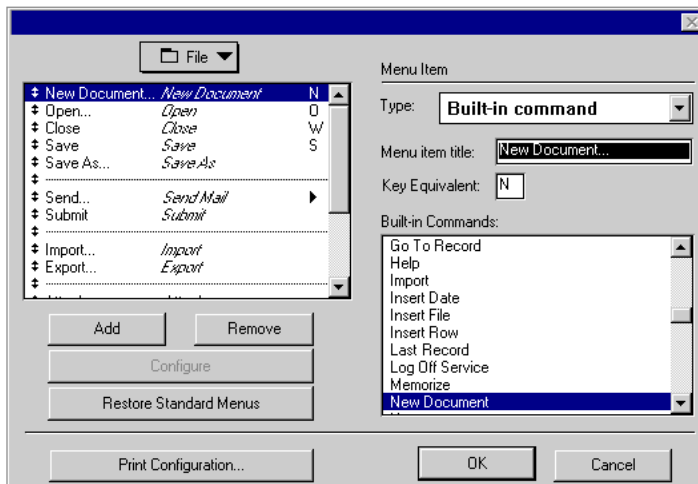
Initially, the scrolling list shows the menus in the menu bar. The top-to-bottom ordering of the menus represents the left-to-right positioning of each one in the menu bar. You can change the position of a menu by selecting it in the list and dragging it up or down. You can change a menu's name by selecting it in the list and typing its new name in the 'Menu name' text box.

To add a new menu, click 'Add.' The new menu is added immediately below the currently selected menu, or at the top of the list if no menu is selected. To remove a menu, select it in the list, then click 'Remove.'

At any time while configuring menus, you can revert to the standard Informed Filler menus by clicking the 'Restore Standard Menus' button.

## Configuring a Menu

Configuring a menu involves adding and removing menu items and specifying the action that each item performs. To configure a menu, first select it in the scrolling list, then click 'Configure,' or simply double-click the menu name in the list. The scrolling list changes to display the menu items for the selected menu, and the drop-down list above shows the title of that menu. To change the list back to the menu bar, select 'Menu bar' from the drop-down list.



Like menus in the menu bar, you can change the position of a menu item by selecting the item in the list and dragging it up or down. To add a new menu item, click 'Add.' A new menu item is added immediately below the currently selected item, or at the top of the list if no menu item is selected. To remove a menu item, select it in the list, and click 'Remove.'

### Note

A menu with no menu items will not appear to the Informed Filler user.

## Menu Item Types

Each menu item is of a particular type. Menu item types include:

- Built-in command
- Script
- Plug-in command
- Submenu
- Document names
- Script names
- Record tags
- Record list formats
- Plug-in command names
- Font
- Size
- Type style
- Separator line

To configure a menu item's type, select the menu item in the scrolling list and select a type from the 'Type' drop-down list. Menu items that are built-in commands, scripts, or plug-in commands can be renamed by typing a new name in the 'Item title' text box.

You can also assign key equivalents to any built-in command, script, or plug-in command. Simply type the key in the 'Key equivalent' text box. A key equivalent is invoked by pressing the assigned key while holding down the Ctrl key (Windows) or the Command key (Mac OS).

### Note

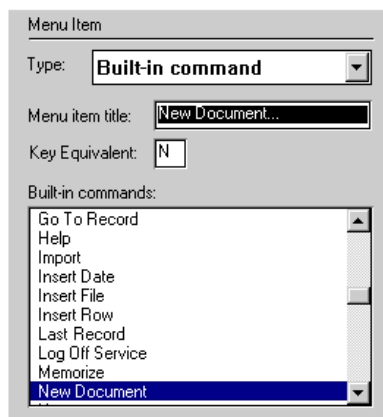
The letters "X," "U," "I," "C," and "P" are reserved for standard Windows commands (EXit, Undo, Cut, Copy, and Paste).



In addition to an assigned key equivalent, the Informed Filler user can choose a menu item using the standard Windows keyboard interface. This involves pressing the Alt key, then typing the letter that appears underlined in the menu or menu item's title. Although Informed Filler will automatically decide which letters are underlined in the menu and menu item titles, you can specify which letter yourself by preceding it with the ampersand character (&). For example, to underline the "D" in "New Document...", type the title as "New &Document..."

## Built-in Commands

Built-in commands correspond to the commands and settings that are built into Informed Filler. They include the commands and settings that are associated with the menu items of Informed Filler’s standard menu configuration. All available built-in commands are listed in the ‘Built-in Commands:’ scrolling list. To select a built-in command, simply click its name in the list.



Certain built-in commands can not have custom menu item titles. They include: Show Attachments, Show Record List, and Show Signed Cells. This is because Informed Filler automatically changes the command name depending on context. For example, if the Attachments window is active, the Show Attachments command changes to Hide Attachments.



Both Windows and Mac OS have standard conventions for naming the command that quits an application. On Windows the command is Exit. On Mac OS the command is Quit. Like commands such as Show Attachments, you cannot change the name of this command. Informed Filler will automatically use the correct command name according to the platform used.



The Mac OS command “Show Clipboard” is not a standard command for Windows applications. A menu item that is configured to invoke this built-in command will not appear in Informed Filler on Windows.

For a brief description of the built-in commands, see Appendix B.

## Scripts

A Script menu item is an item that invokes an attached AppleScript script. An AppleScript script is attached to a template using the Scripts command in the Configure submenu of Informed Designer’s Form menu. For more information, see Chapter 12, “Using AppleScript.”

Menu Item

Type: **Script**

Menu item title:

Key Equivalent:

Scripts:

- Submit Batch
- Create Packing Slip

**Note** You can also configure a menu to include a menu item for each script that is attached to the template without having to configuring each individual item. See “Script Names” for more information.



Since AppleScript is a Mac OS scripting system, AppleScript scripts do not function on computers running Windows. For Windows users, Informed Filler does not show any menu items that are configured to invoke AppleScript scripts.



If AppleScript is not active, Informed Filler will show the menu item, but it will be unavailable.

## Plug-in Commands

Some of Informed Filler’s features are made available by installing Informed plug-ins. Certain plug-ins have commands associated with them. For example, the Receive From Newton command will be displayed if the Newton Filler plug-in is installed. In order to configure a menu item to invoke a plug-in command, you must have the plug-in installed in your plug-ins folder. When you select the Plug-in Command menu item type, the scrolling list below lists all commands, if any, associated with the plug-ins currently installed in your plug-ins folder. Select the command that you want to invoke by clicking it in the list.

Menu Item

Type: **Plug-in command** ▼

Menu item title:

Key Equivalent:

Plug-in Commands:

- Receive From Newton...

If the Informed Filler user does not have the plug-in to which a menu item corresponds installed, the menu item will still appear; however, it will be unavailable.

**Note** You can also configure a menu to automatically include a menu item for each plug-in command so that you do not have to configure each command individually. See “Plug-in Command Names” for more information.

## Submenus

Submenus (also called *hierarchical* menus or *cascading* menus) are menus that are accessed through a single menu item of a parent menu. Submenus are most often used for settings like Font, Type Style, and Size.

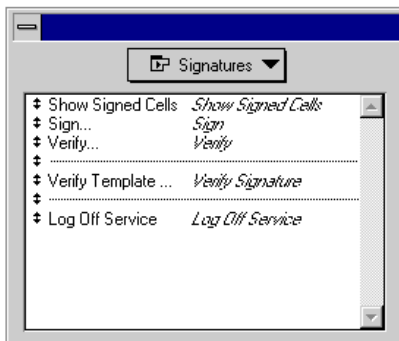
You name a submenu by selecting it in the scrolling list and typing its name in the highlighted text box.

Menu Item

Type: **Submenu** ▼

Menu item title: **Signatures**

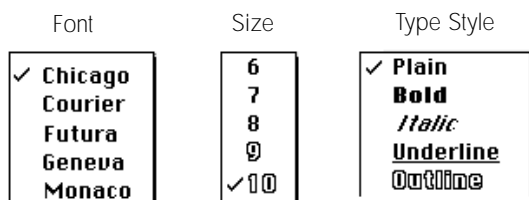
To configure the menu items of a submenu, select the submenu in the list and click ‘Configure,’ or simply double-click the submenu in list. The left list changes to show the submenu’s menu items, and the submenu’s title appears above the list.



By using the various controls on the Menus dialog box, the menu items in a submenu can be configured just like those in a regular menu. To reveal the items for a different menu or submenu, use the drop-down list to display the desired menu in the scrolling list, then double-click it.

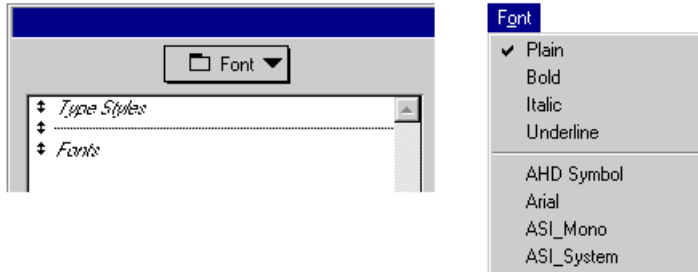
## Font, Size, and Type Style

The Font, Size, and Type Style menu item types allow you to include the menu items that they represent in your menu configuration. Rather than configuring individual menu items for each of the different fonts, sizes, and type styles (especially since each Informed Filler user can have different fonts in their system), these menu items act as placeholders for the respective groups of menu items.



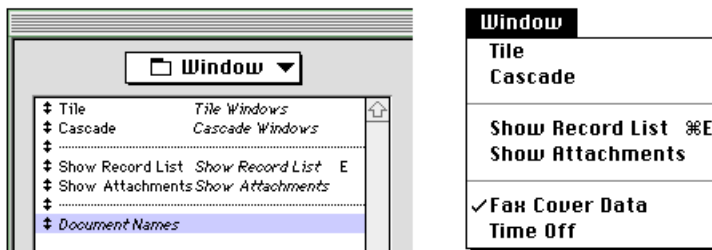
Although these menu item types appear as single menu items on the Menus dialog box, the Informed Filler user will see the individual menu items that they represent. Below is an example configuration that places the Type Style and Font menu items in a menu titled “Font” with a separator line between them.





## Document Names

Most applications allow you to open more than one document at once and select from among them by choosing their names from a menu, often titled “Windows.” As you open and close documents, the Windows menu changes to reflect those that are currently open. The Document Names menu item type represents the names of documents currently open in Informed Filler.



## Script Names

As explained in Chapter 12, “Using AppleScript,” you can attach AppleScript scripts to templates so that they’re accessible to the Informed Filler user. “Scripts,” earlier in this chapter, explains how you can configure a menu item to invoke a particular script.

The Script Names menu item type allows you to include the names of all attached scripts in your menu configuration without having to configure each script individually. Informed Filler’s standard menu configuration includes the script names as the only items under the menu title “Scripts.”



**Note** This placeholder will only show scripts not specifically configured elsewhere.

## Record Tags

Chapter 8 of your *Informed Filler User's Manual* explains how the Informed Filler user can tag a collection of records so that the records can be easily retrieved at a later time. Each record tag is named by the Informed Filler user.

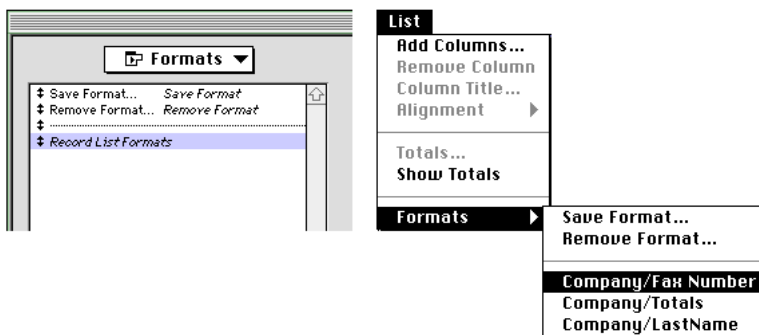
The Record Tags menu item type represents the names of the record tags that the Informed Filler user has defined for the active document. The Informed Filler user restores the collection of records to a set of tagged records by choosing the named record tag from a menu. The standard Informed Filler menus include a menu item of this type in the Tags submenu under the Database menu.



## Record List Formats

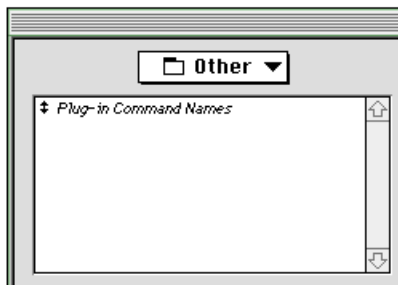
Chapter 8 of your *Informed Filler User's Manual* describes the Record List and how it can be used to display multiple records in a list. The Informed Filler user can customize and name multiple different formats for the Record List. For example, one record list format might show detailed information whereas another might show only summary information.

The Record List Formats menu item type represents the names of the Record List formats that the Informed Filler user has defined. The standard Informed Filler menus include a menu item of this type in the Formats submenu under the List menu.



## Plug-in Command Names

As explained earlier in “Plug-in Commands,” a command that’s associated with an Informed plug-in can be included in the menu configuration of a template. The Plug-in Command Names menu item type represents the names of all plug-in commands for the Informed plug-ins installed on the Informed Filler user’s computer. The standard Informed Filler menus include a single menu item of this type in the Other menu.



The advantage of using the Plug-in Command Names menu item type is that templates do not have to be configured specifically for each different plug-in command. If you expect that different Informed Filler users will have different plug-ins installed, then using the Plug-in Command Names menu item type means that you could still have only one version of the template. Each Informed Filler user would see only those plug-in commands that are associated with the plug-ins installed on his or her computer.

If you would rather position different plug-in commands at different places in the Informed Filler menu bar, then configure each command individually using the Plug-in Commands menu item type.

**Note** This placeholder will only show plug-in commands not specifically configured elsewhere.

## Separator Line

Most menus include separator lines to group commands by type. Grouping commands improves the readability of a menu; use the Separator Line menu item type for this purpose.

## Help Menus

Informed Filler provides on-line help for both the Windows and Mac OS platforms. The commands associated with on-line help, however, are not configured as part of a template’s menu configuration.

On Windows, Informed Filler will automatically append a menu titled “Help” to the end of the menu bar. This menu contains the commands for displaying the on-line help. On the Macintosh,

these commands are available in the standard Guide menu that appears near the right side of the menu bar. For detailed information about Informed Filler's on-line help features, see your *Informed Filler Getting Started Guide*.

## Printing the Menu Configuration

At any time during the menu customization process, you can print your menu configuration. To print your menu configuration, choose **Menus...** to display the Menus dialog box, then click 'Print Configuration....' The page setup used to print the menu configuration (page size, orientation, and so on) is the same as that of the template itself.

# 4 Using Buttons

In this chapter:

- Drawing Buttons 4-2
- Configuring a Button's Action 4-3

# 4 Using Buttons

As we transition away from paper and towards electronic forms, the appearance of forms changes. Forms become more like applications that have a “custom” look-and-feel. Chapter 3, “Customizing Menus,” explains how Informed Filler’s menu commands can be customized for the specific use of a form. This chapter describes Informed Designer’s Button tool and how it can be used to draw and configure buttons to perform specific actions. Buttons can help make a form easier to use by making important or common commands and options visible on the form itself.

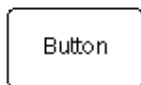
Buttons can be configured to invoke commands that are built into Informed Filler, commands that are available through Informed plug-ins, or AppleScript scripts that are attached to the form template (Mac OS only). In this chapter you’ll learn how to draw buttons, change their appearance, and configure them to perform particular actions.

## Drawing Buttons



Informed Designer’s tool palette contains the Button tool. With this tool you can draw buttons of any size. Buttons can have a normal appearance, or they can be transparent and placed over top of other objects such as pictures or icons.

You draw a button much like you do a rectangle. With the Button tool selected, position the pointer where you want a corner of the button to start, then click and drag the pointer to the opposite corner and release the mouse button.



When you draw a new button, its label is initially “Button.” You change a button’s label using Informed Designer’s Text tool. For a detailed explanation of text editing, please see Chapter 6, “Drawing Tools,” in your *Informed Designer Design and Graphics* manual.

You can customize the appearance of a button by changing its font, size, type style, alignment, pen, fill, line width, and roundness of corners. For information on setting these attributes, please see Chapter 7, “Changing an Object’s Appearance,” in your *Informed Designer Design and Graphics* manual.

In addition to the attributes listed above, you can also choose among two button styles: plain and shadowed. A shadowed button is drawn with a small shadow below and to the right of the button.



To set the style of a button, select it, then choose **Button...** from the Settings menu. When the Button Settings dialog box appears, select either 'Plain' or 'Shadowed' from the 'Style' drop-down list.



Choose a button style from this drop-down list.

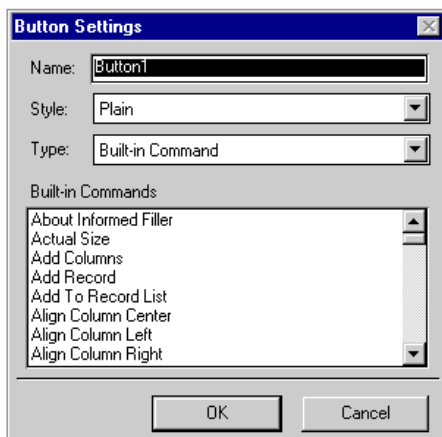
Like any type of object, buttons can be repositioned, resized, and manipulated using Informed Designer's Pointer tool and various commands. For more information see Chapter 8, "Manipulating Objects," in your *Informed Designer Design and Graphics* manual.

## Configuring a Button's Action

A button is configured to perform an action when the Informed Filler user clicks it. There are three types of actions that a button can perform:

- a command that's built into Informed Filler
- a command that's associated with an Informed plug-in
- an AppleScript script.

In addition to its action, a button also has a name. Both the name and action of a button are set using Informed Designer's Button command. Select a button, then choose **Buttons...** from the Settings menu.



#### 4-4 : Using Buttons

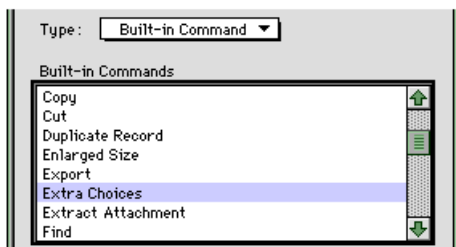
You can give the button a meaningful name regardless of its label. When you draw a new button, it is initially named “Button1,” “Button2,” and so on. To change the button’s name, type a new name in the ‘Name’ text box. Button names must be unique.

To configure a button’s action, first specify the action type by using the ‘Type’ drop-down list, then select the action from those in the scrolling list.

**Note** You cannot test a button in Informed Designer’s Test mode. A button can be clicked, but doing so will not perform its configured action.

### Built-in Commands

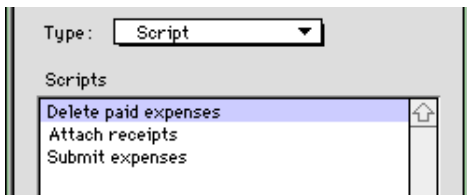
Built-in commands correspond to the commands and settings that are built into Informed Filler. They include the commands and settings that are associated with the menu items of Informed Filler’s standard menu configuration. To select a built-in command, simply click its name in the ‘Built-in commands:’ list.



For a description of the built-in commands, see Appendix B.

### Scripts

A button can be configured to invoke an attached AppleScript script. An AppleScript script is attached to a template using the Scripts command in the Configure submenu of Informed Designer’s Form menu. For more information, see Chapter 12, “Using AppleScript.”







Since AppleScript is a Mac OS scripting system, AppleScript scripts do not function on computers running Windows. For Windows users, Informed Filler will show a button that is configured to invoke an AppleScript script, but clicking it will display the message “That button cannot be used because it relies on AppleScript, a Mac OS scripting system.”



On Mac OS compatible computers, if AppleScript is not active, Informed Filler will show the button, but it will be unavailable.

## Plug-in Commands

Some of Informed Filler’s features are made available by installing Informed plug-ins. Certain plug-ins have commands associated with them. In order to configure a button to invoke a plug-in command, you must have the plug-in installed in your plug-ins folder. When you select the Plug-in Command type, the scrolling list below lists all commands, if any, associated with the plug-ins currently installed in your plug-ins folder.

If you configure a button to invoke a particular plug-in command and the Informed Filler user does not have the plug-in installed, the button will be visible, but it will be unavailable.



# 5 Routing

In this chapter:

- Suggested Routes 5-3
- Suggested Routes for Multiple Platforms 5-3
- Adding, Changing, and Removing Suggested Routes 5-4
- Controlling the Data Format 5-8
- Using Mail Cells 5-9

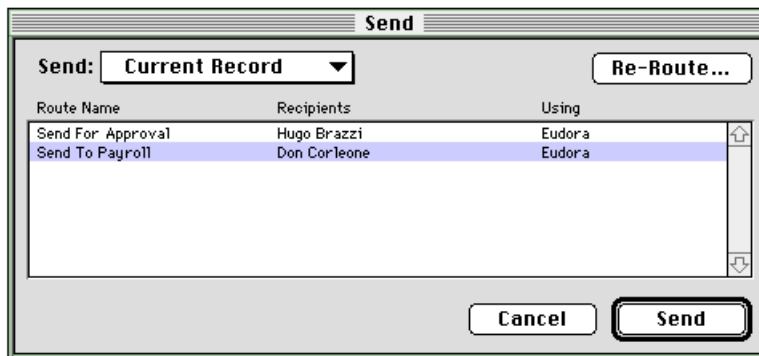
# 5 Routing

Chapter 12 of your *Informed Designer Design and Graphics* manual explains how you can use electronic mail to mail a form template to another person. This chapter explains how you can configure a template to aid the Informed Filler user in addressing and mailing completed forms.

Many types of forms must be sent from person to person for approval and processing purposes. For example, it's very common for a purchase requisition form to be filled out by the requestor, sent to a supervisor for approval, and then forwarded onto the purchasing department. In large organizations, a single form might go through several levels of approval before the process is complete.

To aid the Informed Filler user in selecting the appropriate person or place to send a form to, you can add one or more suggested routes to a template. You can also specify the data format to use when sending forms so that the Informed Filler user doesn't choose an incorrect format by mistake.

When sending a completed form, the Informed Filler user can simply select the appropriate suggested route according to his or her role in the processing of the form. The user also has the option to re-route the form to a different person or place.



Without a list of suggested routes, the user is expected to know where the form must be sent.

Like mailing templates, mailing completed forms with Informed Filler relies on Informed mail plug-ins and the associated e-mail software being installed. If you intend to specify suggested routes, complete with e-mail addresses for recipients, you need to have the appropriate Informed mail plug-in and the e-mail software installed on your computer. Mail plug-ins must be installed in your plug-ins folder.

For details regarding the e-mail systems supported by Informed, see the on-line document "DGRPLG.PDF" (Windows) or "Informed Designer Plug-ins" (Mac OS). This document is installed when you install Informed Designer and is viewed using Acrobat Reader (also included with Informed Designer).

## Suggested Routes

A suggested route consists of a route name and one or more recipients. The route name alone can provide helpful information to the Informed Filler user if it describes the step. For example, if the user has never filled out a particular type of form before, they might not know where to send it. If you give the suggested route an obvious name such as “Send to Payroll Services,” the user can quickly see where the form should be sent after it’s filled out.

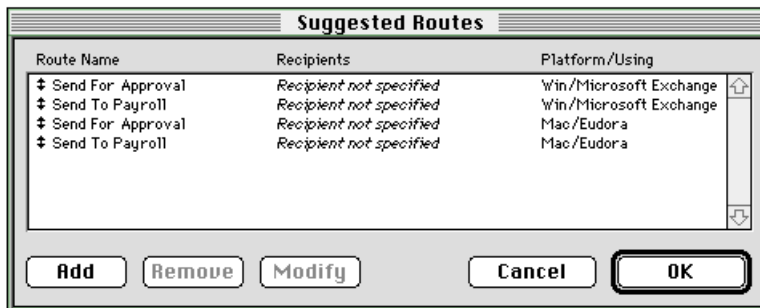
If the recipient of a suggested route is the same for all users, you can specify the recipient too, so the Informed Filler user doesn’t have to. However, it’s common that you, as the designer, cannot specify the recipient for a suggested route because the recipient is different for different people. For example, suppose five users who work in different departments of an organization have to fill out time cards for each week. Even though the suggested route for the form might be named “Send to Approving Manager,” the users would send their completed time cards to different managers. For these routes, you would leave the recipient unspecified. The Informed Filler user can specify the recipient for the suggested route the first time and, if requested, Informed Filler will remember the recipient’s name for subsequent sends.

## Suggested Routes for Multiple Platforms

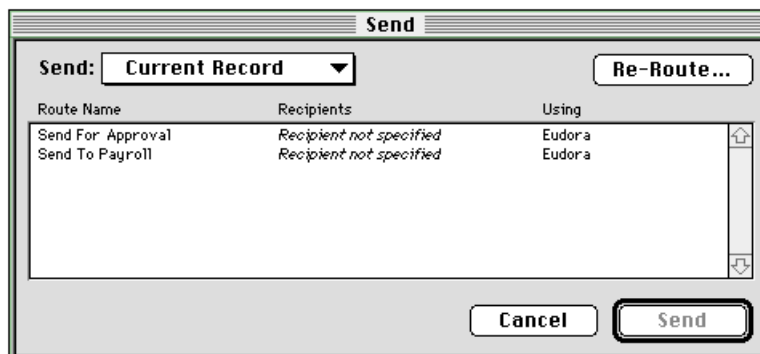
Adding a suggested route to a form template that will be used on multiple platforms requires special consideration. With Informed Designer’s Routing command, you can only define suggested routes for the e-mail systems and mail plug-ins installed on your computer. Likewise, a suggested route is available to the Informed Filler user only if he or she uses the e-mail system and platform for which the recipients of the route have been specified.

If you want to define a suggested route and make it available to Informed Filler users on both Windows and Mac OS computers, you must add the route two times using Informed Designer’s Routing command, once on a Windows computer, and once on a Mac OS computer. To do this, open the template on the platform of your choice (Windows or Mac OS), add the suggested route, and then save the template. Next, open the same template on the other platform (Mac OS or Windows) and add the route again. Adding a suggested route multiple times is also necessary if different Informed Filler users of the same platform use different e-mail systems. For detailed instructions on how to add a suggested route, see “Adding, Changing, and Removing Suggested Routes” later in this chapter.

Informed Designer’s Suggested Routes dialog box displays all suggested routes for the template, regardless of platform and e-mail system used for addressing. The platform and e-mail system are indicated under the “Platform/Using” column.



Informed Filler shows only the suggested routes that are addressed using the e-mail systems and mail plug-ins available on the user's platform.



## Adding, Changing, and Removing Suggested Routes

When you set up a suggested route for a form, you can specify details such as the route name, which e-mail system to use, who the recipient is, and which format the form will be sent in. Some of these details, such as the recipient and the data format, can be specified by the Informed Filler user when the form is mailed. By specifying these details in the suggested route whenever possible, you eliminate that task for the Informed Filler user, making it easier for the person to mail the completed form.

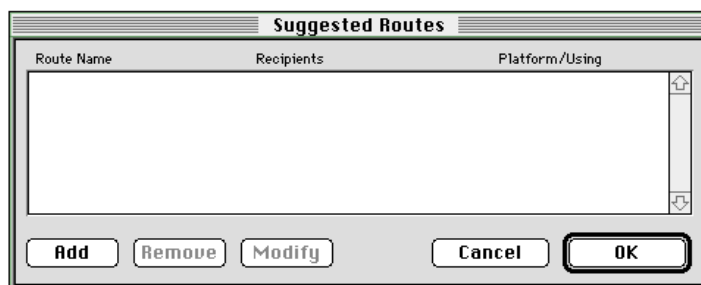
The following table shows the various parts that make up a suggested route and gives a description of each one.

#### Details of a Suggested Route

Detail	Description
route name	The name that you give to a particular step in the routing process. By making the route name descriptive of the step, you make it easy for the user to see where the form should be sent.
e-mail system	The e-mail system used to route the form. The options available correspond to the Informed mail plug-ins that you have installed in your plug-ins folder.
recipient(s)	If appropriate, you can specify the address of the individual(s) the form should be sent to. This detail is optional since you might not know the name of each person in the routing process, and the recipients may differ depending on who is sending the form. You can only edit the recipients for suggested routes that use the e-mail systems and mail plug-ins installed on your computer.
data format	As an option, you can specify the data format that the form will be sent in. The formats available are: Informed data, Informed package, Informed Interchange, Comma delimited text, and Tab delimited text. Other formats may be available through Informed plug-ins.
subject	You can also specify the subject of the mail message so that the Informed Filler user doesn't have to.

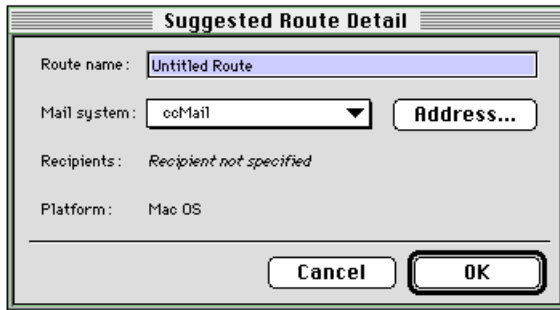
You use Informed Designer's Routing command to add, change, or remove suggested routes on your form template.

Choose **Routing...** from the Configure submenu under Form. The Suggested Route dialog box appears.



The Suggested Routes dialog box contains various controls for editing suggested routes, and a scrolling list showing all suggested routes configured for the template. If you haven't previously configured any suggested routes, the scrolling list will be blank.

When you click ‘Add,’ the Suggested Route Details dialog box appears.



The image shows a dialog box titled "Suggested Route Detail". It has a title bar with the text "Suggested Route Detail". Inside the dialog, there are several fields and buttons:

- Route name:** A text box containing "Untitled Route".
- Mail system:** A dropdown menu showing "ccMail" and an "Address..." button to its right.
- Recipients:** A text box containing "Recipient not specified".
- Platform:** A text box containing "Mac OS".
- Buttons:** "Cancel" and "OK" buttons at the bottom.

Type the name of the suggested route in the ‘Route name’ text box. As mentioned previously, it is helpful to the Informed Filler user if the route name describes the actual step in the routing process.

Specify the mail system to use by clicking the ‘Mail system’ drop-down list and making a selection from the available choices. The choices in the ‘Mail system’ drop-down list correspond to the Informed mail plug-ins you have installed in your plug-ins folder.

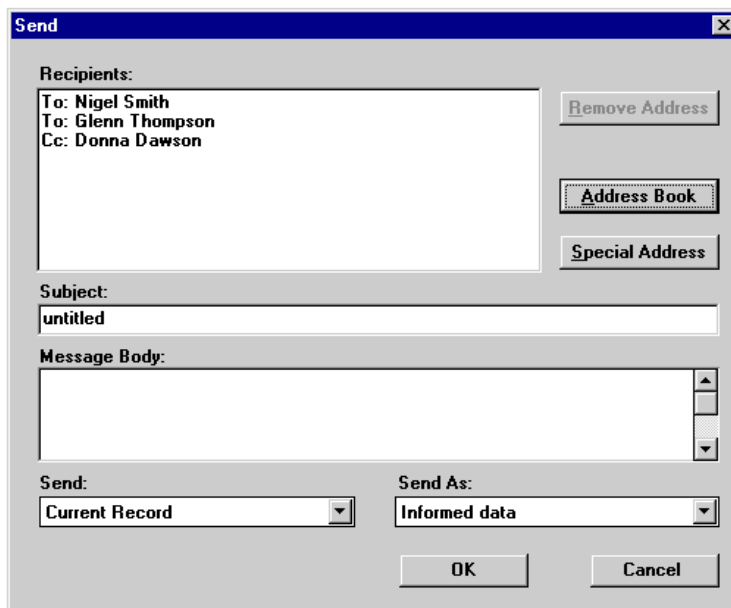
**Note**

You don’t have to be using a particular e-mail system to specify that system for the suggested route. For example, if you are using Microsoft Mail, you can still specify cc:Mail as the e-mail system for a suggested route, as long as the cc:Mail plug-in is installed in your plug-ins folder. However, in order to specify the recipients for a suggested route, you must have the e-mail system installed on your computer.

For details regarding the e-mail systems supported by Informed, see the on-line document “DGRPLG.PDF” (Windows) or “Informed Designer Plug-ins” (Mac OS). This document is installed when you install Informed Designer and is viewed using Acrobat Reader (also included with Informed Designer).

To specify a recipient, click the ‘Address...’ button. If you have the appropriate e-mail software installed, Informed Designer displays the addressing dialog box for that mail system. Select one or more recipients by using the controls on this dialog box. Below is the dialog box you’ll see when addressing a suggested route for Microsoft Exchange on Windows.





The 'Send' dialog box is shown with the following fields and buttons:

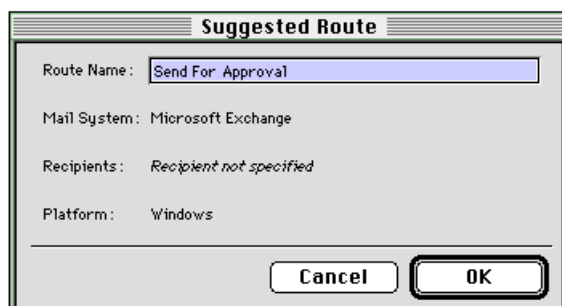
- Recipients:** A text area containing 'To: Nigel Smith', 'To: Glenn Thompson', and 'Cc: Donna Dawson'. To the right are buttons for 'Remove Address', 'Address Book', and 'Special Address'.
- Subject:** A text field containing 'untitled'.
- Message Body:** A large empty text area with vertical scroll bars.
- Send:** A dropdown menu set to 'Current Record'.
- Send As:** A dropdown menu set to 'Informed data'.
- Buttons for 'OK' and 'Cancel' at the bottom.

While the addressing dialog box is displayed, you can specify a subject for the form, and you can also select the data format that the form will be sent in.

### Note

If you've already selected a data format for mailing using the 'Always mail data as' mail preference, that data format will take precedence over the one you specify for a suggested route. For more information, see "Controlling the Data Format" later in this chapter.

To edit an existing suggested route, select it in the scrolling list on the Suggested Routes dialog box, then click the 'Modify' button. The Suggested Route Detail dialog box appears, allowing you to change the specific details for the selected route. If you try to modify a suggested route that's been configured for a platform other than the one you're working on, you are not able to change mail systems or address the route.



The 'Suggested Route' dialog box displays the following information:

- Route Name:** Send For Approval
- Mail System:** Microsoft Exchange
- Recipients:** Recipient not specified
- Platform:** Windows
- Buttons for 'Cancel' and 'OK' at the bottom.

To remove a suggested route, select it in the scrolling list, then click 'Remove.'

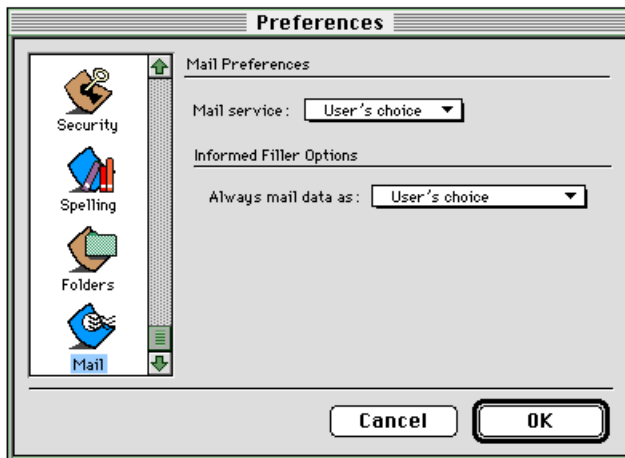
You can change the order of suggested routes in the scrolling list by clicking a route and dragging it either up or down with the mouse.

## Controlling the Data Format

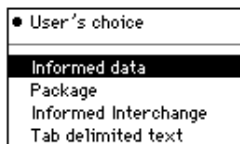
When an Informed Filler user mails a completed form, it's important that the form be sent in an appropriate data format. For example, the Informed package format contains both the form data and the form template. If a user sends a form in this format to someone who already has the template, the user is sending unnecessary information. Depending on the architecture and environment on which your Informed solution is based, it might be necessary that Informed Filler users send forms using a particular data format, Informed data, for example.

Informed Designer's 'Always mail data as' mail preference allows you to control the data format used when a completed form is mailed by the Informed Filler user.

To specify this format, choose **Preferences...** from the Edit menu and click the Mail icon in the scrolling list. The dialog box changes to show the Mail Preferences panel.



Click the 'Always mail data as' drop-down list and make a selection from the available choices.



The format you select using the 'Always mail data as' preference overrides any formats that the Informed Filler user selects when mailing, as well as any formats that you specify for a suggested route.

Selecting the 'User's choice' option allows Informed Filler users to mail completed forms in the format of their choice.

## Using Mail Cells

A mail cell is a cell on the form template whose value is used to specify a parameter when the Informed Filler user mails a form. For example, when the user mails a form, Informed Filler looks for a cell named 'Mail Send To.' If such a cell exists, the form is automatically addressed to the recipient named in that cell. By using mail cells on your template, forms can be dynamically routed to different users based on different conditions.

The following table shows the four mail cells that can be used to set send parameters when the Informed Filler user mails a form.

### Mail Cells

Mail Cell	Description
Mail Send To	The name of the recipient. The value in this cell must identify a valid user for the e-mail system used by the Informed Filler user.
Mail Subject	The subject of the mail message.
Mail Comment	The body of the mail message.
Mail Enclosure	The filename of the enclosure that is attached to the mail message.

By calculating a mail cell, you can have a formula automatically check for different conditions and automatically set certain parameters accordingly. For example, if a purchase order form needs to be sent to a supervisor for approval, you could calculate the 'Mail Send To' cell so that the recipient of the form is based on the value of the purchase order. The formula below demonstrates this calculation.

```
If POTotal > 500 Then
    Return "Lauren Quinn"
Else
    Return "Bob Johnson"
End
```

Based on this calculation, any purchase orders over 500 dollars would automatically be addressed to Lauren Quinn. All purchase orders less than or equal to 500 dollars would go to Bob Johnson.



# Form Tracking

In this chapter:

- How it Works 6-2
- Configuring Tracking 6-6
- Using the Informed Tracker Server 6-11



## Form Tracking

Many forms are routed from person to person for approval purposes. Chapter 5, “Routing,” explains how you can add one or more suggested routes to a form template so that the Informed Filler user can better choose where to send a form. This chapter describes Informed’s built-in form tracking features.

The purpose of form tracking is to provide the Informed Filler user with a way to quickly and easily find out where a form is in the routing process. After filling out a form and sending it to the next step, the user can choose a single command to see any subsequent steps that the form has travelled, and on whose desk the form currently sits. Form tracking provides a quick alternative to “physically” tracing the path of a form to find its current location.

### How it Works

When an Informed Filler user sends a form to another person, the form data is stored in a file and attached to an electronic mail message. The message, along with the attached form data, is sent to the recipient using one of the supported e-mail systems. The recipient receives the “original” form whereas the sender retains a “copy.” The recipient performs the required task, then, if necessary, sends the form to the next step in the routing process.

In order to track a form, the information detailing each step that the form travels must be stored in a central database that is accessible by all Informed Filler users. That way, any sender of a form can retrieve the details regarding the other people to whom the form has been sent.

As a form is sent from person to person, the tracking status for the form is updated. Informed Filler does this by storing information, including the sender and recipient names, the date and time that the form was sent, and attributes that identify the form, in the tracking database. When the user requests the tracking status of a form, Informed Filler connects to the tracking database and retrieves the tracking details pertaining to the particular form and displays them on a dialog box.

**Form Tracking Status**

DATESENT	RECIPIENT	PRNUMB
1996-08-19	Sandy Beech	9608068
1996-09-03	John Nelson	9608068

Details OK

Form tracking requires configuration using Informed Designer. To do so, you select the tracking database, specify which cell is to be used to uniquely identify forms, and link tracking information to the tracking database.

**Tracking**

Track this form

**Configure Tracking**

Connection type: Oracle for This platform

Define Connection Connection defined for host 'Oracle Server'.

Choose Table Selected table is 'tracking'.

Cells on form	Links to remote data	Req.
Template ID	Template ID → tracking•TemplateID	
Template Name	FormNumber → tracking•FormNumb...	
Date Sent	Sender → tracking•Sender	
Time Sent	Recipients → tracking•Recipient	
Sender	Date Sent → tracking•DateSent	✓
Recipients	Time Sent → tracking•TimeSent	✓

"Purchase Req." "Oracle"

Cancel OK

Informed Filler can track forms using any database that is accessible via Informed data access plug-ins. These include Oracle, Sybase, and others using ODBC and DAL. Other data access plug-ins that support form tracking might be available after this documentation is published.

By allowing you to use a wide range of databases for form tracking, you have the option of using the database services that might already be available in your organization. For large organizations, this is particularly beneficial because you don't have to install and administer yet another database solely for form tracking purposes.

If you don't already have a database suitable for form tracking, Informed Designer comes with Informed Tracker Server, a DAL (Data Access Language) database server that runs on the Mac OS. Informed Tracker Server can support anywhere from 20 to 250 Mac OS Informed Filler users, depending on the number of forms being tracked and the frequency at which forms are routed.

Each time the Informed Filler user sends a form, a minimum set of information should be stored in the tracking database. This information includes:

- the template ID
- the form number
- the sender's name
- the recipient's name
- the date and time that the form was sent

The template ID is a number or value that identifies the form template—that is, the type of form. Each different template (a purchase order template, a travel expense form template, or a time card template, for example) should have a unique template ID. You specify the template ID using Informed Designer's Template Information command. See "Template Information" in Chapter 2, "Manipulating Documents," of your *Informed Designer Design and Graphics* manual for more information.

The form number is a number or value that uniquely identifies a completed form. The form number (a purchase order number or invoice number, for example) is stored in a cell on the form. Form numbers are often generated using Informed's auto-increment feature. In order to track a form, the form template must contain a form number cell. For information about Informed's auto-increment feature, see "Auto-incrementing Numbers" in Chapter 1, "Adding Intelligence," for more information.

Together, the template ID and form number uniquely identify a particular completed form. These values must be included with the tracking information so that Informed Filler can later retrieve the information that pertains to a particular record when the user requests the tracking status. Also included are the sender's name, the recipient's name, and the date and time that the form was sent.

If you intend to use a database other than Informed Tracker Server as your tracking database, it is necessary that you create a table in your database with the appropriate columns to store the tracking information. This function is normally performed by the database administrator.



The table that you create for form tracking purposes should contain columns for the template ID, form number, sender's name, recipient's name, the date that the form was sent, and the time that the form was sent. All column types can be text. However, if you intend to later analyze the tracking information, you might want to store the date sent and time sent values using date and time columns, respectively. The tracking database you use might impose limitations here. The following is an example specification for a tracking table.

Tracking Table Specifications

Column name	Column type	Length
template_ID	char	20
form_number	char	20
sender_name	varchar	...255
recipient_name	varchar	...1024
send_date	date	
send_time	time	

Be sure that the length of any text column is long enough to store the longest values. Notice that in the example above, the length of the "recipient\_name" column is longer than that of the "sender\_name." This is because a form can be sent to multiple recipients whereas the sender is always only one individual. The column length for "recipient\_name" is longer so that at least two or three recipient names can be stored.

In addition to the minimum set of tracking information, you can also include other "custom" information if you like. When configuring tracking for a form template, you can link any cell on the template to a column in the tracking table. This feature is useful if you plan to later analyze the tracking information to learn about the routing or approval process.

Analysis of the tracking information can reveal useful insights into the efficiency of the routing process. You could, for example, determine the minimum, maximum, and average times that it takes a form to travel through the entire routing process, or from one step to the next. You could experiment by changing the process, then repeat the analysis to see if the change brings improved efficiency.

Depending on the type of analysis that you intend to do, you might need to track more than the minimum set of tracking information. For example, if you want to analyze cycle times as in the above example, you will need to identify the particular step of the routing process when a form is sent and include this with the other tracking information. That way, when you analyze the tracking information, you can determine the dates and times of the different steps that a particular form has travelled.

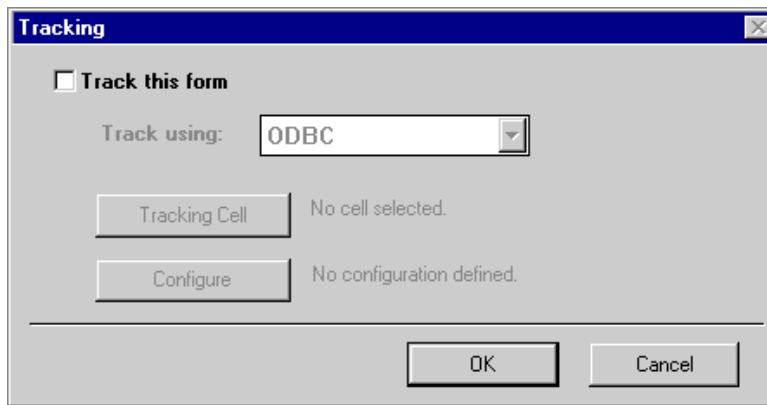
To identify and include the routing step along with the other form tracking information, you could draw a special cell on the template and calculate its value according to different criteria. You might, for example, check if a signature cell contains a signature and, in turn, infer the routing step. You would include this cell along with the other tracking information that is stored in the tracking database.

## Configuring Tracking

Before you can configure form tracking, the tracking database that you intend to use must be set up and running. If access to the database is controlled on a user by user basis, you also need to create an appropriate identity (or identities) for tracking purposes. This, like configuring the tracking database itself, is normally performed by the database administrator.

You must have the data access plug-in for the type of tracking database that you intend to use installed in your plug-ins folder. For example, if you're configuring form tracking to store the tracking information in an Oracle database, you must have the Oracle data access plug-in installed in your plug-ins folder. You must also have the Tracking plug-in installed. These plug-ins are installed automatically when you install Informed Designer.

To configure tracking, choose **Tracking...** from the Configure submenu in Informed Designer's Form menu. The Tracking dialog box for the active template appears.

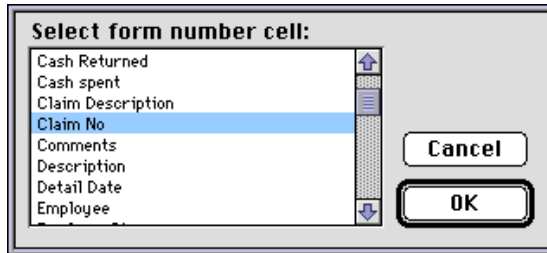


The screenshot shows a dialog box titled "Tracking". At the top left, there is a checkbox labeled "Track this form" which is currently unchecked. Below this, the text "Track using:" is followed by a drop-down menu that has "ODBC" selected. Underneath, there are two buttons: "Tracking Cell" and "Configure". To the right of "Tracking Cell" is the text "No cell selected.", and to the right of "Configure" is the text "No configuration defined.". At the bottom of the dialog, there are two buttons: "OK" and "Cancel".

To turn tracking on, select the 'Track this form' checkbox. Make your choice of database or access method from the 'Track using' drop-down list. The items in this list correspond to the data access plug-ins found in your plug-ins folder. Only those that support form tracking will appear as options in the 'Track using' drop-down list.

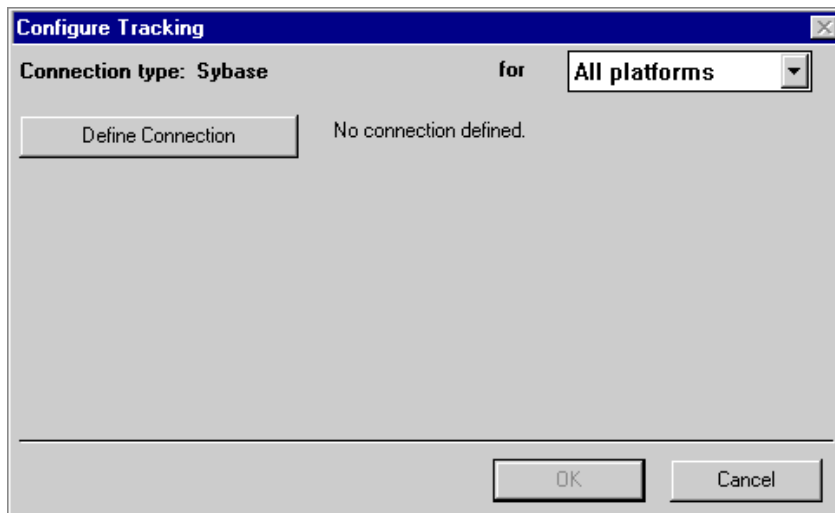
As explained earlier, in order to track a form, the form template must contain a form number cell. This cell is like any other cell, except its purpose is to store a unique number or value for identification purposes. Its value is often obtained from a database or an application like Informed Number Server using Informed's auto-increment value feature. The form number along with the template ID of the form template are used to uniquely identify each completed form for tracking purposes.

In the context of form tracking, the form number cell is called the “tracking cell.” You select this cell by clicking ‘Tracking Cell’ on the Tracking dialog box. A list of the cells on the template, excluding any column cells, appears.



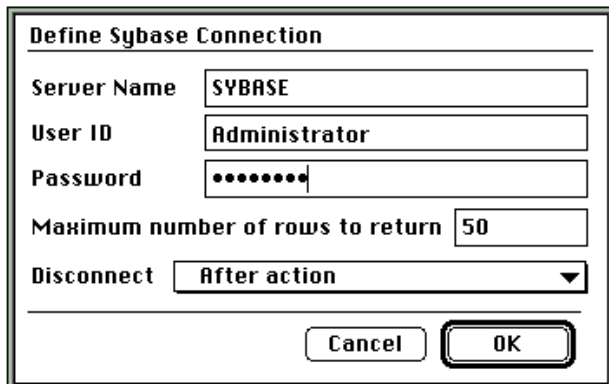
Select the tracking cell, then click ‘OK.’ The name of the cell appears next to the ‘Tracking Cell’ button.

To continue, click the ‘Configure’ button to display the Configure Tracking dialog box.



The database or access method that you’ve selected appears at the top of the dialog box next to ‘Connection type.’ Below the connection type is a single button. The title of this button is specific to the type of database or access method you’ve chosen. For many types, the button title is ‘Define Connection.’

With most databases, it is necessary to provide connection information in order to access the database. Connection parameters usually consist of a user ID, a password, and information that identifies the data source or server. This information is specific to the database to which you’re linking. The Define Connection dialog box for Sybase is shown below.



The image shows a dialog box titled "Define Sybase Connection". It contains the following fields and controls:

- Server Name:** Text box containing "SYBASE".
- User ID:** Text box containing "Administrator".
- Password:** Text box containing masked characters ".....".
- Maximum number of rows to return:** Text box containing "50".
- Disconnect:** Dropdown menu with "After action" selected.
- Buttons:** "Cancel" and "OK" buttons at the bottom.

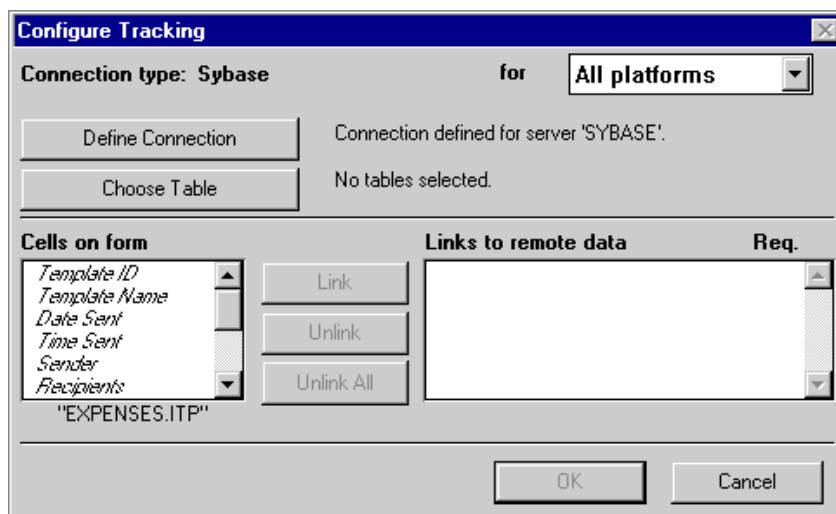
Depending on the database you're linking to, some of the connection parameters might be optional. For example, the 'User ID' and 'Password' parameters on the Sybase Connection dialog box are optional. If you leave either of these parameters blank when you configure form tracking, Informed Filler users will be asked to enter them when they mail a form or request the tracking status of a form.

The details of connecting to a particular type of database or data source are the same regardless of whether the connection is configured for a lookup, for an auto-incrementing cell, for form submission, or for form tracking. The details of the Define Connection dialog box as well as other relevant database-specific information, can be found in the on-line document "DGRPLG.PDF" (Windows) or "Informed Designer Plug-ins" (Mac OS). This document is installed when you install Informed Designer and can be viewed using Acrobat Reader (also included with Informed Designer).

**Note**

Before you can configure form tracking with an external data source, the data source (a dBase file, for example) must already exist. Informed Designer will not create the database or data source for you

If additional configuration information is needed (which is the case for most databases), once you've defined the connection, the Configure Tracking dialog box will change to show additional buttons and two scrolling lists.



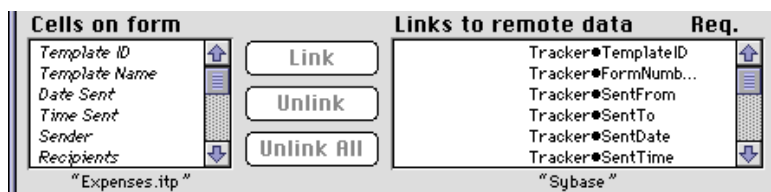
Depending on the type of database or access method you've selected, the appearance of button titles and dialog boxes may vary. The examples shown in this section correspond to tracking with Sybase.

After defining the connection, you then specify the database table in which form tracking information will be stored. You select which table by clicking the 'Choose Table' button.

#### Note

In order to choose a table, Informed Designer needs to connect to the database to obtain the list of available tables and columns. Be sure the connection has been properly defined and the database or data source is available before you click 'Choose Table.' If you have left any connection parameters blank, you might be asked to enter them when you click 'Choose Table.'

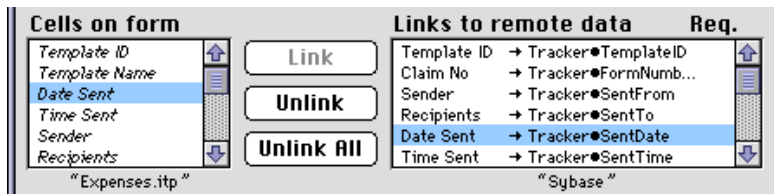
When configuring for Sybase, multiple dialog boxes will appear when you click 'Choose Table,' one to select a database and one to select a table. Once you've selected a table, the corresponding columns will be listed in the 'Links to remote data' scrolling list. Each column name will be prefixed with the name of the table to which it belongs.



The scrolling lists titled 'Cells on form' and 'Links to remote data' are used together to specify which tracking information is entered into which columns in the tracking database. The 'Cells on form' list contains all cells on the form template as well as special items that are specific to form tracking. The items "Template ID," "Template Name," "Date Sent," "Time Sent," "Sender," and

“Recipients” are not cells. They correspond to the tracking information that is available when Informed Filler sends a form (see “How it Works” earlier in this chapter for more information).

The ‘Links to remote data’ list contains the names of the columns in the selected database table. To specify that a value is to be entered into a column, simply select the cell or tracking-specific item in the left list and the column in the right list, then click the ‘Link’ button. The name of the item will appear in the ‘Links to remote data’ list with an arrow pointing towards the column name.



To unlink one column, select the column then click ‘Unlink.’ To unlink all columns, click ‘Unlink All.’

In order for form tracking to work, you must link at least the Template ID item and the cell that you’ve selected as the tracking cell. Furthermore, since the purpose of form tracking is to track the date and time that a form is sent, as well as the sender and recipient(s), you should also link the Date Sent, Time Sent, Sender, and Recipients items.

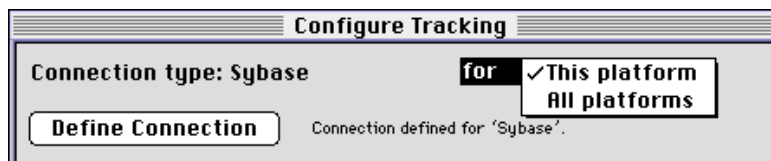
Depending on how the tracking database is configured, linking might also be required for other columns in the database table. If linking is required for a particular column, you’ll see checkmark in the ‘Req.’ column of the ‘Links to remote data’ scrolling list.

Once you’ve defined the connection and performed the necessary configuration, click ‘OK’ on the Configure Tracking and Tracking dialog boxes.

## Configuring for Multiple Platforms

Many of the databases and data sources that Informed can link with are accessible from both the Windows and Mac OS platforms. However, the details of accessing a database or data source from each of the platforms might be different. For example, suppose that you’re configuring form tracking with an Oracle database. For Mac OS users, you might be accessing the Oracle database using the Macintosh Oracle client software (SQL\*NET), whereas on Windows you might be using ODBC instead. The specific parameters needed to connect to the database, therefore, might be different depending on which platform the Informed Filler user is using.

The Configure Tracking dialog box contains a drop-down list with the items ‘This platform’ and ‘All platforms.’



For each different connection type, Informed Designer knows if the configuration details are the same or different for the two platforms. If the connection type is supported on both platforms and the configuration details are the same on both, the ‘All platforms’ option will be available. With this option selected, the linking you configure on one platform will function on both.

If the configuration details are different for each platform, or if the connection type is available only on the platform you’re using, ‘This platform’ will be the only choice available in the drop-down list. For accessing these types of databases and data sources, you have to configure tracking on one platform, then move the form template to the other platform and repeat the configuration. Informed Designer stores the configuration for both platforms. Informed Filler uses the configuration that corresponds to the user’s platform.

Although it may be necessary to configure form tracking twice, once on each platform, the resulting form template document is still a platform neutral document. That is, a single version of the template will work with Informed Filler on both platforms. Informed Filler automatically uses the configuration information that is appropriate for the user’s platform.

## Using the Informed Tracker Server

Informed Tracker Server is a DAL (Data Access Language) database that runs on Mac OS compatible computers. For Informed Filler users who also use Mac OS compatible computers, Informed Tracker Server can act as your tracking server. Depending on the number of forms being tracked and the frequency at which forms are routed, Informed Tracker Server can serve anywhere from 20 to 250 users.

Installation of Informed Tracker Server requires that you install client, server, and administration components. Be sure to install the server component on a Mac OS computer that is connected to the same network as the Informed Filler users. Install the client component on all Informed Filler users’ computers. You can install the administration component on the same computer as the server, or on a different computer from which you will administer the tracking database. For detailed installation instructions, see your *Informed Designer Getting Started Guide*.

Informed Filler communicates with Informed Tracker Server using the IAC (Inter-Application Communications) capabilities of the Mac OS. Informed Filler users and the computers on which you run and administer the Informed Tracker Server must have version 7.0 or later of the Mac OS

installed. These computers must also have program linking turned on with the appropriate access privileges set.

Each Informed Filler user can access Informed Tracker Server as a specific person or as a guest. You configure access using the Users & Groups control panel on the server computer. If you configure access for individual users, each user will have to enter his or her name and password each time a form is mailed and when a form's tracking status is requested. Since this can prove to be bothersome, you might consider using guest access instead. For detailed information about program linking and administering access privileges, please see your *Macintosh Owner's Guide*.

## Informed Tracker Server Files

Installation of the Informed Tracker Server installs the server application as well as the tracking database and related files. The server application is installed at the location you specify during installation. The tracking database and related items are placed in a folder named "Informed Tracker Preferences." This folder is installed in the Preferences folder found in the System Folder on the computer's startup volume.

## Starting Informed Tracker Server

To start Informed Tracker Server, double-click the application's icon. After a few seconds you'll see the application's welcome screen and then its main windows.

Various commands are available under the File, Edit, Options, and Windows menus but should not be used. These commands should only be used under the direction of Shana's technical support staff. The only command you should use is the Quit command located in the File menu. Choose **Quit** when you want to quit the Informed Tracker Server application.

## Configuring the Tracker Connection File

In order to access Informed Tracker Server, the Informed Filler user's computer must know which computer the server is running on. The location of the server is stored in a file named "Tracker Connection." You create the Tracker Connection file using the Connection Maker application. This application is installed when you install the administrator component of Informed Tracker Server.

After creating the Tracker Connection file, you must install it in the system Preferences folder on all Informed Filler users' computers.

### Note

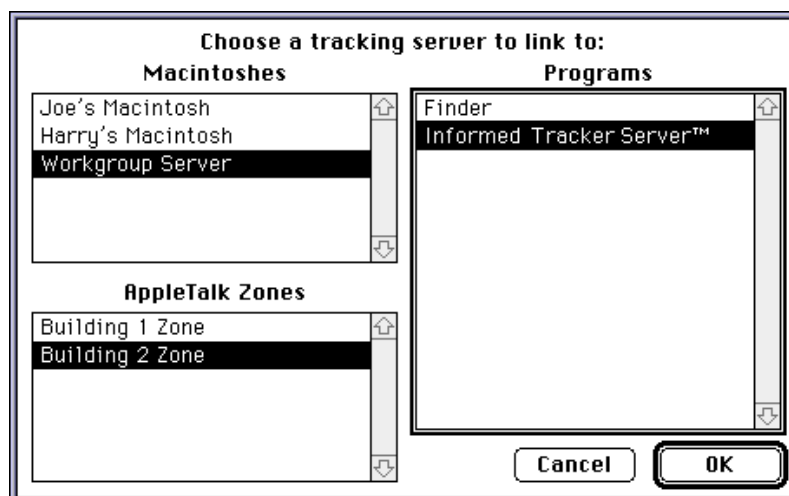
The Informed Tracker Server must be running when you create the Tracker Connection file.



Connection Maker requires that you have AppleScript installed in order to run. To run Connection Maker, simply double-click its icon. You'll see the welcome screen briefly, then an untitled window will appear.



To choose the server where the Informed Tracker Server is running, click the 'Choose Server' button. The Program Linking dialog box appears.



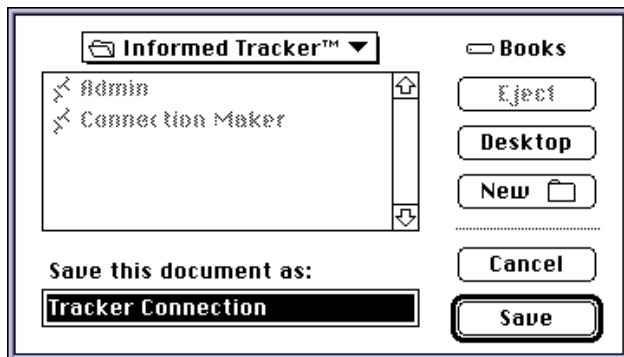
You use this dialog box to locate and select the Informed Tracker Server application. First choose the AppleTalk Zone, then select the Mac OS computer on which the server is running. From the Programs scrolling list, select Informed Tracker Server and click 'OK.' The Tracker Connection dialog displays the location of the server.

Once you've selected the server application, save the Tracker Connection file.

**Note**

You must name the connection file "Tracker Connection." If you use another name, Informed Filler will not be able to find the server.

To save the Tracker Connection file, choose **Save** from the File menu. The Save dialog box appears.



Use the buttons on this dialog box to select the location to save the file. Click 'Save' to save the file. If you specify a different filename here, be sure to change it back to Tracker Connection before placing the file in the Preferences folder of each Informed Filler user's computer.

After creating the Tracker Connection file, you must install it in the Preferences folder on all Informed Filler users' computers. The Preferences folder is located inside the System Folder on the computer's startup volume.

## Changing the Server Location

Occasionally you may have to change the location of the Informed Tracker Server. When the Informed Tracker Server is moved (or if the name of the server computer or its network zone is changed), the information in the Tracker Connection file will no longer be valid. To reestablish the connection, you can either create a new connection file or update the current file. To create a new file, follow the instructions earlier in this chapter.

To update the current file, run Connection Maker (close the untitled window that appears automatically) and choose **Open** from the File menu. Locate the file, open it, and update the server location by clicking the 'Choose Server' button (see earlier). Once you've specified the new location, choose **Close** from the File menu to save and close the file. Be sure to distribute the updated Tracker Connection file to all Informed Filler users, replacing their existing copy.

As explain earlier in "Informed Tracker Server Files," installation of the server installs both the server application and the tracking database and related files. If you move Informed Tracker Server from one computer to another, be sure to move both the Informed Tracker Server application and the Informed Tracker Preferences folder.

## Preparing a Template for Tracking

“Configuring Tracking,” earlier in this chapter, explains how you configure a template for form tracking. This section provides the details specific to configuring tracking for use with Informed Tracker Server. Before continuing, be sure to install the client component of Informed Tracker Server on your computer.

Access to Informed Tracker Server is provided through Informed’s DAL data access plug-in. This plug-in must be installed in the plug-ins folder on your computer and on all Informed Filler users’ computers. When configuring tracking, select ‘DAL’ from the ‘Track using’ drop-down list on the Tracking dialog box.

The screenshot shows a dialog box titled "Tracking". It contains the following elements:

- A checked checkbox labeled "Track this form".
- A "Track using:" label followed by a dropdown menu showing "DAL".
- A "Tracking Cell" label followed by a text box containing ""Claim No."".
- A "Configure" button with the text "No configuration defined." below it.
- At the bottom right, "Cancel" and "OK" buttons.

When you define the connection to the tracking database, you’ll see the Define DAL Connection dialog box.

The screenshot shows a dialog box titled "Define DAL Connection". It contains the following elements:

- A "DAL Extension" label followed by a dropdown menu showing "Informed Tracker".
- A "Host Name" label followed by a text box containing "Informed Tracker Host".
- A "User ID" label followed by a text box containing "Tracker User".
- A "Password" label followed by a text box filled with dots.
- A "Connection Str" label followed by an empty text box.
- A "Maximum number of rows to return" label followed by a text box containing "50".
- A "Disconnect" label followed by a dropdown menu showing "After action".
- At the bottom right, "Cancel" and "OK" buttons.

In order to access a DAL database server, you must specify a DAL extension to use. When you install the client and administration components of Informed Tracker Server, a DAL extension named “Informed Tracker” is installed in your system’s Extensions folder. To select this DAL extension, click the ‘DAL Extension’ button on the Define DAL Connection dialog box. The standard Open dialog box appears allowing you to select an extension. Select “Informed Tracker” in the Preferences folder then click ‘Open.’ The extension name appears next to the ‘DAL Extension’ button.

The ‘Host Name,’ ‘User ID,’ and ‘Password’ connection parameters are required. You must enter the following parameter values:

Parameter Values

Parameter	Value
Host Name	Informed Tracker Host
User ID	Tracker User
Password	Tracker Password

**Note**

Passwords are case sensitive. This means that upper and lower case letters are considered different. Be sure that your Caps Lock key is not on when you enter the password. If you leave any of the above parameters blank, the Informed Filler user will be prompted to enter them whenever a connection the tracking database is needed.

The Informed Tracker Server database accommodates the minimum set of tracking information as well as one user definable column. The database name is “Tracker\_db” and the database table name is “Tracker.” The columns included in this table are listed below.

Column name	Column type	Length
Template_ID	Char	20
Form_number	Char	20
Sender	VarChar	...255
Recipients	VarChar	...1024
Send_date	Date	4
Send_time	Time	4
Comment	VarChar	...1024

“Template\_ID,” “Form\_number,” “Sender,” “Recipients,” “Send\_date,” and “Send\_time” form the minimum set of tracking information. The “Comment” column has no pre-defined purpose. You can use this column for anything you like. Customization of the tracking database is not supported.

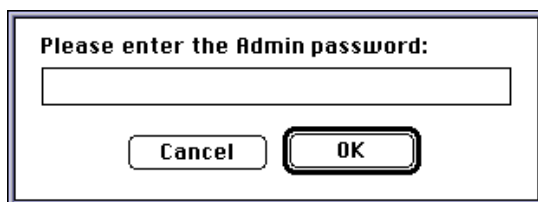
## Administering the Tracking Database

As Informed Filler users fill out and send forms, tracking information accumulates in the tracking database. If you're using Informed Tracker Server as your tracking database, you can view, export, and delete tracking information using the Informed Tracker Admin application. This application is installed when you install the administrator component of Informed Tracker Server. Informed Tracker Admin requires that you have AppleScript installed on your computer.

To start the Admin application, simply double-click its icon.

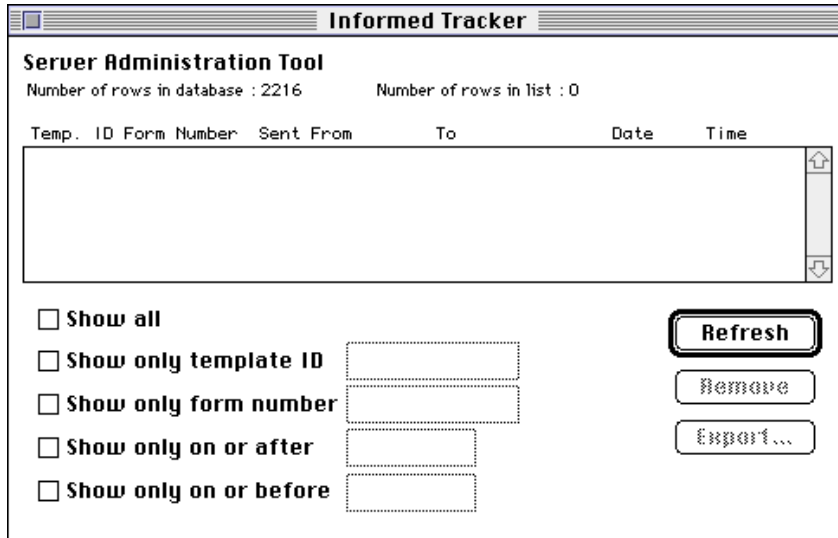
**Note** Informed Tracker Server must be running when you launch the Informed Tracker Admin application. You must also have the correct Tracker Connection file installed in your Preferences folder. For more information, please see “Configuring the Tracker Connection File” earlier in this chapter.

The Admin application is password protected to prevent unauthorized users from viewing or manipulating the form tracking information. When you start the Admin application, the password dialog box appears requesting that you enter a password:



Initially the password is blank. If the password is not blank, enter it before you click ‘OK’ to continue. We suggest that you add a password to secure the tracking database. For information on changing the password, see “Changing the Password” later in this chapter.

After you click ‘OK’ on the password dialog box, the Admin application will automatically connect to the tracking database and display its main window.



Closing the window automatically disconnects the Admin application from the database. You can also disconnect by choosing **Disconnect** from the File menu. To reconnect, choose **Connect** from the File menu. As when you run the Admin application, you'll be asked to enter the Admin password.

The Admin window contains all of the controls you need to view, export, and delete tracking information. The scrolling list initially contains zero rows. Each row corresponds to the tracking information for one routing step. Before you can remove or export tracking information, you must first display it in the scrolling list.

## Displaying Tracking Information

The 'Number of rows in database' indicator, located at the top-left of the Admin window, shows the total number of rows in the database (2216 in the example above). You display tracking information by using the search controls and the 'Refresh' button.

To display all of the tracking information in the database, click the 'Show all' checkbox, then click 'Refresh.' The full contents of the database are displayed in the scrolling list.

**Informed Tracker**

**Server Administration Tool**

Number of rows in database : 2216      Number of rows in list : 2216

Temp. ID	Form No.	Sent From	To	Date	Time
PR-100	019892	John Smith	Harry Jones	06/12/94	10:43 AM
PR-101	019892	Harry Jones	Karen Johnson	06/13/94	03:50 PM
PR-102	019901	Jeff Harris	Debby Peterson	06/02/94	11:31 AM
PR-100	019982	Kevin Dawson	Tracey Larson	07/10/94	04:45 PM
PR-101	029837	Don Murphy	Dave Perman	07/15/94	09:10 AM
PR-102	199820	Glenn Smith	John Murphy	06/09/94	08:06 AM
PR-100	199820	John Murphy	Glenn Smith	06/09/94	08:08 AM

Show all  
 Show only template ID   
 Show only form number   
 Show only on or after   
 Show only on or before

The more entries that are in the database, the longer it will take to refresh and scroll through the list. That's why it's better to refine your search before clicking 'Refresh.'

To display the tracking information for a specific form, click the 'Show only template ID' checkbox and type the template ID into the text box. Next, click the 'Show only form number' checkbox and type the form number into its adjacent text box. Then click the 'Refresh' button. In the example below, the tracking information for a form with the template ID 'Purchase Requisition,' and the form number 'PR-01274' is displayed.

**Informed Tracker**

**Server Administration Tool**

Number of rows in database : 2216      Number of rows in list : 3

Temp. ID	Form Number	Sent From	To	Date	Time
Purchase	PR-01274	Mark Johnson	Sharon Miller	8/2/94	11:44:51
Purchase	PR-01274	Sharon Miller	Harry Schwartz	8/2/94	11:47:22
Purchase	PR-01274	Harry Schwartz	Karen Madsen, cc	8/2/94	11:50:44

Show all  
 Show only template ID   
 Show only form number   
 Show only on or after   
 Show only on or before

If you want to refine your search even further, you can enter a date in either or both of the ‘Show only on or after’ or ‘Show only on or before’ text boxes.

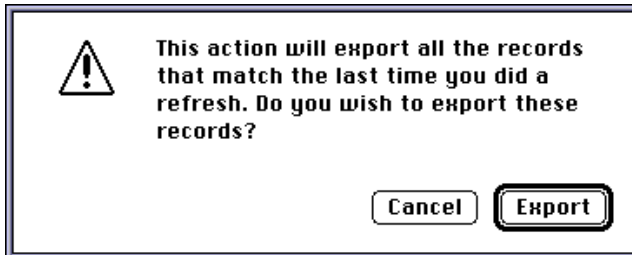
## Removing Tracking Information

You can permanently remove the tracking information for all the forms in the database, or for only specific forms. To remove tracking information, first perform a search and display the desired tracking information in the list (see “Displaying Tracking Information” above), then click ‘Remove.’ A dialog box appears asking you to confirm the operation. After confirmation, every row displayed in the list is permanently removed from the tracking database.

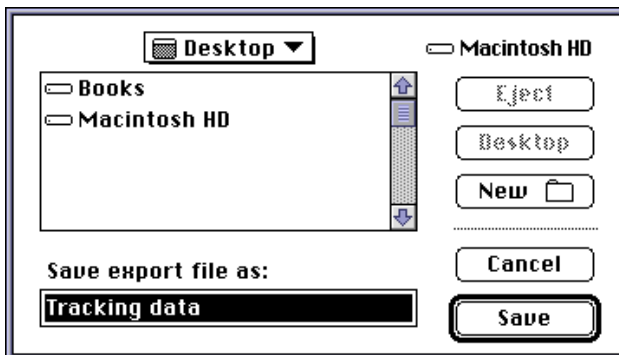
## Exporting Tracking Information

You can export the tracking information displayed in the list into a tab delimited text file. The text file can then be opened by a word processor, or imported into a database or spreadsheet for analysis. The text file can also be used for archival purposes.

To export the tracking information, first perform a search and display the tracking information in the list (see “Displaying Tracking Information” above), then click ‘Export.’ The following message appears, asking you to confirm the export operation.



To cancel the operation, click ‘Cancel.’ To proceed with the export operation, click ‘Export.’ You’ll be asked to name the file and choose the disk or volume to store it on.





Specify the location to store the text file, then click 'Save.' To cancel the export operation, click 'Cancel' instead.

## Changing the Admin Password

When you run the Admin application and connect to the tracking database, you're asked to enter the Admin password. The Admin password is initially blank. To change the Admin password, choose **Admin Password** from the File menu. The following dialog box appears:



**Please enter the old admin. password and the new password.**

**Old password:**

**New password:**

If the current Admin password is blank, do not enter anything in the 'Old password' text box. Otherwise, enter the current password. Enter the new password in the 'New password' text box, then click 'OK.'

### Note

The Admin password is case sensitive. This means that upper and lower case letters are considered different. Be sure that your Caps Lock key is not on when entering a password.

If the old password that you enter is not the current Admin password, you'll see a message indicating so. You cannot change the Admin password if you do not know the current password. If you've entered the correct old password, you'll then be asked to reenter the new password to be sure that you've entered it correctly.



**Please reenter the new password to make sure you typed it correctly the first time.**

**New password:**

:

:

Type your new password again, then click 'OK.' If the new password matches the one you entered on the previous dialog box, your new password will then take effect. Otherwise, you'll see a message asking you to reenter the new password again.



## Authorizing Form Templates

In this chapter:

- How it Works 7-2
- Verification with Informed Filler 7-5
- Authorizing and Verifying Templates 7-6

# 7

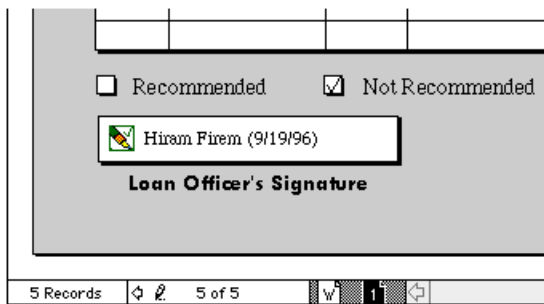
## Authorizing Form Templates

Chapter 2 explains how you can use signature cells to allow Informed Filler users to digitally sign completed forms, much like you sign a paper form. In this chapter, you'll learn how you can further protect the integrity of your electronic forms by authorizing templates for use in your organization. Authorizing templates provides the same authentication and tamper detection for your form templates as signing completed forms with Informed Filler does for form data. With authorized templates, Informed Filler users can verify the authenticity of a template at any time.

### How it Works

Signing a completed form with Informed Filler signs only the form data, leaving the template open to tampering. Without changing the data on a form, a malicious user could alter the form template and, in doing so, change the meaning or context of the data.

For example, suppose a loan application must go through two levels of screening before it can be approved. A loans officer interviews the client and then selects a 'Not recommended' checkbox on the application form and signs it with a digital signature. The application then goes to the bank manager who has the final authority to approve or reject the loan application.



If a person wanted a loan to be approved, he could secretly alter the form template used by the bank manager so that the application appears to have been recommended by the loans officer. To do so, he would simply use Informed Designer to change the title of the 'Not recommended' checkbox field to 'Recommended,' and the 'Recommended' title to 'Not recommended.' When the bank manager views the data with the altered template, it might look like the one shown in the illustration on the following page.

<input type="checkbox"/> Not Recommended	<input checked="" type="checkbox"/> Recommended
	
<b>Loan Officer's Signature</b>	

5 Records | 5 of 5

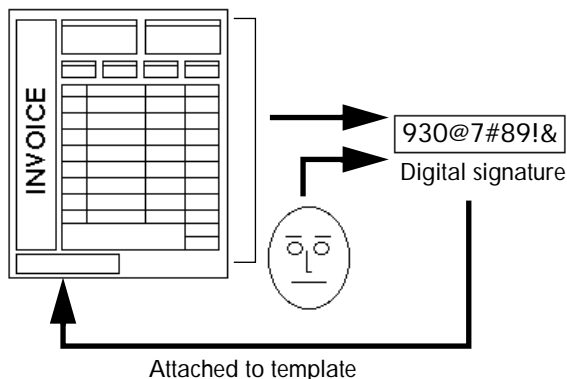
Verifying the loans officer's signature would show that the data on the form has not been altered because the values of the checkboxes have not changed. However, by switching the cell titles on the form template, the meaning or context of those values is different. The bank manager would be misled to believe that approval of the loan application was recommended.

In the example above, the tampering is fairly obvious and is used to illustrate the point. In real world situations, tampering could be much more subtle while still causing serious security breaches.

Informed Designer's Authorize command provides a method for securely authorizing a form template. Doing so allows the Informed Filler user to check the template to ensure its representation is authentic (see "Verifying with Informed Filler" later in this chapter).

Like signing completed forms with Informed Filler, Informed Designer relies on signing services such as Nortel's Entrust or Apple's DigiSign for authorization purposes. Informed Designer accesses signing services through the Informed signing plug-ins you have installed in your plug-ins folder.

When you authorize a template, Informed Designer examines the characteristics of the template, including all objects, their size, position, and attributes. The signing service signs, or processes, this information to create a digital signature. A digital signature is like a special number that is derived from information about the person signing and the characteristics of the template being authorized. This number can reliably identify the signer and detect any changes in the authorized template. While the digital signature is stored with the template, the authorization process does not alter the template itself.



Once a template has been authorized, you or the Informed Filler user can verify its authenticity. The verification process involves re-creating parts of the digital signature using the current template, then comparing the results with the original signature. If they are not equal, then either the authorized template or the digital signature itself has been changed or tampered with.

## Accommodating Revisions

Over time, templates usually need to be revised. A revision can be as simple as adjusting the spacing or size of a cell, or as significant as adding or removing cells, or changing calculations and check formulas to account for changes in the underlying workflow process. In a technical sense, even the simple conversion of a template to a newer format when the Informed software is upgraded to a new version can constitute a “new” version of the template. (Even though the template’s visual appearance and functionality do not change, the internal—byte for byte—representation of the template might.)

When changes to a template are needed, the form designer decides whether or not it is appropriate to revise the existing version of the template, or to create and introduce a completely new template without affecting the previous version. For example, revising an existing template to accommodate a cosmetic change (such as a new company logo, or changing the size of a cell) might not affect the meaning of the data on previously filled out forms, whereas adding or removing cells on an existing template might.

### Note

Care must be taken when revising and replacing an existing version of a template with a new version. Be sure to do so only if the new version provides an appropriate context for Informed Filler users to “see” the data for forms they’ve already filled out. For information about form revision and related issues, please see Chapter 8, “Form Template Distribution and Revision.”

It is because of the need to revise templates, particularly in minor ways, that it is necessary to separate the authorizing of templates from the actual signing of the data on completed forms. If signing a completed form involved signing both the form data and the template together, you would not be

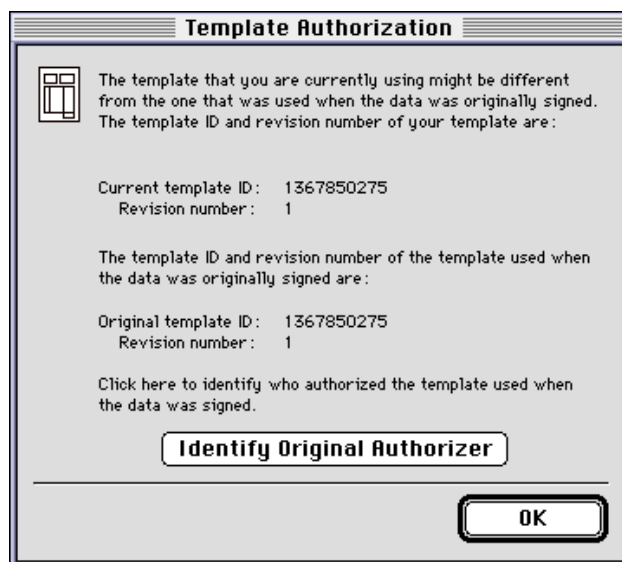
able to later replace the template with a new revision, even if doing so were appropriate, given the nature of the template revision changes. Replacing the template would invalidate the signature of the previously completed and signed form because the template would have changed in comparison with the one that was used when the completed form was signed.

## Verification with Informed Filler

Authorizing a template provides information that can be useful to many users during the processing of the form. For example, if John is going to fill out an expense form with Informed Filler, he can first verify the template he is using by choosing the Verify Template command from the Signatures submenu. The template is verified and the details of the person who authorized it are revealed. If the template has been altered or tampered with, the verification process will reveal this.

When John fills out and signs his form, information about the version of his template and the person who authorized it is included with his digital signature. This information is useful later when John's signature is verified. The form is then sent to Donna, John's manager, for approval.

Like John, Donna can verify that the template she is using (which could be a different version from the one John used to complete his form) is valid before she approves or rejects the form. Furthermore, when she verifies John's signature to ensure that his data has not been altered, she can also view the Template ID and Revision Number of the template that John used. The name of the person who authorized John's template is also revealed.



For more information on verifying signatures and templates with Informed Filler, see Chapter 4 of your *Informed Filler User's Manual*.

## Authorizing and Verifying Templates

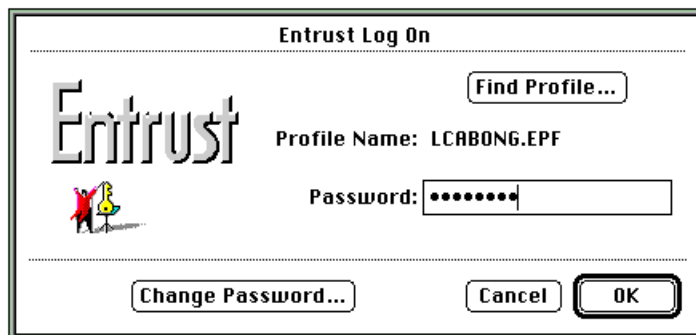
Like signing and verifying completed forms with Informed Filler, you can authorize and verify a template using any available signing service. Informed Designer accesses a signing service through the use of Informed signing plug-ins. The use of signing plug-ins makes it easy for Shana to provide support for new signing services as they become available. At the time this documentation was prepared, Informed Designer included two signing plug-ins, one for Nortel's Entrust signing service, and one for Apple's DigiSign.

To authorize a template, choose **Authorize...** from the Authorization submenu under the Form menu. If you have more than one signing plug-in in your plug-ins folder, you'll be asked to choose which service to use.



Select the signing service that you want to use from the 'Sign using' drop-down list, then click 'OK.' To cancel the Authorize command, click 'Cancel' instead.

After you click 'OK,' you'll see the authentication dialog box that corresponds to the signing service you are using. Follow the signing procedure for your chosen service to authorize the template. Below is the authentication dialog box for Entrust.





To verify a template, choose **Verify...** from the Authorization submenu under Form. Informed Designer verifies the template and, if valid, displays the name of the person who authorized it.



If verification fails, you'll see a dialog message indicating so.

Some signing services allow you to remain authenticated after you've done so once. Informed Designer will automatically log off when you quit the application. You should be careful to never leave your computer unattended while you're authenticated. Otherwise someone else might use your signing identity to authorize templates or sign forms. You can explicitly log off from a signing service by choosing **Log Off Service** from the Authorization submenu.

An authorized form template can be "de-authorized" by choosing the De-Authorize Template command from the Authorization submenu under Form. If you intend to revise a previously authorized template, you may choose to de-authorize it first, then authorize it again once all revisions have been made.





## Form Template Distribution and Revision

In this chapter:

- Background 8-2
- Overview 8-2
- How it Works 8-4
- Maintaining Distribution Center Profiles 8-8
- Revision Information 8-12
- Maintaining Distributed Templates 8-13



## Form Template Distribution and Revision

Once a form template is designed, it must be distributed to Informed Filler users. When a template is revised, the new revision must also be made available to Informed Filler users. This chapter provides guidelines for distributing new form templates and new revisions of templates. Also described are Informed's built-in template distribution features.

### Background

Informed allows form templates to be accessed either locally on each Informed Filler user's computer, or via a central file server. Both methods have advantages and disadvantages. Storing templates locally means that the Informed Filler user can use the templates any time, even while disconnected from the network. Storing templates locally, however, means that users may not always have the current versions of the templates, and introduces additional network costs for the distribution of templates to all Informed Filler users. As explained in Chapter 1, "Overview," of your *Informed Designer Design and Graphics* manual, each Informed Filler user sets a preference to specify where templates are stored.

Accessing form templates from a central file server simplifies distribution. Once a new template is designed, making it available to Informed Filler users is as simple as placing the template in the appropriate directory or folder on the designated file server. When a template is revised, the existing template on the file server is replaced with the new one. Central storage, however, requires that Informed Filler users be connected to the server at all times in order to access the templates.

Informed's form distribution features give Informed Filler users all the benefits of both local and network based template storage. Templates can be stored locally without the need to be "on-line" to use them, and Informed's distribution feature automatically notifies the user when new versions are made available.

### Overview

After a form template is designed and distributed for use, it is normal to expect the template to change from time to time. There are many different reasons why a template might change. Below is a list of some example reasons.

- a typing mistake is found
- the size of a cell must be adjusted to allow for more information
- a newer color version of the company logo is available and should be incorporated on all form templates
- the company has relocated and the address on all form templates must be changed
- an error was identified in the check formula for a cell

- a new cell is needed to store additional information on the form
- a tax form is being revised for a new year; some tax calculations are being changed

When you change a template, it is important that you decide carefully to introduce the new version either as a replacement for the previous version, or as a new template. This decision is based on the nature of the changes being made.

Consider the example of a travel expense form. At the beginning, version 1 of this form template is created and distributed for use by Informed Filler users. These users begin filling out forms as required. Suppose that you later realize that there's a typing mistake in one of the cell titles on the template. You proceed to fix the mistake and, in doing so, create the second version of the template. In this situation, the new version of the template should replace the previous version. You can ask yourself the question "Would it have been appropriate for the previous version of the template to include the changes made in the new version?" If your answer is "yes", then you probably want to replace the previous version with the new version.

When you replace a template with a newer version, it is important to realize that any forms that Informed Filler users have previously filled out will now be "seen" through the new template. With the example of the typing mistake, this means that a travel expense form that was filled out using the template that had the mistake will now be viewed and manipulated using a template with the correction.

In contrast, consider the example where the travel expense form is revised for a new calendar year. Suppose that the revision includes changes both to the visual appearance of the template as well as the calculations and check formulas that affect the mileage rate that is permitted for personal car usage. For the new year, employees are paid 20 cents per mile rather than 15. Also, a cell that was used to store the employee's date of birth has been removed because it is information no longer needed on the travel expense form.

In this case it's important that the new version of the form template be introduced as a new template rather than replace the previous version. If you were to replace the previous version, Informed Filler users would see the forms that they filled out for the previous year through the template for the new year. The employee's birth date would be missing, and if the user happened to edit a form that was completed in the previous year, the mileage amount would recalculate according to the new year's calculations. To avoid this from happening, it would be important to introduce the new year's version as a new form template. To do so you would assign the new template a unique Template ID and use a revision number appropriate for the first version of a new template (perhaps "1").

When deciding whether to replace the previous version of a template with a new revision, or introduce the new revision as a new template, you should keep the following points in mind.

- If the changes made in the new revision change the meaning or context of the data, then you probably want to introduce a new template rather than replace the previous version.
- Be cautious of templates with signature cells. If forms have been filled out and signed using a signature cell on the template, be sure to never replace the template with a new version that affects the signed data or the signature cell itself. Doing so could invalidate the existing digital signatures.

For example, suppose that a signature cell on a template signs seven other cells on the template. With this version of the template, Informed Filler users fill out and sign forms. You later decide to delete a cell that is signed by the signature cell. Since the signature originally signed seven cells and now the form contains data for only six, the signed data has essentially changed.

## How it Works

If you choose to distribute form templates to Informed Filler user's computers, Informed's built-in forms distribution capabilities can help you automate the process.

Distribution of a template is a two step process. First, the template is made available at one or more distribution centers. Second, Informed Filler users connect to a distribution center and copy the template to their local templates folder. Informed Filler makes it easy to select and copy a template from a distribution center. Furthermore, depending on the revision options you select, Informed Filler will also notify the user when a new revision of a template is made available.

The posting of a template to a distribution center using Informed Designer, and the copying of that template to the Informed Filler user's templates folder are accomplished through the use of Informed distribution service plug-ins. A distribution service plug-in enables distribution via a particular type of distribution service. At the time this documentation was prepared, Informed Designer included distribution service plug-ins for the following types of distribution centers:

- file server
- FTP server

With these distribution service plug-ins, you can use Informed's built-in distribution features to distribute templates via file servers and FTP servers. By using distribution service plug-ins, Shana can easily support new methods of distribution by simply developing new plug-ins.

Before you can post a new form template or revision to a distribution center, and before the Informed Filler user can access the distribution center, you must create a distribution center profile. A distribution center profile is a file that contains information that identifies the distribution center along with the connection information necessary to connect to the service. For example, a distribution center profile might specify an FTP server IP address, the user ID and password necessary to connect to the server, and the path at which distributed templates are stored.

You create and edit distribution center profiles using Informed Designer's Distribution Centers command. To do so, the distribution center—that is, the file server or FTP server, for example—must be available and accessible.

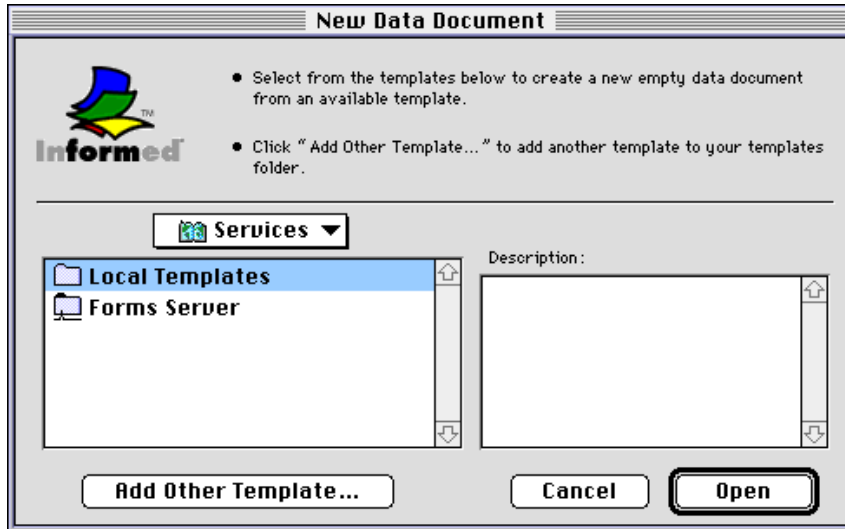
Before you can add or update a form template at a particular distribution center, and before the Informed Filler user can access the distribution center, the corresponding distribution center profile must be placed in a folder named "DISTCTRS" (Windows) or "Distribution Centers" (Mac OS). This folder is found in the "PREFS" (Windows) or "Preferences" (Mac OS) folder which, by default, is located in the Informed folder.

The distribution center profile must be installed on both your computer, and on all Informed Filler user's computers needing access to the distribution center. When you create a distribution center profile using the Distribution Centers command, Informed Designer automatically places the file in your distribution centers folder.

When you add a template to a distribution center, you first select the distribution center profile that identifies the distribution center. You then specify where in the distribution center's hierarchy you would like to place the distributed template. Most types of distribution centers allow you to create folders so that you can organize your distributed templates by type or category.

After adding a template to a distribution center, Informed Filler users can then navigate to find the template and copy it to the place where they store all of their templates. In order to do so they must also have the distribution center profile installed in their "DISTCTRS" (Windows) or "Distribution Centers" (Mac OS) folder.

For Informed Filler users, access to distribution centers and the templates they contain is available from the New Data Document dialog box. This dialog box appears when you choose Informed Filler's New Document command. When creating a new form data document with Informed Filler, the user is asked to select which form template to use. It is here where the user can navigate through the available distribution centers to find a template that they have not yet filled out.



The New Data Document dialog initially shows a list of all templates in the user's templates folder. Selecting 'Services' from the drop-down list reveals the distribution centers that correspond to the distribution center profiles found in the user's "DISTCTRS" (Windows) or "Distribution Centers" (Mac OS) folder. To display the templates available at a distribution center, the user selects the center in this list and clicks 'Open.' Informed Filler connects to the distribution center and lists the templates available. The user selects a template and clicks 'New' to copy it to his or her local templates folder. The template is then available for use with Informed Filler.

Access to distribution centers is also available when the Informed Filler user opens a form data document for which the required template cannot be found. A dialog box similar to the New Data Document dialog box appears providing the same method for connecting to a distribution center and selecting a template.

A distributed template contains distribution information which describes the distribution centers in which the template is available. Informed Filler uses this information to periodically check distribution centers for new revisions. You can choose various settings that determine when and how often Informed Filler checks the distribution centers. You can even specify a status and enter a status message for the template.

When a revision check occurs, Informed Filler compares the revision number of the locally stored template with that of the distributed template. If they are different, it is assumed that the distributed template is a newer version and Informed Filler proceeds to notify the user and, at the user's request, copy the new version to his templates folder.



To summarize, the steps necessary to distribute a template using Informed's built-in distribution features are:

- create the new form template
- establish a distribution center (a file server or FTP server, for example)
- create a distribution center profile using Informed Designer's Distribution Centers command
- add the template to the distribution center
- distribute the distribution center profile to all Informed Filler users
- each Informed Filler user selects the template from the distribution center

As explained earlier, when a form template is revised, you need to decide whether to replace the earlier version with the new version, or introduce the new version as a different form. To replace the earlier version:

- make the necessary changes to the current version of the template
- change the template's revision number (be sure to leave the Template ID unchanged)
- update the distributed template in the distribution centers where it is available

To introduce a new version as a different form:

- save a copy of the current version of the form template (use Informed Designer's Save As command)

Then, using the new copy of the template:

- remove any distribution centers from the template's distribution list (do not delete the distributed templates)
- assign a new unique Template ID and an appropriate revision number to the template
- make the necessary changes to the template
- add the new template to the distribution center or centers

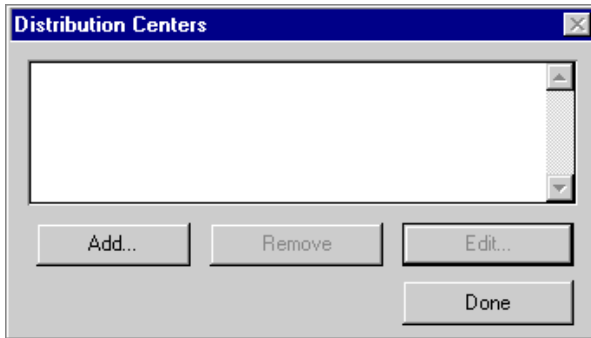
## Using Multiple Distribution Centers

You use Informed Designer's Distribution command to add a template to a distribution center. You can add a template to more than one distribution center. You might do this in order to make a template available at primary and backup distribution centers.

When Informed Filler performs a revision check to see if a new version of a template is available, it does so by checking at the first distribution center on the distribution list for the template. If that distribution center is inaccessible for any reason, the second distribution center is checked, and so on, until the revision check is successful. This behavior allows you to configure multiple distribution centers for redundancy purposes. A particular distribution center can act as the primary distribution center, whereas a second center can act as a backup in the case that access to the primary distribution center is interrupted.

## Maintaining Distribution Center Profiles

You create, edit, and remove distribution center profiles using Informed Designer's Distribution Centers command. Choose this command from the Configure submenu under Informed Designer's Form menu to display the Distribution Centers dialog box.

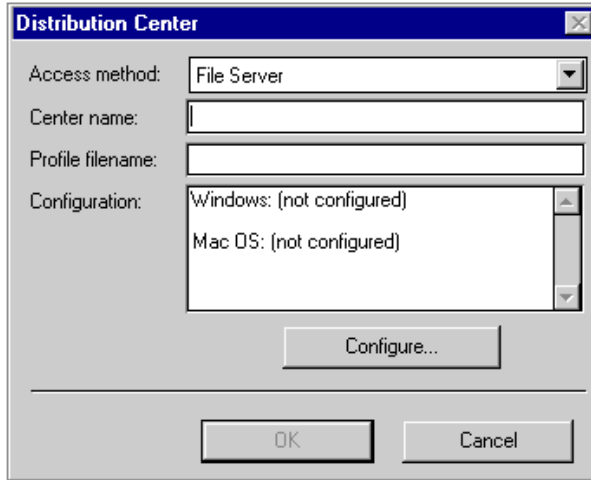


The scrolling list on this dialog box shows all distribution center profiles in your “DISTCTRS” (Windows) or “Distribution Centers” (Mac OS) folder. This folder is located in Informed's “PREFS” (Windows) or “Preferences” (Mac OS) folder. (For information about Informed's special folders and where they are located, please see “Where Everything Goes” in Chapter 1 of your *Informed Designer Design and Graphics* manual.)

You click the different buttons on the Distribution Centers dialog box to add, edit, and remove distribution center profiles. When you've finished working with this dialog box, click the 'Done' button.

### Adding or Editing Distribution Center Profiles

To create a new distribution center profile, click 'Add.' To change an existing profile, select it in the list, then click 'Edit.' The Distribution Center dialog box for the new or existing distribution center profile appears.



The image shows a dialog box titled "Distribution Center". It has a blue title bar with a close button. The dialog contains the following elements:

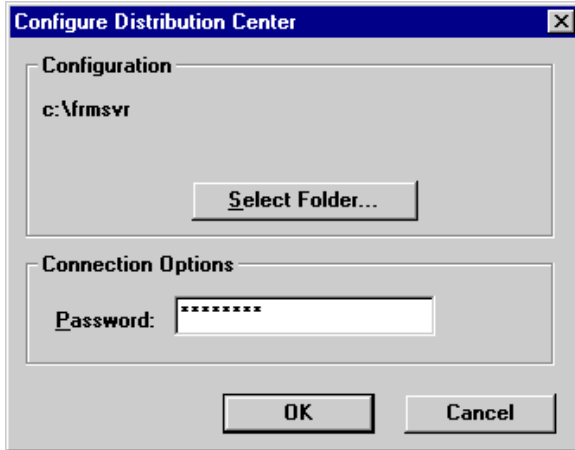
- Access method:** A drop-down menu currently showing "File Server".
- Center name:** An empty text input field.
- Profile filename:** An empty text input field.
- Configuration:** A list box containing two entries: "Windows: (not configured)" and "Mac OS: (not configured)".
- Buttons:** A "Configure..." button is located below the configuration list. At the bottom of the dialog are "OK" and "Cancel" buttons.

Enter the name of the distribution center and the filename for the distribution center profile in the text boxes provided.

Select the type of distribution service from the 'Access method' drop-down list. The options in this list correspond to the distribution center plug-ins available in your plug-ins folder. To configure the distribution center profile, click the 'Configure' button. The configuration dialog box that appears depends on the access method you've selected.

The details specific to the different access methods are not provided here. This information can be found in the on-line document "DGRPLG.PDF" (Windows) or "Informed Designer Plug-ins" (Mac OS). This document is installed when you install Informed Designer and is viewed using Acrobat Reader (also included with Informed Designer).

The configuration dialog box for configuring a file server distribution center profile for the Windows platform is shown on the following page.



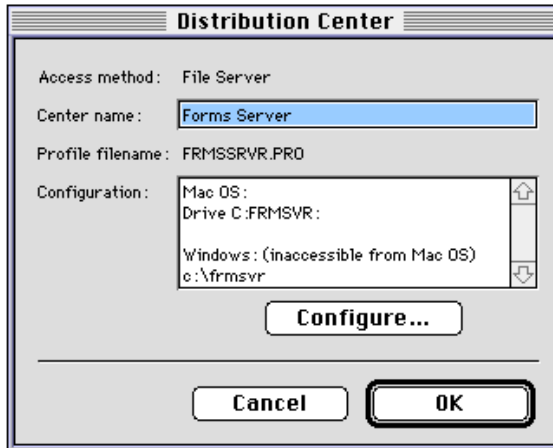
After you've named and configured the distribution center profile, click 'OK' on the Distribution Center dialog box. Informed Designer creates the distribution center profile file and stores it in your "DISTCTRS" (Windows) or "Distribution Centers" (Mac OS) folder.

## Distribution Center Profiles for Multiple Platforms

If you use both Windows and Mac OS compatible computers in your organization, then you might want to allow both Windows and Mac OS Informed Filler users to access distributed templates at the same distribution center(s).

Informed Designer allows you to create a single distribution center profile that works for both Windows and Mac OS users. Depending on the access method that you select, you may or may not have to configure the distribution center profile twice. For some access methods, the configuration details that you specify on one platform are valid, without change, for the other platform. Other access methods require that the configuration be specified once on each platform.

The Distribution Center dialog contains a scrolling list labeled "Configuration." This list shows the details of the configuration.



If a distribution center profile for the selected access method requires platform specific configuration, you'll see separate configuration details for each platform. On the above dialog, the "File Server" access method is selected and the location of the distribution center is specified as "c:\frmsvr" for Windows users and "Drive C:FRMSVR:" for Mac OS users. If configuration is not available for one platform or the other, you'll see "(not configured)" next to the platform identifier.

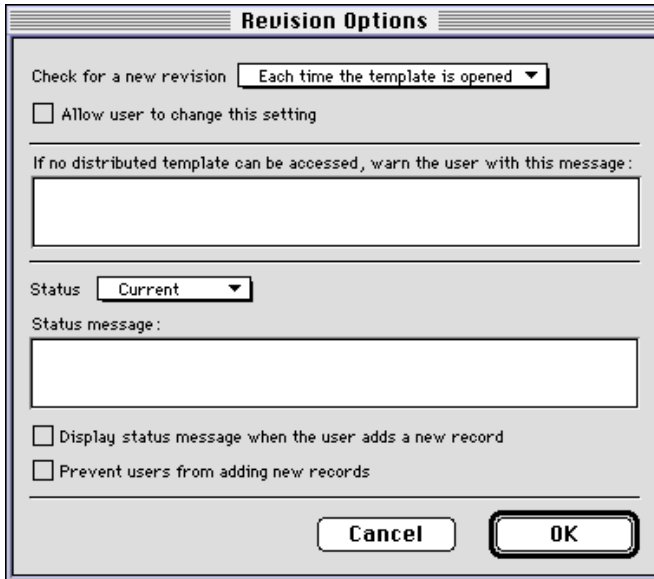
Although it may be necessary to configure a distribution center profile multiple times, once on each platform, the resulting profile document is still a platform neutral document. That is, a single distribution center profile will work with Informed Designer and Informed Filler on all platforms. Informed Designer and Informed Filler will automatically use the configuration information that is appropriate for the current platform.

## Removing Distribution Center Profiles

To remove a distribution center profile, select its name in the list and click 'Remove.' You are asked to confirm the operation before the profile is removed. Removing a distribution center profile deletes the profile file from the "DISTCTRS" (Windows) or "Distribution Centers" (Mac OS) folder.

## Revision Information

Each form template has associated revision options. These options are most applicable for templates that are distributed via Informed's built-in distribution features. Choose **Revision Options...** from the Form menu to display the Revision Options dialog box.



The image shows a dialog box titled "Revision Options". It contains the following elements:

- A label "Check for a new revision" followed by a drop-down menu currently set to "Each time the template is opened".
- A checkbox labeled "Allow user to change this setting", which is currently unchecked.
- A label "If no distributed template can be accessed, warn the user with this message:" followed by a large empty text input box.
- A label "Status" followed by a drop-down menu currently set to "Current".
- A label "Status message:" followed by a large empty text input box.
- Two checkboxes at the bottom: "Display status message when the user adds a new record" (unchecked) and "Prevent users from adding new records" (unchecked).
- Two buttons at the bottom right: "Cancel" and "OK".

The setting of the 'Check for a new revision' drop-down list determines how often Informed Filler will perform a revision check. If revision control is important, you might want to set the frequency to 'Each time the template is opened.' That way, a revision check will occur each time before the template is used. Other settings check only periodically.

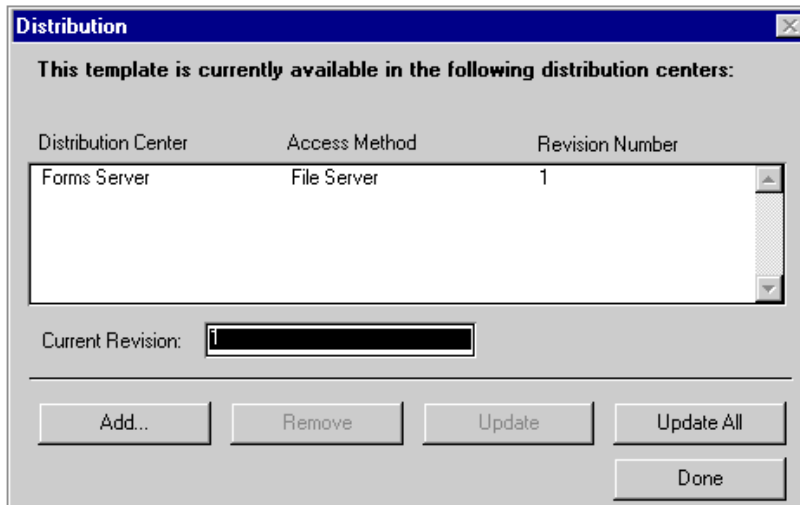
If a revision check occurs, but Informed Filler is unable to connect to the distribution center, or an error of some type occurs while accessing the distributed template, Informed Filler will display the custom message that you specify on the Revision Options dialog box. If you leave this message blank, Informed Filler will substitute a standard message.

The status and status message of a template are intended to indicate to the Informed Filler user the current status of the template. The three statuses include: Current, Non-current, and Discontinued. Choose the status that is most appropriate for the template. You can enter a status message in the text box to provide a more descriptive indication of the template's status. The Informed Filler user can display the status and status message of a template by choosing the Revision Status command from the View menu.

The ‘Display status message when the user adds a new record’ checkbox allows you to warn the Informed Filler user of the status of a template when new record is created. This is particularly useful if a template is used only under certain conditions. The status message could remind the user of the intended uses of the template. To prevent the Informed Filler user from creating new records, select the ‘Prevent users from adding new records’ option.

## Maintaining Distributed Templates

You use Informed Designer’s Distribution command to create, update, and delete distributed form templates, and to add and remove distribution centers to and from a template’s distribution list. Choosing this command from the Configure submenu under the Form menu displays the Distribution dialog box.



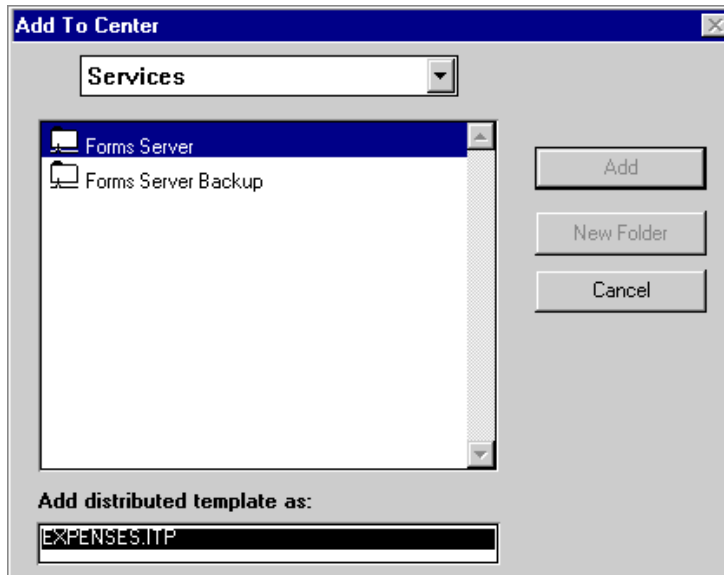
The scrolling list shows all distribution centers that are on the distribution list for the template. Also listed are the access methods used to access each distributed template and the current revision number. When you choose the Distribution command, Informed Designer connects to each distribution center in the list to verify the current revision number of the distributed template before displaying the Distribution dialog. The revision number is shown only if the distributed template is successfully accessed. Otherwise you’ll see the word “inaccessible.” If a distributed template is inaccessible, it could be because of various reasons, including those listed below.

- the distribution center is unavailable
- the distribution center is available, but the distributed template is missing or cannot be accessed
- the required distribution center profile is not found in the DISTCTRS (Windows) or Distribution Centers (Mac OS) folder
- the Informed distribution plug-in for the access method used is not installed in your plug-ins folder

You add, remove, and update distribution centers by clicking the different buttons on the Distribution dialog box. When you've finished using these features, click the 'Done' button to close the Distribution dialog box.

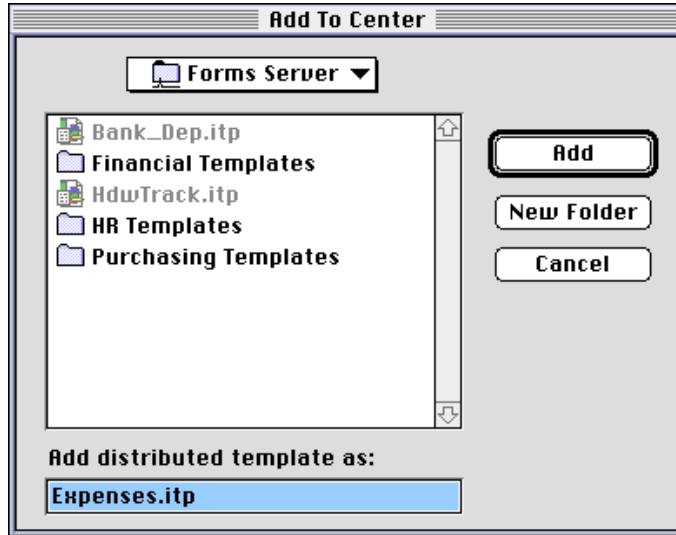
## Creating a Distributed Template

To create a new distributed template at a distribution center, click the 'Add' button on the Distribution dialog box. The Add To Center dialog box appears.



The scrolling list on this dialog box initially lists the distribution centers that correspond to the distribution center profiles found in your DISTCTRS (Windows) or Distribution Centers (Mac OS) folder. To select a distribution center, click its name in the list then click the 'Open' button (Mac OS only), or simply double-click its name. The list changes to show the items at that distribution center.





Many types of distribution centers allow you to organize distributed templates in folders much like you can organize files on a hard disk. If the selected distribution center permits folders, you'll see a 'New Folder' button on the Add To Center dialog box. You can create a new folder by clicking this button and entering a folder name. You can create a folder inside another folder.

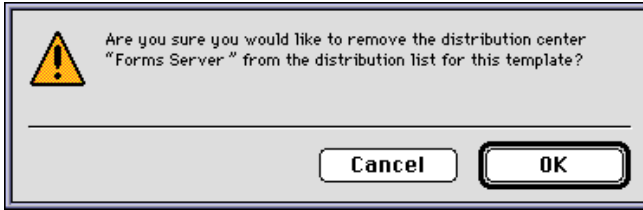
When the scrolling list shows the contents of a folder, the folder's name will appear as the current item in the drop-down list above. You reveal a folder's contents by selecting its name in the list and clicking 'Open' (Mac OS only), or by simply double-clicking its name. You can select an enclosing folder from the drop-down list to navigate out of folders.

The default name of the distributed template is the template's filename. You can change this name by typing a different name in the 'Add distributed template as' text box.

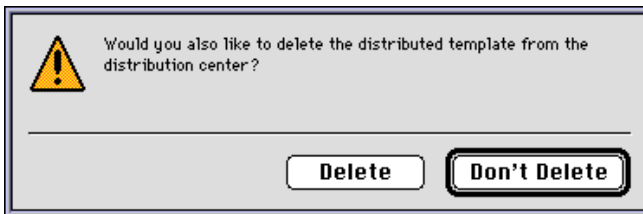
Once you've named the distributed template and selected the folder to store it in, click the 'Add' button. (If the button label is 'Open', press Tab to select the distributed template name. Doing so will change the button label to 'Add.') Clicking 'Add' adds the distribution center to the template's distribution list and creates the distributed template at the specified location.

## Removing a Distribution Center

To remove a distribution center from the template's distribution list, select the distribution center in the list and click 'Remove.' You are asked to confirm that you would like to proceed.



Click 'OK' to continue or 'Cancel' to cancel the remove operation. If you choose to continue, the distribution center is removed from the distribution list. You are then asked if you'd like to also delete the distributed template from the distribution center.



To permanently delete the distributed template, click 'Delete.' Click 'Don't Delete' to leave the distributed template untouched. Be careful when deleting a distributed template. Do so only once Informed Filler users have discontinued use of any templates that are linked to the distributed template.

## Updating Distributed Templates

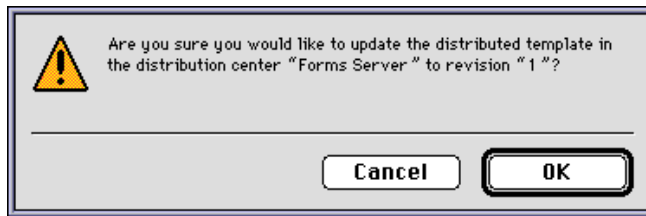
When you revise a form template, you have the option to either replace the previous version with the new version (the revision number is changed but the Template ID remains the same), or introduce the new version as a new template (a new unique Template ID is assigned). You must use caution when replacing a template with a new version because of the potential effects doing so might have on the data for any existing forms that were filled out using the previous version. For more information, please see "Overview" earlier in this chapter.

When you replace an existing form template with a new version, you do so by updating the distributed template at the appropriate distribution centers. That way, each Informed Filler user will be automatically notified of the new version the next time a revision check occurs. In order for Informed Filler to recognize the new version, you must change the revision number of the template. You can do this on the Distribution dialog itself or by using Informed Designer's Template Information command.

**Note**

You can update a distributed template without changing its revision number. However, Informed Filler will not recognize the updated distributed template as a new version of the template and, therefore, will not automatically notify the user of the new version. You may choose to do this in cases where the changes made in the new revision are minor in nature and you do not want to incur the networking costs of updating each Informed Filler user's template. Updating the distributed template in this manner, however, has the advantage of ensuring that any users who obtain the template for the first time will receive the most current version.

The Distribution dialog lists the distribution centers at which the template is available. To update the distributed template at a distribution center, select the distribution center in the list and click 'Update.' You are asked to confirm the operation.



Click 'OK' to proceed or 'Cancel' to cancel the update operation. If you choose to proceed, the distributed template is replaced with the new version.

If you want to update the template at all distribution centers listed on the Distribution dialog box, click the 'Update All' button.

In order to update a distributed template, access to the distribution center and the distributed template itself must be available. If access is available, you'll see the current revision number of the distributed template on the Distribution dialog box. If access is unavailable, it might be because of one of the reasons listed below.

- the distribution center is unavailable
- the distribution center is available, but the distributed template is missing or cannot be accessed
- the required distribution center profile is not found in the DISTCTRS (Windows) or Distribution Centers (Mac OS) folder
- the Informed distribution plug-in for the access method used is not installed in your plug-ins folder

If the distributed template is inaccessible for any of these reasons, you'll see the word "inaccessible" in place of the current revision number on the Distribution dialog box.





## Using Formulas

In this chapter:

- Overview 9-2
- Operands 9-3
- Operators 9-7
- The 'If' Statement 9-14
- Precedence 9-15
- Formula Result 9-18
- Type Conversion 9-19



## Using Formulas

Formulas are an important part of Informed’s data intelligence capabilities. With formulas, you can create cells whose values are calculated or checked when you fill out a form with Informed Filler. You can also use formulas to configure a dynamic tab order for cells on a form. In this chapter you’ll learn about the following topics:

- the uses of formulas
- operands and operators
- constants and functions
- conditional formulas and the IF statement
- formula results
- precedence rules
- automatic type conversion

For information about entering formulas, see “Calculations,” “Data Verification,” and “Conditional Tabbing” in Chapter 1. Functions are explained in detail in Chapter 10.

### Overview

The most common use of formulas is to calculate a cell’s value based on the values of other cells on your form. Calculated cells are automatically entered for you when you fill out a form with Informed Filler or when you test your form with Informed Designer. For example, a cell can use a formula to calculate an extended price based on a quantity and a unit price.

You can use formulas to check for data entry errors. A check formula can test for different error or warning conditions. For example, you could use a check formula to make sure that the discount rate on a sales slip doesn’t exceed twenty percent of the total sale. You can even use alert dialog boxes and help messages to describe error or warning conditions to the person filling out the form.

You can also use formulas to calculate the next tab position for any cell. The result of a tab formula is the name or tab position of the next cell to tab to. By using tab formulas, you can dynamically change the tab order of a form according to different conditions.

Formulas combine values, called *operands*, with special symbols, called *operators*, to derive a particular result. Each operator applies a particular action on its operands. For example, the following formula uses the addition operator (+) to add 2 to the value contained in ‘Cell1.’

```
Cell1 + 2
```

The result of a formula is used according to where the formula appears. It can be the calculated value of a cell, a function parameter, a check condition, a tab condition, or it can be part of a larger formula.

## Operands

The values from which a formula derives its result are called *operands*. You combine operands with operators to create a formula that produces the defined result. An operand can be:

- a value that you enter directly into a formula
- the value of a cell
- the result of a function
- the result of another formula

The following example shows a formula that uses all four kinds of operands. It adds 2 to the value in 'Cell1' and then multiplies that result by the sum of 2, 4, and 8.

```
(2 + Cell1) * Sum (2, 4, 8)
```

When you use an operand with an operator, the operand's type must match the type expected by the operator. For example, the multiplication operator (\*) expects number operands. If the types don't match, Informed will try to convert the operand's value to the appropriate type. You can also use the type conversion functions (ToNumber, ToText, and so on) to explicitly convert an operand's value to a different type. For more information about type conversion and type compatibility, see "Type Conversion" later in this chapter.

## Constants

*Constants*, sometimes called *literals*, are values that you enter directly into a formula. The term constant is used to describe these values because they don't change; they remain constant. For example, the second operand in the following formula always evaluates to the numeric value -1. The result of the formula is always -1 plus the value in the cell called 'Cell1.'

```
Cell1 + -1
```

If you use a formula where all the operands are constants, the formula will always return the same result. For example, the following formula always returns the date August 23, 1996.

```
ToDate ("August 3, 1996") + 20
```

Three kinds of constants are used in the above example: a number constant, a text constant ("August 3, 1993"), and a date constant (the result of the ToDate function). You can also use time, name, and boolean constants as operands in formulas.

## Number Constants

A number constant consists of an optional sign indicator (+ or -) followed by one or more digits. A decimal point can appear anywhere in the digits. Some example number constants are:

```
45
+3.33
-.05
89.99
-545.63
-5.
```

Informed provides two predefined number constants. The values of pi (3.141592654...) and e (2.718281828...) are represented by the constants shown below.

Predefined Number Constants

Constant	Value
Pi or $\pi$ or $\Pi$	3.141592654
e	2.718281828

## Text Constants

Text constants must be enclosed in single or double quotation marks. Any typeable character can appear in a text constant. If the text constant itself contains the same type of quotation marks as those enclosing it, you must precede the quotation mark with the *backslash* character (\). To enter the backslash character, type the Backslash key. The backslash character tells Informed that the next character is part of the text constant. The following table lists some text constants. Note that the constant “\t” adds a tab, and constant “\n” adds a carriage return. ASCII codes can also be entered as text constants. For example, to enter a carriage return into a text string you type “\013.” The carriage return is inserted when the formula is evaluated.

Text Constants

Text Constant	Text Value
"Received - code #546"	Received - code #546
"Received - code \t#546"	Received - code #546
"Received - code \n#546"	Received - code #546
'For this office\'s use only.'	For this office's use only.
"Don't remove this label!"	Don't remove this label!
"Welcome to \"Herb's Diner\""	Welcome to "Herb's Diner"
'A.'	A.

## Boolean Constants

You can use two predefined boolean constants. The boolean values of “True” and “False” are represented by the same words typed without the enclosing quotation marks.



## Name, Date, and Time Constants

To enter name, date, or time constants, you must use the ToName, ToDate, and ToTime functions. These functions convert values from other types to the name, date, and time data types. You enter a name, date, or time constant by enclosing the textual representation of the value in single or double quotation marks and entering that value as the single parameter to the appropriate function. For example, you'd enter the time constant 12:51 PM by typing ToTime ("12:51 PM").

When you enter the textual representation of a name, date, or time constant, you can enter the value in any format that Informed will recognize. For example, the two date constants below are equivalent.

```
ToDate ("January 1, 1996")  
ToDate ("1/1/96")
```

If the result of the function is used as a cell's calculated value, the format of the cell determines how the value is displayed on your form. See the "Cell Types" section in Chapter 1, "Adding Intelligence to Your Forms." for more information.

## Cell References

Formulas can reference information in other cells on your form. To reference a cell, you type the cell's name in the formula. When Informed evaluates the formula, the cell's name is replaced with the current value of the cell.

For example, suppose that your form contains a cell called 'Quantity,' which contains the number of items, and a cell called 'Price,' which contains the price of a single item. You could use the following formula to calculate the extended price.

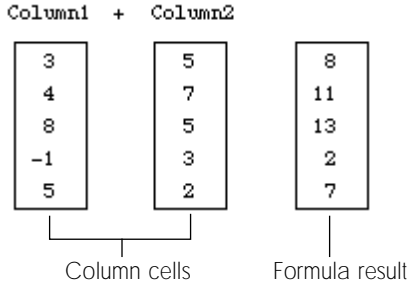
```
Quantity * Price
```

### Note

Never enclose the name of a cell within quotation marks. A cell name within quotation marks would be interpreted as a text constant and not as a cell value.

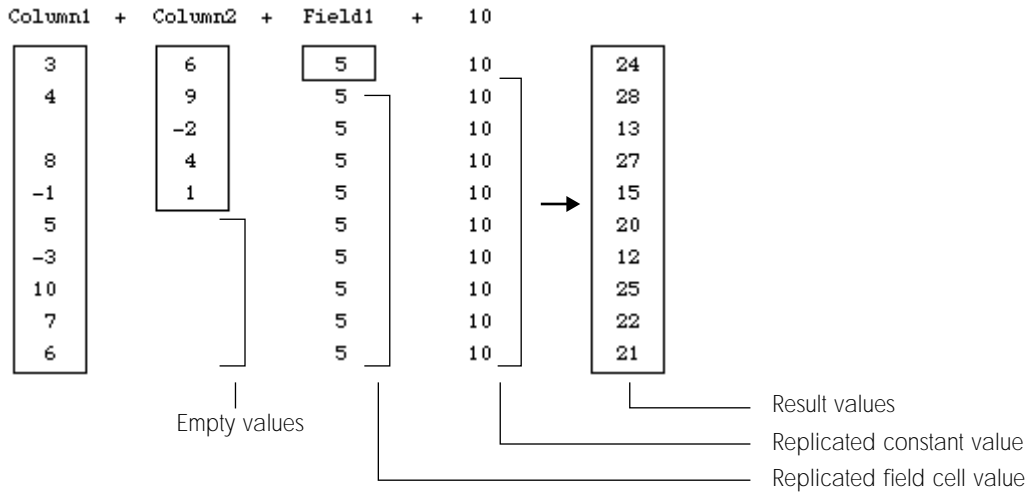
## Column Cells

When you use a column cell as an operand in a formula, the formula's result will also be a column. That is, the result will consist of multiple values. For example, if you use the addition operator to add two column cells, each containing five rows, the formula's result will also consist of five rows. Each row in the result will contain the sum of the corresponding rows in the two column cells.



If the column cells in the above example were of different heights, the formula's result would contain as many values as the longest column. Informed would insert empty values in the missing rows of the shorter column cell.

If you use column cells with field cells or constants in the same formula, the field cell and constant values are applied to each row of the column cells. The figure below illustrates a formula that adds a field cell, a number constant, and two column cells.



You can use column cells in a formula only when the result of the formula is also expected to be a column. If a single value result is required, Informed Designer won't allow you to use a column cell in the formula. For example, suppose that your form contains the field cell called 'Total' and the column cells called 'Quantity' and 'Price.' Informed Designer would not allow you calculate the 'Total' cell as 'Quantity \* Price.'

You can reference a specific row of a column cell in your formula. For example, to reference the third row of the column cell 'Quantity' you would enter 'Quantity[3].' The row identifier can be a numeric constant, or a formula that returns a number.

## Functions

A *function* performs a predefined calculation using a set of input values, called *parameters*. For example, the following function calculates the mean of the numbers 1, 8, and 12.

```
Mean (1, 8, 12)
```

You can use the result of a function's calculation as an operand in a formula. A function can appear in a formula anywhere a cell reference or a constant can appear. As with constants and cells, the result type of a function should match the type expected by the operator with which the function is used. See "Type Conversion" for more information.

When you use a function as an operand, the function is evaluated first. Its result value is then used to evaluate the formula. For example, the following formula calculates the sum of 7—which is the result of the 'Mean' function—and 10 to give a result of 17.

```
10 + Mean (1, 8, 12)
```

For more information about functions, parameters, and function results, see Chapter 10, "Using Functions."

## Formulas as Operands

You can use a formula as an operand to another formula. This allows you to create complex formulas that consist of many operators and operands. For example, the formula 'Cell1 \* Cell2' is used as the first operand to the addition operator (+) in the formula below.

```
Cell1 * Cell2 + Cell3
```

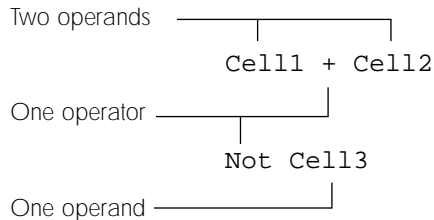
In the above example, the formula 'Cell1 \* Cell2' is evaluated first. Its result value is then used as the first operand to the addition operator.

You can combine any number of operands and operators. There are rules, called *precedence* rules, that determine which formulas are evaluated before others. In the above example, the precedence rules dictate that the multiplication operator (\*) is evaluated before the addition operator (+). For a complete description of the precedence rules, see "Precedence" later in this chapter.

## Operators

*Operators* combine operands to create new values. To create a new value, each operator performs a particular action on its operands. The type of the new value depends on the operator's action and on the types of operands used. Example actions are summation, comparison, and negation.

Most operators combine two operands to produce a new result. Others manipulate only one operand, or any number of operands. For example, the addition operator (+) adds two operands, whereas the boolean negation operator (Not) negates a single operand.



Each operator requires that its operands be a certain type. For example, since the multiplication operator multiplies numbers, its operands must be number values. When an operand's type doesn't match the type required by the operator, Informed will try to automatically convert the value to the correct type. See "Type Conversion" for more information.

Informed provides five kinds of operators:

- arithmetic
- text
- comparison
- boolean
- column

Arithmetic operators perform mathematical operations such as addition and multiplication. Text operators manipulate text values to produce new text values. Comparison operators compare values to check for conditions or make choices. Boolean operators combine boolean values to produce new boolean values. The list operator ( { } ) produces a list of any number of values.

## Arithmetic Operators

Arithmetic operators allow you to create formulas that add, subtract, multiply, divide, and exponentiate numbers. You can also use arithmetic operators to find the quotient and modulus of a division, negate a number, and convert a number to a percentage value.

The symbols for the arithmetic operators are shown in the following table. Along with each operator, the table lists the number of operands the operator requires and a simple example. The result of each example follows the arrow symbol (→).

## Arithmetic Operators

Operator	# of Operands	Description	Example
+	2	addition	$2 + 3 \rightarrow 5$
-	2	subtraction	$2 - 3 \rightarrow -1$
*	2	multiplication	$4 * 5 \rightarrow 20$
/	2	division	$18 / 4 \rightarrow 4.5$
Div	2	quotient	$18 \text{ Div } 4 \rightarrow 4$
Mod	2	modulus	$18 \text{ Mod } 4 \rightarrow 2$
^	2	exponentiation	$4 ^ 3 \rightarrow 64$
%	1	percentage	$1.5\% \rightarrow 0.015$
-	1	negation	$-(2 + 3) \rightarrow -5$

As shown above, you use arithmetic operators to create mathematical equations. For example, the following formula uses the multiplication and percentage operators to calculate 25 percent of the value in 'Price.'

`Price * 25%`

You can also use the addition and subtraction operators to add and subtract date and time values. You can add days to a date and seconds to a time. You can also subtract one date or time from another. The following table illustrates addition and subtraction of dates and times with examples. To avoid repetition, the ToDate and ToTime functions are not used to create proper date and time constants.

## Addition and Subtraction of Dates and Times

Operator	Operand Types	Example
+	date + number	$\text{May } 12 + 2 \rightarrow \text{May } 14$
	time + number	$5:45:30 + 14 \rightarrow 5:45:44$
-	date - number	$\text{Dec } 22 - 2 \rightarrow \text{Dec } 20$
	time - number	$12:30:15 - 14 \rightarrow 12:30:01$
	date - date	$10/24/96 - 10/14/96 \rightarrow 10$
	time - time	$14:20:10 - 14:10:00 \rightarrow 610$

Subtracting a date from another date or a time from another time yields the number of days between the two dates or seconds between the two times. A negative result indicates that the first operand is an earlier date or time than the second. For example, if 'Birth date' is a date cell containing your date of birth, the formula 'Today - Birth date' calculates your age in days (the Today function returns the current date).

## Text Operators

Informed provides you with one text operator. This operator allows you to join two text values to create a new larger text value. The operator is called the *concatenation* operator. It appears in the following table along with an example.

Text Operator With Example

Operator	Operand Types	Description	Example
&	text & text	concatenation	"In" & "formed" → "Informed"

You use the concatenation operator to construct new text values. For example, if 'Airline' is a text cell containing the name of an airline, the formula below creates a message that indicates the airline on which a flight has been booked.

```
"Your flight is booked on " & Airline
```

You can build complex messages by using the result of one concatenation operator as the operand to another.

```
"Your flight is booked on " & Airline & ". Have fun!"
```

If the value of 'Airline' is "Sunny Airlines," the formula returns the result "Your flight is booked on Sunny Airlines. Have fun!."

### Note

You can also use the 'Concat' function to concatenate text values. See Chapter 10 for a complete description of this function.

## Comparison Operators

Comparison operators compare two values. For example, you can compare two numbers to determine if one is greater than, equal to, or less than the other. The result of a comparison operator is always a boolean value; either True or False. The comparison operators are shown in the following table. Each operator is shown with the types of operands allowed and an example using constant values. Again, to avoid repetition, the type conversion functions are not used to create proper date, time, and name constants.

Comparison Operators With Examples

Operator	Operand Types	Description	Example
=	number = number date = date time = time text = text name = name boolean = boolean	equals	4 = 5 → False May 5 1996 = 05/05/96 → True 12:30:00 = 12:30:01 → False "aaaaA" = "aaaa" → False A. New = Al New → False True = True → True

## Comparison Operators With Examples (continued)

Operator	Operand Types	Description	Example
<>	number <> number	not equals	4 <> 5 → True
	date <> date		May 5 96 <> 05/05/96 → False
	time <> time		12:30:00 <> 12:30:01 → True
	text <> text		"After" <> "Before" → True
	name <> name		A. New <> Al New → True
	boolean <> boolean		False <> False → False
<	number < number	less than	5 < 6.5 → True
	date < date		01/01/92 < 05/05/93 → True
	time < time		12:30 PM < 12:30:01 → False
	text < text		"ABC" < "abc" → True
	name < name		J Smith < A Smith → False
	boolean < boolean		True < False → False
>	number > number	greater than	8.9 > 9 → False
	date > date		10/05/93 > 05/05/93 → True
	time > time		12:30:00 > 12:30:01 → False
	text > text		"123" > "abc" → False
	name > name		J Smith > A Smith → True
	boolean > boolean		True > False → True
<=	number <= number	less than or equal to	1 <= 0 → False
	date <= date		Jan 1 <= May 5 → True
	time <= time		15:00:00 <= 15:00:00 → True
	text <= text		"Dog" <= "cat" → True
	name <= name		J Smith <= J Jones → False
	boolean <= boolean		False <= True → True
>=	number >= number	greater than or equal to	5.5 >= 5.5 → True
	date >= date		10/05/95 >= 10/05/93 → True
	time >= time		12:30:00 >= 12:30:01 → False
	text >= text		"say 123" >= "say 456" → False
	name >= name		J Smith >= J Jones → True
	boolean >= boolean		True >= False → True

The comparison operators compare values of the same type. If the operand types differ, Informed will convert both values to text and compare the two text values.

The comparison operators return boolean results True or False. You use comparison operators to create check formulas or formulas that set the values of boolean cells. You also use comparison operators with the IF statement to make decisions or select different actions. The following example checks if the value of 'Quantity' is greater than 5. If 'Quantity' is greater than 5, the value in 'Price' is discounted by 5 percent. If 'Quantity' is less than or equal to 5, the value in 'Price' is returned unadjusted.

```

If (Quantity > 5) Then
  Return Price * 95%
Else
  Return Price
End

```

When you compare two values, Informed follows the comparison rules outlined below.

- Numbers are compared numerically. Larger numbers are greater than smaller numbers, and negative numbers are less than positive numbers.
- Dates and times are compared temporally. A “later” date or time is greater than an “earlier” date or time.
- Text values are compared using the character ordering called ASCII. In general, digits are ordered before capital letters, and capital letters are ordered before small letters. Two text values are compared character by character from left to right until they differ. The relative ordering of the differing characters determines which is the “greater” text value. The result of each of the following comparisons is True.

```

"Hello" = "Hello"
"123 Street" < "Whyte Avenue"
"Informed" > "Inform"
"begin" < "end"
"Design West" < "design West"

```

- Names are compared part by part. The parts of two names are compared in the following order: surname, first name, middle names, prefix, then suffix. If one of the names doesn’t contain a part and the other name does, the name that’s missing the part is the “lesser” name (assuming the names are equal up to that part).
- The boolean value True is greater than False.
- Pictures and Signatures. You can compare two picture values or signature values to see if they are equal.

## Boolean operators

Boolean operators perform the logical operations of “and,” “or,” and “not.” Use boolean operators to check if different boolean values or comparisons are True or False. The following table summarizes the boolean operators.

Boolean Operators With Examples

Operator	Description	Example
And	logical “and”	True And True → True True And False → False False And True → False False And False → False



## Boolean Operators With Examples (continued)

Operator	Description	Example
Or	logical “or”	True Or True → True True Or False → True False Or True → True False Or False → False
Not	logical negation	Not True → False Not False → True

You can use boolean operators to decide if one or more conditions are True. The condition could be the value of a boolean cell on your form or it could be the result of a formula that compares two values using a comparison operator. Use the And operator to check if two conditions are True simultaneously. For example, the formula below checks if the value in ‘Discount’ is greater than or equal to zero and less than 15.

```
Discount >= 0 AND Discount < 15
```

Similarly, you can use the Or operator to check if at least one of two conditions is True. The following formula checks if the value in ‘Discount’ is less than zero or greater than 15.

```
Discount < 0 OR Discount > 15
```

Boolean operators are often used with comparison operators and the ‘If’ statement to select different actions. The following example checks if ‘Override’ is not True and if ‘Size’ is less than 25. If both conditions are True, the text value “OK” is returned. If ‘Override’ is True or if ‘Size’ is greater than or equal to 25, the text value “Not OK” is returned instead.

```
If Not Override AND Size < 25 Then
    Return "OK"
Else
    Return "Not OK"
End
```

## The Column Operator

The column operator consists of the two brace characters “{” and “}.” Use these characters to create a column consisting of multiple values. Simply enclose multiple values, separated with commas, within the brace characters. A column result can be used to set the value of a column cell. The example below uses the column operator to create a column containing the rows of Column1 followed by the rows of Column2.

```
{Column1, Column2}
```

The MakeColumn function can be used in place of the column operator. For information on this and other functions, see Chapter 10, “Using Functions.”

## The 'If' Statement

The 'If' statement provides you with a decision making capability for your formulas. A formula can return different results based on different conditions.

**Note**

You can also use the IFT and IFTE functions in place of the 'If' statement. See Chapter 10, "Using Functions" for more information.

The condition of an 'If' statement can be any formula or function that returns a boolean result. For example, the formula below returns the value of 'Cell3' multiplied by 100 only if the value of 'Cell2' is not equal to zero.

```
If Cell2 <> 0 Then
    Return Cell3 * 100
End
```

The condition for the above 'If' statement is 'Cell2 <> 0.' If the value of 'Cell2' is zero, the formula returns the empty value.

You can select between two different actions by using the word 'Else' with the 'If' statement. If the condition being checked by the 'If' statement is False, the formula after the 'Else' is evaluated. For example, the formula below returns the value "Senior" if 'Age' is greater than 65.

```
If Age > 65 Then
    Return "Senior"
Else
    Return "Junior"
End
```

If the value of 'Age' is not greater than 65 (that is, if 'Age' is less than or equal to 65), the formula returns "Junior" instead.

The 'If' statement is commonly used to create check formulas. A check formula is a formula that checks for different error or warning conditions. The result of a check formula tells Informed if the entry in a cell is valid or not. For example, the formula below checks if the value in 'Shipping' is at least 5 and less than 20.

```
If (Shipping >= 5) AND (Shipping < 20) Then
    Return True
Else
    Return False with Alert 'Discount out of range.'
End
```

If the 'If' condition evaluates to False—which means the value in 'Shipping' is out of range—the formula returns the value False and displays an alert message. For more information about creating check formulas, see “Data Verification” in Chapter 1, “Adding Intelligence to Your Forms.”

You can extend the 'If' statement further by using the word 'ElseIf.' With this word you can choose between several alternative actions. An 'If' statement can contain zero or more 'ElseIf' terms and zero or one 'Else' term. If you use an 'Else' term, it must appear last. The following 'If' statement uses several ElseIf terms and an 'Else' term to choose between several actions. It returns the value in 'Price' scaled by a factor that is determined by the value in 'Age.'

```
If Age < 18 Then
    Return Price * 50%
ElseIf (Age >= 18) AND (Age < 45) Then
    Return Price
ElseIf (Age >= 45) AND (Age < 65) Then
    Return Price * 150%
Else
    Return Price * 175%
End
```

You can use an 'If' statement as an action of another 'If' statement. This is commonly referred to as *nesting* (one 'If' statement is nested inside another). Consider the formula below.

```
If TotalSale < 5000 Then
    If Discount Rate > 0.15 Then
        Return False with Alert 'The discount rate
        cannot exceed 15%.'
    End
Else
    If Discount Rate > 0.20 Then
        Return False with Alert 'The discount rate
        cannot exceed 20%.'
    End
End
```

This formula checks for an invalid discount rate. The outermost 'If' statement checks if the total sale is less than \$5,000. Depending on the result of that condition, the formula then checks for a discount rate that's either greater than 15 percent or greater than 20 percent.

## Precedence

*Precedence* rules are the rules that determine the order in which parts of complex formulas are evaluated. When you create a formula with more than one operator, the precedence rules dictate which operator is evaluated first, which is evaluated second, and so on. Without precedence rules, the same formula could yield different results, depending on the order in which its operators are evaluated. For example, the formula below could be evaluated two different ways.

$2 * 4 + 3$

The multiplication operator (\*) could be evaluated first and its result used as the first operand to the addition (+) operator:

$$8 + 3$$

The result would be 11. Alternatively, the addition operator could be evaluated first and its result used as the second operand to the multiplication operator:

$$2 * 7$$

The result would be 14 instead. Clearly, rules that dictate how formulas are evaluated are needed to avoid confusion.

Informed uses three rules when it evaluates a complex formula. The precedence rules are based on:

- parentheses
- operator precedence
- left-to-right precedence

Before any operators are evaluated, Informed calculates all functions in a formula and uses the function result values as operands.

When you enclose part of a complex formula in parentheses, that part of the formula is evaluated first. If you nest parentheses by enclosing one set within another, the part of the formula in the “innermost” set of parentheses is evaluated first. The following example demonstrates how parentheses affect precedence.

$((3 + 5) * (6 - 3)) * 2$	innermost parentheses first
$(8 * 3) * 2$	outermost parentheses next
$24 * 2$	evaluate last operator
48	final result

Operator precedence is used when you don’t include parentheses to explicitly indicate the order in which a formula’s operators should be evaluated. Operator precedence specifies the order in which different operators in the same formula are evaluated. Operators with higher precedence are evaluated first. The following table shows Informed’s operators grouped into precedence levels. The highest precedence operators come first.

Informed Operator Precedence Levels

Operators	Precedence
% Not - (negation)	highest
^	
* / Div Mod	
+ - &	
= ≠ <> < ≤ ≤= > ≥ >=	
And	
Or	lowest

The following example shows how operator precedence affects the order of evaluation of the operators in a formula.

<code>4 * 5 + 3 &lt; 4 - 6 / 3</code>	evaluate multiplication and division first
<code>20 + 3 &lt; 4 - 2</code>	evaluate addition and subtraction next
<code>23 &lt; 2</code>	evaluate comparison operator last
<code>False</code>	final result

If you create a formula that uses operators of the same precedence level and no parentheses to explicitly alter the order of evaluation, the operators with the same precedence are evaluated left to right. The following examples shows how left-to-right precedence affects the evaluation of a formula.

<code>5 * 2 Div 3 / 1.5</code>	evaluate each operator from left to right
<code>10 Div 3 / 1.5</code>	
<code>3 / 1.5</code>	
<code>2</code>	

When you enter a complex formula, you can include parentheses and any valid combination of operators and operands. The following example shows a complex formula and how Informed evaluates it using all precedence rules.

<code>10 + 2 * ((5 + -2) / 3) - 5 = 8</code>	unary negation first
<code>10 + 2 * ((5 - 2) / 3) - 5 = 8</code>	innermost parentheses
<code>10 + 2 * (3 / 3) - 5 = 8</code>	outermost parentheses
<code>10 + 2 * 1 - 5 = 8</code>	multiplication
<code>10 + 2 - 5 = 8</code>	addition and subtraction
<code>7 = 8</code>	comparison last
<code>False</code>	final result

When operator or left-to-right precedence causes your formula to be evaluated incorrectly, you have to use parentheses to override the other precedence rules. For example, the formula below is intended to subtract from 50, the number of days between May 1, 1996 and April 25, 1996.

```
50 - ToDate ("May 1, 1996") - ToDate ("Apr 25, 1996")
```

However, left-to-right precedence causes the leftmost subtraction operator to be evaluated first (which attempts to subtract a date from a number). You could correct the formula either by switching the order of operands, or by using parentheses.

```
ToDate ("Apr 25, 1996") - ToDate ("May 1, 1996") + 50
50 - (ToDate ("May 1, 1996") - ToDate ("Apr 25, 1996"))
```

In either case, the subtraction operator subtracts one date from the other before adding the value 50. The correct result is returned.

## Formula Result

Every formula returns a result of a particular type. A formula's result type depends on the operators, operands, and functions used in the formula.

The result type of a formula should match the type expected according to where the formula is used. For example, if a formula sets the value of a cell, the formula's result type should match the type of the cell, whereas a check formula should always return a boolean result.

If the result type of a formula is different than the type required, Informed will attempt to automatically convert the result value to the correct type. For example, if the calculation formula for a text cell returns a numeric value, Informed will automatically convert the number result to its textual representation when the formula is evaluated. If you want to explicitly change the result type of a formula from one type to another, use the type conversion functions (ToText, ToNumber, and so on). See "Type Conversion" for more information.

When you create a calculation, check, or tab formula, you can use the 'Return' statement to explicitly indicate the formula's return value. The 'Return' statement is optional; it allows you to create more readable formulas. For example, the two calculation formulas below are equivalent.

```
3 * Mean (Cell1, Cell2)
Return 3 * Mean (Cell1, Cell2)
```

The 'Return' statement is particularly useful with the 'If' statement. Using 'Return' makes it clear which parts of the 'If' statement are potential return values. The following 'If' statement uses several 'Return' statements.

```
If Age < 18 Then
    Return Price * 0.5
ElseIf (Age >= 18) AND (Age < 45) Then
    Return Price
ElseIf (Age >= 45) AND (Age < 65) Then
    Return Price * 1.5
Else
    Return Price * 1.75
End
```

The 'Return' statements make it clear that the above formula can return one of four different results. Since the 'Return' statement is optional, the following formula is equivalent to the one above.

```
If Age < 18 Then
    Price * 0.5
ElseIf (Age >= 18) AND (Age < 45) Then
    Price
ElseIf (Age >= 45) AND (Age < 65) Then
    Price * 1.5
Else
    Price * 1.75
End
```

## Type Conversion

A value can be one of the following eight types: text, number, date, time, name, boolean, picture, and signature. The type of a value determines the type of information that the value can represent. For example, number values store numbers, whereas date values store dates. For a complete explanation of each type, see the “Cell Types” section in Chapter 1, “Adding Intelligence to Your Forms.”

### Note

When you use the Format command to select a cell’s type, you choose from nine different types. The discussion about type conversion involves only eight types. This is because the Text and Character cells both use the text type.

*Type conversion* refers to the process of changing a value from one type to another. Type conversion allows you to represent the same information in different forms. For example, you could convert a number value to a text value so that you could manipulate the value using Informed’s text functions.

Suppose that your form contains a name cell called ‘Customer Name’ and a text cell called ‘Comment.’ Let’s say that you want to create a calculation formula for the ‘Comment’ cell that combines the text message “Have a nice day” with the customer’s name. Since ‘Customer Name’ is a name cell, the cell’s value has to be converted to a text value before you can combine it with the text message.

Type conversion is required in the following situations:

- When the result type of a calculation formula doesn’t match the type of the cell that it sets.
- When the result type of a check formula is not boolean.
- When the result type of a tab formula is not text or number.
- When the type of an operand doesn’t match the type required by the operator with which it’s used.
- When the type of a function parameter doesn’t match the type expected by the function.
- When the result type of an ‘If’ or ‘ElseIf’ statement’s condition is not boolean.
- When you change the type of a cell on a form and use it to view old data.

You can explicitly change the type of a value using Informed's type conversion functions. For convenience reasons, however, Informed will automatically convert a value from one type to another in certain situations. The following section explains how Informed converts values.

## Type compatibility

Informed can convert values only between certain types. For example, a number can be converted to a text value, but a date can't be converted to a number. If a particular type can be converted to another type, the two types are said to be *compatible*. The following table shows which types are compatible. The types labelled on those rows and columns that intersect with a check mark are compatible.

Compatible Types

Cell Type	text	number	name	date	time	boolean	picture	signature
from <b>text</b> to	√	√	√	√	√	√	√	√
from <b>number</b> to	√	√		√	√	√		
from <b>name</b> to	√		√					
from <b>date</b> to	√			√				
from <b>time</b> to	√				√			
from <b>boolean</b> to	√	√				√		
from <b>picture</b> to	√						√	
from <b>signature</b> to	√							√

All types are compatible with themselves and all types are compatible with the text type.

The fact that two types are compatible doesn't mean that all possible values will convert from one type to the other. For example, even though the text and number data types are compatible, not all text values will convert to numbers. Although text values such as '123,' '34.9328,' and '-15.50' will convert to their numeric equivalents, the values 'abc' and '123z' won't because they don't represent valid numbers.

Informed uses special rules when it converts values between the text and boolean types. The boolean values True and False are converted to the text values "True" and "False." The following table lists the text values that will convert to boolean values.

Converting Text Values to Boolean Values

Text Value	Converts To
"True"	True
"T"	True
"False"	False
"F"	False
"Yes"	True
"Y"	True



## Converting Text Values to Boolean Values (continued)

Text Value	Converts To
“No”	False
“N”	False
“On”	True
“Off”	False

When a text value is compared with those values in the previous table, upper and lower case is ignored. This means, for example, that the text values “true” and “TRUE” will also be equivalent to the boolean value True.

When Informed converts a numeric value to a boolean value, the resulting boolean value will be True if the numeric value is non-zero, and False otherwise. The boolean values True and False convert respectively to the text values “True” and “False,” and to the numeric values 1 and 0.

When a date or time value is converted to a text value, the textual date or time value will format in a default manner. Date values will assume the format ‘M/D/YY,’ whereas time values will format according to ‘H:MM:SS AM.’ For more information about date and time formats, see the “Cell Types” section in Chapter 1, “Adding Intelligence to Your Forms.”

Both pictures and signatures are binary values. Binary values are values that don’t normally display as a series of characters. Informed understands how to display pictures and signatures in an appropriate representation (that is, a picture appears as a picture, whereas a signature displays as the signer’s name next to a signature icon). When Informed converts a picture or signature to text, it converts the binary information to a series of characters. It is useful to be able to convert pictures and signatures to text (and back) if you want to export or submit form data to a database, or through a medium that does not support binary information.

If you attempt to convert a value between two incompatible types, or if the conversion of a value between two compatible types fails, Informed will change the value to the empty value. For example, suppose that the cell called ‘Purchase Date’ is a date cell. The result of each formula below would be the empty value.

```
5 * Purchase Date
15 / 'ABCDEFG'
```

In the first formula, the type of the cell ‘Purchase Date’ is incompatible with the multiplication operator. In the second formula, the text value “ABCDEFG” can’t be converted to a valid number.

## Automatic Type Conversion

For convenience reasons, Informed will automatically convert a value from one type to another when it’s clear that the value must be a particular type. For example, if the result of a calculation formula doesn’t match the type of the cell that it sets, Informed will automatically convert the value to the correct type. The type conversion rules explained in the previous section apply as expected.

Automatic type conversion allows you to create formulas with more freedom. You don't have to tediously convert values from type to type in most common situations. For example, both formulas below return the numeric result 150.

```
3 * ToNumber ("50")
3 * "50"
```

Since the multiplication operator multiplies numbers only, Informed will automatically convert both operand values—if necessary—to numbers.

Suppose that a date cell called 'Ship Date' contains the value September 12, 1996. The formula below uses the concatenation operator to combine the value of this cell with a text message.

```
"Your order will ship on " & Ship Date & "."
```

The value of 'Ship Date' is automatically converted to the text value "9/12/96" and then concatenated to produce the result "Your order will ship on 9/12/96."

Some operators and function parameters allow values of different types. For example, the subtraction operator can subtract numbers, dates, or times, and the 'Choose' function can select a value of any type. To ensure that your formulas calculate correct results in these situations, Informed allows you to use special type conversion functions. These functions are explained in the following section.

## Type Conversion Functions

When you create a formula, you can explicitly convert a value from one type to another. Informed provides eight type conversion functions for converting values between compatible types. The following table lists these functions.

Type Conversion Functions

Function	Description
ToText	Converts any text-compatible value to text
ToNumber	Converts any number-compatible value to a number
ToName	Converts any name-compatible value to a name
ToDate	Converts any date-compatible value to a date
ToTime	Converts any time-compatible value to a time value
ToBoolean	Converts any boolean-compatible value to a boolean value
ToPicture	Converts any ASCII represented picture to a picture value
ToSignature	Converts any ASCII represented signature to a signature value

The single parameter to any of these functions can be any value that's compatible with the destination type. For example, the formula below converts a text value to the date value April 5, 1996.

```
ToDate ("April 5, 1996")
```

You should use the type conversion functions when the type of a value can be interpreted differently. Some operators and function parameters allow values of different types. For example, the comparison operators compare values of any type, as long as the type of each operand is the same. If you want to compare two differently typed values according to the comparison rules of a particular type, you should use the type conversion functions to convert both operands—if necessary—to the correct type. The formula below compares the value of the time cell 'Time In' with the time constant 8:00 AM.

```
If Time In <= ToTime ("8:00 AM") Then
    Return True
Else
    Return False
End
```

If you were to compare the value in 'Time In' with the text constant "8:00 AM" instead, Informed would convert the value in 'Time In' to text and then compare two text values. By using the ToTime function, both operands are interpreted as time values. Informed therefore uses the rules for comparing time values to compare the two operands.



# 11

## Using Informed Number Server

In this chapter:

- Overview 11-2
- Configuring Informed Number Server 11-3

# 11

## Using Informed Number Server

As discussed in “Auto-incrementing Numbers” in Chapter 1 of this manual, there are several ways that you can generate unique identification numbers such as invoice numbers, time sheet numbers, and purchase order numbers. You can configure an auto-incrementing cell to get its next available number from the form template itself, from an Apple event aware application (Mac OS only), or from an external database or data source accessible through Informed’s data access plug-ins.

One of the Apple event aware applications that you can link an auto-incrementing cell to is Informed Number Server. An *Apple event aware* application is one that can send and receive messages using the Apple event capability of the Mac OS. Furthermore, Informed Number Server understands the specific Apple events that Informed Filler uses to communicate.

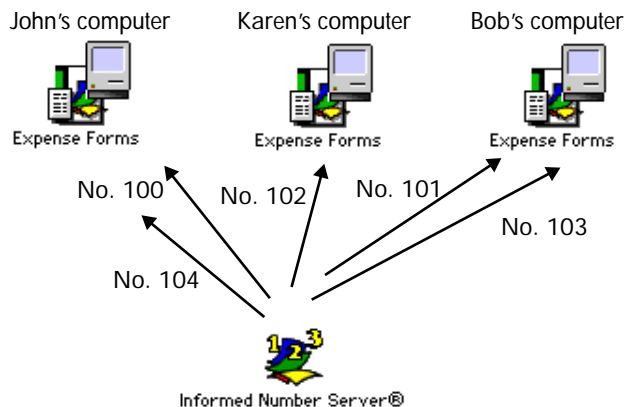
Informed Number Server is included with Informed Designer, but it is a completely separate application, and as such, it must be installed and configured separately.

This chapter deals specifically with how to configure Informed Number Server for use with forms filled out with Informed Filler. For information on installing Informed Number Server, see your *Informed Designer Getting Started Guide*. For information on how to link forms to Informed Number Server, see “Auto-incrementing Numbers” in Chapter 1 of this manual.

### Overview

Form numbers, such as invoice numbers and expense authorization numbers, are important because they ensure that all forms are uniquely identified. This is helpful for auditing purposes.

Informed Number Server has been designed to automate the assignment of form numbers to multiple users. Since all forms of a particular type obtain new form numbers from the same Informed Number Server application, numbers are assigned consecutively and duplicates are avoided.



In order to use Informed Number Server, all users must be running System 7 or later on their computers. System 7 brings IAC (Inter-Application Communications) capabilities to the Mac OS. These capabilities allow two different applications—either on the same computer, or on two different computers connected to the same network—to communicate directly with each other. Messages and information can be sent from one application to another. Informed Number Server uses Apple events to send new form numbers to the users filling out forms with Informed Filler.

Informed Number Server requires that one Mac OS computer act as a server. This can be any computer that's connected to the network of users filling out forms. It can be a computer that acts as a dedicated server for other applications such as electronic mail or file sharing, or it can be a user's computer. The Mac OS computer acting as a server must have program linking turned on.

Once you've installed Informed Number Server, as the administrator you'll configure the application for each of the form templates that you intend to administer form numbers to. For example, if people fill out invoices and purchase orders, you'll create two form numbers, one for each type of form. The form number cells are then linked to Informed Number Server using Informed Designer.

## Configuring Informed Number Server

Like most server type applications, Informed Number Server requires an administrator to configure the server for use on the network. This is a simple process that involves specifying which form numbers Informed Number Server will assign and their starting values.

Like networks and file servers, configuration and maintenance of Informed Number Server should be the responsibility of an administrator. This person will configure Informed Number Server and ensure its correct operation. An administration password ensures that only the administrator has the ability to change the configuration. Once Informed Number Server has been configured, the application will run virtually maintenance-free. You may, however, want to change its configuration from time to time.

## Registering Informed Number Server

When you run Informed Number Server for the first time, you'll be asked to enter your name and the name of your organization (if applicable).

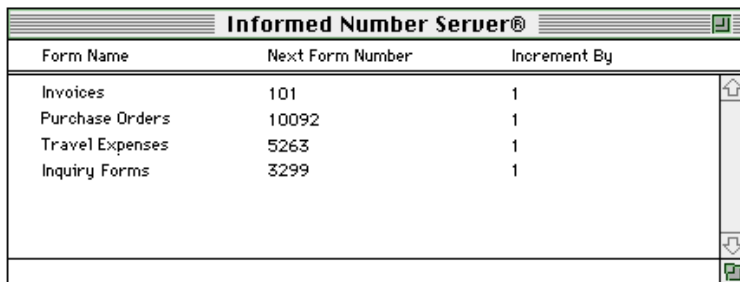


A registration dialog box for Informed Number Server. It features a small logo with the numbers 1, 2, and 3 in the top left corner. The text reads: "Please personalize your copy of Informed Number Server®." Below this, there are two text input fields. The first is labeled "Name:" and contains the text "John Smith". The second is labeled "Company:" and contains the text "ABC Company". At the bottom of the dialog, there are two buttons: "Cancel" and "OK".

Once you've registered Informed Number Server, you'll see the two names on the welcome screen and the About dialog box.

## The Informed Number Server Window

When you run Informed Number Server, a window appears showing the current configuration. Each entry in the list corresponds to a form number. The name of the form, the next available form number value, and the increment value are shown.



Form Name	Next Form Number	Increment By
Invoices	101	1
Purchase Orders	10092	1
Travel Expenses	5263	1
Inquiry Forms	3299	1

As you add, change, and remove form numbers, the information in this window changes. Like most Mac OS windows, you can move or resize the window by clicking and dragging its title bar or size box, respectively.



## Number Server Data

When you run Informed Number Server for the first time, it automatically creates a file called ‘Number Server Data’ and stores it in the ‘Preferences’ folder inside the ‘System Folder’ on the computer’s hard disk. This file contains Informed Number Server’s configuration information.

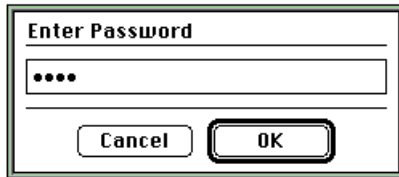
### Note

If you move Informed Number Server to a different computer, be sure to move the ‘Number Server Data’ file as well, and place it in the Preferences folder inside the System folder of the new Mac OS computer.

## Administration Capabilities

Before you can add, change, or remove form numbers, you must enable the administration capabilities of Informed Number Server. In order to do so you must know the administration password. If the password is blank, Informed Number Server will automatically invoke the administration capabilities when you start the application.

To enable or disable the administration capabilities, choose either **Enable Administration** or **Disable Administration**, respectively, from the File menu. If the administration password is not blank, you’ll be asked to enter it when you choose the Enable Administration command.



With the administration capabilities enabled, commands are available to quit Informed Number Server, change the administration password, and add, change, and remove form numbers.

## Changing the Administration Password

When you start Informed Number Server for the first time, the administration password will be blank. With administration capabilities enabled, you can change the password by choosing **Change Password...** from the File menu. If the current administration password is not blank, you’ll be asked to enter it first.



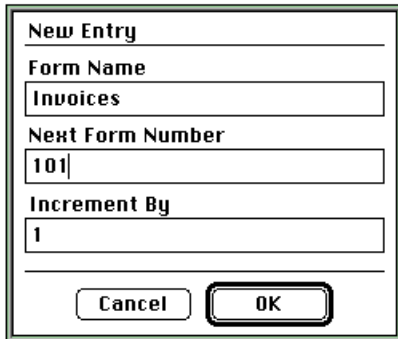
You are then asked to enter the new password in a dialog box similar to the one shown above. If you'd like to clear the password, simply leave it blank.

Once you've successfully changed the password, a message will appear for confirmation.

## Adding, Changing, and Removing Form Numbers

With administration capabilities enabled, commands in the Entry menu are available for adding, changing, and removing form numbers.

To add a new form number to the configuration, choose **New...** from the Entry menu. A dialog box appears requesting that you enter the form name, the next available form number value, and the increment value. The increment value determines the value that the form number is advanced by each time a number is assigned. A value of 1 is most common.



The image shows a dialog box titled "New Entry". It contains three text input fields. The first field is labeled "Form Name" and contains the text "Invoices". The second field is labeled "Next Form Number" and contains the text "101". The third field is labeled "Increment By" and contains the text "1". At the bottom of the dialog, there are two buttons: "Cancel" and "OK".

From time to time you may want to change the configuration of an existing form number. To do so, click the form number in the list on the Number Server window, then choose **Change...** from the Entry menu. A dialog box appears allowing you to change the next available number and the increment value.

To remove a form number, select it in the list on Informed Number Server window, then choose the Remove command from the Entry menu. You will be asked for confirmation before the form number is permanently removed.

## Important Precautions

Once you've configured Informed Number Server, cells on form templates can be linked to the application using Informed Designer. For information on how to link cells to Informed Number Server, see "Auto-incrementing Numbers" in Chapter 1, "Adding Intelligence to Your Forms."

After you link a form and distribute it for use with Informed Filler, it's important that you do not change:

- the name of the Informed Number Server application
- the name of the Mac OS computer that's running the Informed Number Server application
- the form numbers as specified in Informed Number Server's configuration

Informed Filler uses these names to find the Number Server application and obtain the correct form number each time a new form is filled out (or when the user explicitly requests a new number). If you change any of the above names, an error will occur when Informed Filler requests a new number. Information about possible errors and how to recover from them can be found in your *Informed Filler User's Manual*.

## Quitting the Number Server Application

Like most applications, you quit Informed Number Server by choosing **Quit** from the File menu. However, before you can quit, you must enable administration capabilities. See "Administration Capabilities" earlier in this chapter for more information.

11-8 : Using Informed Number Server  
:

# 12 Using AppleScript

In this chapter:

- Overview 12-2
- Entering and Editing Scripts 12-3
- Writing Scripts 12-5

# 12

## Using AppleScript

AppleScript is a scripting language that allows you to control Macintosh applications with program-like scripts. Rather than using the keyboard and mouse, you can write scripts to perform tasks. A task can be as simple as opening and printing a form, or as complex as controlling sophisticated workflow processes.

With Informed and AppleScript you can:

- automate tasks
- customize forms
- integrate Informed Filler with other applications.

### Note

Informed's AppleScript features are available only on Mac OS compatible computers.

This chapter provides an overview of Informed's scripting capabilities and describes in detail how you can customize forms using AppleScript. You'll learn specifically how to use Informed Designer's Scripts command to attach scripts to templates so that they can be configured to run when the Informed Filler user invokes certain actions.

To learn how to write scripts, you should first read the *AppleScript Language Guide* (included with the AppleScript Software Development Toolkit). You should then refer to "Writing Scripts" later in this chapter, for a description of the terminology that you use when writing scripts for Informed Filler.

## Overview

With Informed and AppleScript, a single script can automate a task that normally requires several manual steps. For example, you could write a script which searches for and prints all California invoices that exceed five hundred dollars. A different script could create a new purchase order form and fill it in with information from one or more purchase requisition forms. Performing such tasks becomes as simple as selecting a script

An important feature of AppleScript is its ability to integrate many different scriptable applications. By controlling different applications, a single script can effectively combine different features from different products to provide more powerful solutions. You could, for example, write a script which instructs Informed Filler to collect information from different forms, chart the information using a spreadsheet application, then insert the results into a letter using a word processor.

You write scripts using a script editor or an application such as Informed Designer. The script editor that comes with AppleScript stores scripts in script documents. The script in a script document can be played by double-clicking the document's icon.

Informed Designer can store scripts in form documents. That way, whenever you copy a form document to another place, or mail a form to another person, the scripts remain part of the form. Applications that can store scripts, such as Informed Designer and Informed Filler, are often called *attachable* applications. This is because scripts can be *attached* to particular actions in the application. When the user performs an action, the application triggers a script.

You configure forms with Informed Designer so that Informed Filler invokes scripts when the user performs certain actions. You can attach scripts to the following actions:

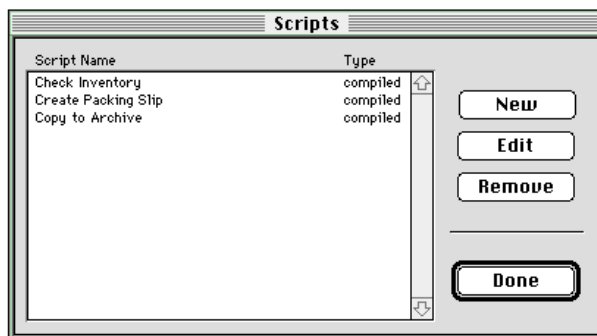
- selecting a menu item
- clicking a button
- typing a value in a lookup cell
- submitting a form

The details of configuring the actions listed above can be found in other sections of this manual. Before configuring an action, however, you must first enter the script using Informed Designer's Scripts command.

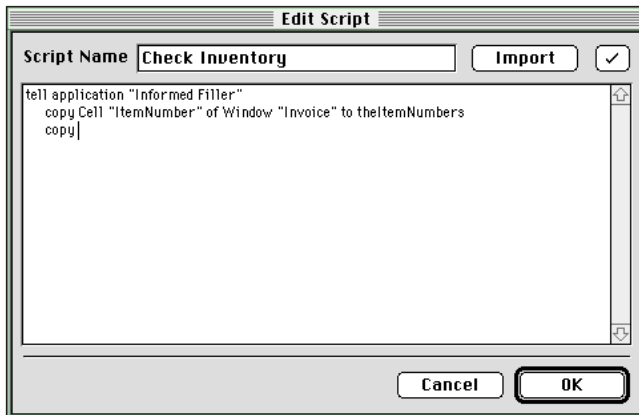
Chapter 1, "Adding Intelligence to Your Forms", describes Informed's data access features. You can configure forms to lookup and store information in other applications. For information about AppleScript lookups and form submission, please see Chapter 1. The remaining sections of this chapter explain how you can enter and edit scripts so that they appear in Informed Filler's Scripts menu.

## Entering and Editing Scripts

You use Informed Designer's Scripts command to add, remove, and edit scripts. Choosing this command displays the Scripts dialog box.

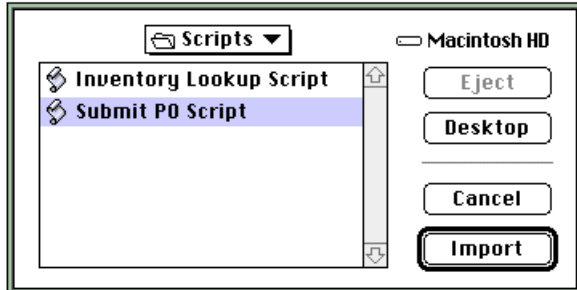


The scrolling list shows the names of the scripts currently attached to the form in the order that they were created. To add a new script, click 'New.' The Edit Script dialog box appears.



Enter the name of the script in the ‘Script Name’ text box. This is the name that you’ll see when you configure an action to invoke a script.

You can enter a script by typing in the text box, or by importing a script from another file. To import a script, click ‘Import.’ The standard Open dialog appears allowing you to select a file containing a script.



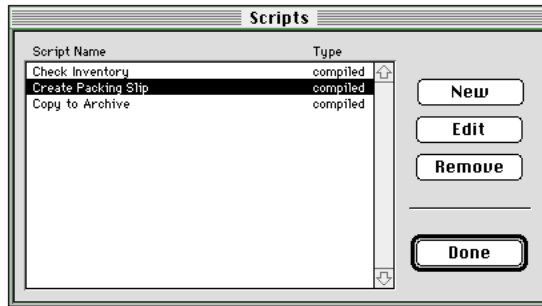
Find and select the file that contains the script that you want to import then click ‘Import.’ You can import a text or compiled script. The script in the selected file is read and displayed on the Edit Script dialog box.

After entering a script, you can check for errors by clicking the check mark button (✓). If an error is detected, you’ll see a message describing the error and Informed Designer will highlight the mistake in the script. If there are no errors, the script will re-display properly formatted.

When you click the check button, Informed Designer tells your computer to *compile* the script. Compiling is the process of converting the english-like commands into instructions that scriptable applications can understand. Scripting errors are detected during this process.



After entering the script, click ‘OK’ on the Edit Script dialog box. Informed Designer will store the compiled script with the form. Storing the script compiled means that Informed Filler will execute the script much faster. If the script contains any errors, you’ll see a dialog box indicating so, and Informed Designer will store the script in its text version. The Scripts dialog box indicates whether the script is stored in its compiled or text version.



To edit an existing script, select its name in the scrolling list, then click ‘Edit.’ You edit an existing script the same way you enter a new script (see above). To remove a script, select its name, then click ‘Remove.’

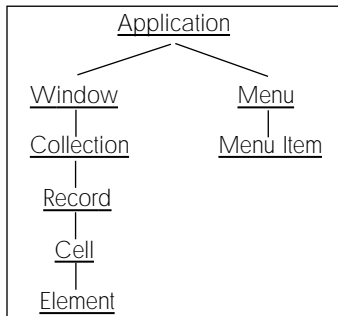
## Writing Scripts

This section explains how to use AppleScript to manipulate Informed Filler, and provides examples of the most common commands you will use. It is assumed that you already understand the basics of AppleScript and are familiar with Informed Designer and Informed Filler.

### Objects, Properties, and Containment

An object is a representation of some thing that you can manipulate in an application, such as a window or a collection of records. Properties are attributes of objects, such as a window’s name or size. Using AppleScript to communicate with an application can be thought of as a conversation with various objects in that application. For example, to get the value of a cell, you would ask the cell to respond with its “value” property.

Objects can contain other objects. For example, a “window” object can contain things such as collections of records, individual records, or cells. Object containment follows a strict hierarchy, from the most general object type (the application) to the most specific. The object containment hierarchy in Informed is as follows:



To refer to a specific object, such as the cell named “Company,” you specify each object in the containment hierarchy. Here are some examples of how to specify objects at different levels of the containment hierarchy.

```
tell application "Informed Filler™"
    index of window "Sample Form"
end tell
result: 1
```

```
tell application "Informed Filler™"
    every cell of first record of current collection of window "Sample Form"
end tell
result: {"John Smith", "World Corporation", "ext. 4-5678"}
```

```
tell application "Informed Filler™"
    cell "Company" of current record of window "Sample Form"
end tell
result: "World Corporation"
```

To refer to a specific object, you would normally be required to specify the complete containment hierarchy. That is, to specify a cell, you also must specify the record, collection, window, and application. Complete specifications can be quite long, and typing them would quickly become tiresome. Fortunately, Informed Filler provides a “default containment” that will help shorten your typing. Most of the outer objects in a complete object specification can be left out, and appropriate defaults will be provided.

The following table lists the default objects that Informed uses.

#### Default Containment

Class	Default Object
window	window 1, or the frontmost window
collection	the current collection
record	the current record
cell	none
element	none
menu	none
menu item	none

Notice that cell, element, menu, and menu item have no default objects. In any reference to an object or property that includes one of these classes, you must explicitly state which object or objects of each of these classes you are referring to. For example, to refer to the name of a menu item, you must provide the menu item object and its containing menu object. To refer to the value of an element, you must provide the element and its containing cell. Window, collection, and record have defaults that will be supplied, although you can still specify those objects as well, if the ones you want are different from the defaults.

Here are some examples of object references that use default containers.

```
tell application "Informed Filler™"
  -- use default collection and window:
  record 1

  -- default collection and window
  every record

  -- default record, collection, and window
  cell "Company"

  -- default record, collection, and window
  current element of cell "Company"

  -- default collection and window
  current cell of record 1
end tell
```

Note that “current element” and “current cell” are properties of an object, and so you must specify the object to which they belong. Properties have no default container. Once you have provided an object at the cell level or higher, you can rely on the default containment to complete the object specification.

The following sections introduce each kind of object in Informed Filler, and briefly describes the scope of each.

## Application

The application object represents the Informed Filler application. By talking to the application object, you can get information like its name, whether it is the frontmost application, who the application is registered to, and so on.

The following example script tells the Informed Filler application to make itself the front-most application.

```
set frontmost of application "Informed Filler™" to true
```

This next script asks Informed Filler who the registered user of the application is, and then displays a dialog with the user's name and company name.

```
tell application "Informed Filler™"
    set theName to registered name
    set theComp to registered company
end tell
display dialog "Informed Filler™ is registered to " & theName & ", of " & ¬
    theComp & "."
```

The properties of the Informed Filler application are listed in the table on the following page.

### Application Properties

Property	Writeable?	Description
frontmost	yes	Is this the frontmost application?
name	no	The name of the application.
registered company	no	The company name that this copy of Informed Filler is registered to.
registered name	no	The user name that this copy of Informed Filler is registered to.
serial number	no	The serial number of this copy of Informed Filler.
version	no	The version number of Informed Filler.
properties	no	All of the properties of the application, returned as an AppleScript record.

## Window

The window object represents each document window that is currently open in Informed Filler. Record lists, the choices palette, and other secondary windows are not included. You can ask a window for information about the document it is displaying, such as its name, where on the disk it is stored, who the author of that document is, and what data it contains.

The following sample script gets the name of every window, and then moves the second window to the front.

```
tell application "Informed Filler™"
    set theNames to name of every window
    set index of window 2 to 1
    theNames
end tell
result: {"Expense Report", "Price List", "Holiday Request"}
```

Here is a more complicated example. This script gets the employee name and address from a bunch of address change forms, and updates the address in an employee database.

```
tell application "Informed Filler™"
    set changeCount to number of records in window "Change Form"
    repeat with n from 1 to changeCount
        set empName to cell "Name" of record n of window "Change Form"
        set empAddr to cell "Address" of record n of window "Change Form"
        set (cell "Address" of (first record whose cell "Name" = empName) of -
            window "Employee DB") to empAddr
    end repeat
end tell
```

The following table lists the properties of windows:

#### Window Properties

Property	Writeable?	Description
author name	no	The name of the author of this template, as entered in the Template Information dialog box.
author organization	no	The name of the template author's organization, as entered in the Template Information dialog box.
bounds	no	The bounds of the window, as a list of integers. The values returned are {left, top, right, bottom}.
closeable	no	Is the window closeable? (always returns true)
current cell	yes	The current cell of the form. If there is no current cell, the result will be empty. The default behavior is to return the value of the current cell. You may also request a reference to the current cell.
current collection	yes	The current collection of records.
current record	yes	The current record.
description	no	The description of the template and its use, as entered in the Template Information dialog box.
disk file	no	The disk file that contains the window's document.
TemplateID	no	The TemplateID of this template, as entered in the Template Information dialog box.

## Window Properties (continued)

Property	Writeable?	Description
ID	no	The unique ID of the data document. Valid only until the data document is closed.
index	yes	The front-to-back index of the window, relative to the other document windows.
floating	no	Answers whether or not the window floats. Always returns True.
modal	no	Answers whether or not the window is modal. Always returns False.
name	no	The name of the window.
resizable	no	Answers whether or not the window is resizable. Always returns True.
revision	no	The revision number of the template, as entered in the Template Information dialog box.
titled	no	Answers whether or not the window has a title. Always returns True.
visible	no	Answers whether or not the window is visible. Always returns True.
zoomable	no	Answers whether or not the window is zoomable. Always returns True.
zoomed	yes	Answers whether or not the window is zoomed.
properties	no	All of the properties of the window, returned as an AppleScript record.

Windows may be accessed with the methods shown in the following table.

## Accessing Windows

Access Method	Example
by name	window "Expense Form"
by index	window 1, every window, first window
by range	windows 1 thru 2
special	front window
by tests	every window whose name contains "Expense"

## Collection

The collection object represents a collection of records, usually the result of a find operation the user has performed. Each window has a current collection, which is the set of records that are currently being displayed.

The current collection in a window may be set to a specific set of records. This is equivalent to performing a find operation. The following script sets the current collection to the first three records in the document.

```
tell application "Informed Filler™"
    set current collection of window 1 to record 1 thru 3 of window 1
end tell
```

You can also perform a find all operation by setting the current collection to every record.

```
tell application "Informed Filler™"
    set current collection of window 1 to every record of window 1
end tell
```

Simple or complex find operations can be performed by using the “whose” form of referring to records.

```
tell application "Informed Filler™"
    -- find records matching one criterion
    set current collection of window 1 to (every record whose cell ↵
        "Employee" = "Fred") of window 1

    -- find records matching 2 criteria
    set current collection of window 1 to (every record whose cell ↵
        "Salary" = "56000" and cell "Name" = "John Smith") of window 1
end tell
```

A collection has very little information in and of itself. The principle value of a collection is in accessing the records that it contains.

The following table lists the properties of collections:

Collection Properties

Property	Writeable?	Description
container	no	A reference to the container of the collection (the collection’s window).
ID	no	The unique ID of the collection. Valid only until the data document is closed.
properties	no	All of the properties of the collection, returned as an AppleScript record.

Collections can be accessed with the methods shown in the following table.

#### Accessing Collections

Access Method	Example
by index	collection 1 of window 1 every collection of window 1
current collection property	current collection of window 1

## Record

The record object represents a single record in a form data document. Each window has a current record, which is the record currently being displayed to the user.

Records can be created, duplicated, deleted, printed, and so on. This script shows examples of record manipulation:

```
tell application "Informed Filler"
  -- make a new record
  Make new record at window 1

  -- Set some cells in the record
  set cell "Name" of current record of window 1 to "Fred"
  set cell "Start Date" of current record of window 1 to "10/12/93"

  -- Duplicate the current record
  duplicate current record of window 1

  -- Remove all the records in the collection
  Delete every record of current collection of window 1
end tell
```

The following table lists the properties of a record.

#### Record Properties

Property	Writeable?	Description
container	no	A reference to the container of the record (the record's window).
ID	no	The unique ID of the record. The ID property of a record is persistent, and will always be valid until the record is deleted.
selected	yes	Returns whether or not a record is selected in the Record List.
properties	no	All of the properties of the record, returned as an AppleScript record.



Forms may be accessed with the following methods:

#### Accessing Records

Access Method	Example
by index	record 1 of window 1 last record of window 1 every record of window 1
relative to other records	record after record 1 of window 1 record before current record of window 1
by range	records 1 thru 5 of window 1
current record property	current record of window 1
by tests	(every record whose cell "Name" = "Ed") of window 1 (every record whose cell "Salary" < 35000 and cell "Supervisor" = "Ed") of window 1 (every record whose selected = True) of window 1

## Cell

The cell object represents a single data item in a record. Cells include single value fields (drawn with the Field tool in Informed Designer), and multiple value fields (drawn with the Table tool in Informed Designer).

There is a current cell in each window, representing the cell that is currently being edited. If there is no currently edited data in the window, the current cell property will return an empty value.

The most important property of a cell is the value of the cell, or the data it contains. References to a cell will by default refer to the value property. That is, if you simply request the data for a cell, you are by default requesting the value property of that cell.

The following two scripts will therefore return the same information.

```
tell application "Informed Filler™"
    value of cell "Name" of window 1
end tell
result: "Fred"
```

```
tell application "Informed Filler™"
    cell "Name" of window 1
end tell
result: "Fred"
```

You can also request a cell as reference, which will return a reference to the cell in the most efficient form for that class.

```
tell application "Informed Filler™"
    cell "Name" of window 1 as reference
end tell
result: cell id 106 of window id 1 of application "Informed Filler™"
```

Cells that have multiple values will return every value in the cell, in a list.

```
tell application "Informed Filler™"
    cell "Qty" of window 1
end tell
result: {2, 5, 1, "", "", "", "", ""}
```

Every value is returned. Elements that are not filled in are returned as blank strings.

The following table lists the properties of a cell.

#### Cell Properties

Property	Writeable?	Description
main choices/ extra choices	no yes	The list of main/extra choices for the cell. These choices will appear in the choices palette when the cell is active.
container	no	A reference to the container of the cell (the cell's window).
current element	yes	A reference to the current element of the cell. This property is only valid when the cell is active.
data type	no	The default data type for data entered in the cell.
display only	no	Returns whether or not the cell is display only, as set in the Value dialog box.
ID	no	The unique ID of the cell. The ID property of a cell is persistent, and will always be valid unless the cell is deleted.
index	no	The index of the cell. This is equivalent to the cell's tab order. Each column cell in a table has a sequential index.
name	no	The name of the cell.
signed	no	Returns whether or not the data in the cell has been signed by a signature cell. This returns True only when there is an actual signature.
table ID	no	The ID of the table that encloses this cell. This is valid only for column cells. This may be used to identify which other column cells are part of the same table.
value	yes	The value of the cell.
writeable	yes	This property is by default always set to True, but you may set it to False to prevent the user from entering data in the cell. This property is initialized to True for all cells each time a form is opened.
properties	no	All of the properties of the cell, returned as an AppleScript record.

Cells may be accessed with the methods shown in the following table.

Accessing Cells	
Access Method	Example
by name	cell "Name" of window 1
by index	cell 1 first cell every cell
relative to other cells	cell after cell 1 cell before current cell of window 1
by range	cells 1 thru 5
current cell property	current cell of window 1

## Element

The element object represents a single piece of information in a cell. Single value cells, drawn with Informed Designer's Field tool, will have only one element. Column cells, drawn with Informed Designer's Table tool, will have one element for each row in the table.

References to the value of a column cell return the values for every element of the cell, in a list. Use the element class to refer to a single row of the column.

```
tell application "Informed Filler™"
    cell "Qty" of window 1
end tell
result: {2, 5, 1, "", "", "", "", ""}
```

```
tell application "Informed Filler™"
    element 2 of cell "Qty" of window 1
end tell
result: 5
```

A cell has a "current element" property that can be used to identify which row is currently being edited. In the following example, assume that the user is currently editing the second row of the "Qty" cell.

```
tell application "Informed Filler™"
    current element of current cell of window 1 as reference
end tell
result: element 2 of cell id 131 of window id 6 of application "Informed Filler™"
```

You can find out which row is being edited by requesting the index property of the current element.

```
tell application "Informed Filler™"
    index of current element of current cell of window 1
end tell
result: 2
```

This might be used to perform some operation on the same row of the other cells in a table.

The following table lists the properties of an element.

#### Element Properties

Property	Writeable?	Description
container	no	A reference to the container of the element (the element's cell).
data type	no	The default data type for data entered in the element.
index	no	The index of the element. This is the row position of the element within its enclosing cell.
value	yes	The value of the element.
properties	no	All of the properties of the element, returned as an AppleScript record.

Elements may be accessed with the methods shown in the following table.

#### Accessing Elements

Access Method	Example
by index	element 1 of cell 1 first element of cell 1 every element of cell 1
by range	elements 1 thru 5 of current cell of window 1
current element property	current element of cell 1 current element of current cell of window 1

## Menu

The menu class represents all of the menus of the application. This includes the menus that appear in the menu bar, and all sub menus. Menus are primarily used to access the menu items contained therein.

The following table lists the properties of a menu:

#### Menu Properties

Property	Writeable?	Description
enabled	no	Returns whether or not the menu is enabled.
index	no	Returns the index of the menu.
name	no	Returns the name of the menu.
properties	no	All of the properties of the menu, returned as an AppleScript record.

Menus may be accessed with the methods shown in the following table.

#### Accessing Menus

Access Method	Example
by name	menu "File"
by index	menu 1 first menu every menu
by range	menus 1 thru 5

## Menu Item

The menu item class represents a single item in a menu. Menu items can be executed with the “execute” event, which performs the same action as if the user had selected that menu item.

Menu items may be used to script a task that is not easily scripted with the other objects. To execute a command from a menu, use the “execute” event.

```
tell application "Informed Filler™"
    execute menu item "Submit" of menu "File"
end tell
```

You can also test if a menu item is enabled before executing it:

```
tell application "Informed Filler™"
    if menu item "Submit" of menu "File" is enabled
    then
        execute menu item "Submit" of menu "File"
    else
        display dialog "The Submit menu was not enabled."
    end if
end tell
```

The following table lists the properties of a menu item.

#### Menu Item Properties

Property	Writeable?	Description
enabled	no	Returns whether or not the menu item is enabled.
index	no	Returns the index of the menu item within its containing menu. Some menu items may change their index depending on what plug-ins are installed.
name	no	Returns the name of the menu item.
properties	no	All of the properties of the menu item, returned as an AppleScript record.

Menu items may be accessed with the methods shown in the following table.

#### Accessing Menu Items

Access Method	Example
by name	menu item "Close" of menu "File"
by index	menu item 1 of menu "Size" first menu item of menu "Size" every menu item of menu "Size"
by range	menu items 1 thru 5 of menu "Size"

## Getting and Setting Data

The contents or value of any property or object can be accessed by simply referring to that property. AppleScript will send a Get Data command to Informed Filler to request the data for that property or object. The value will be returned as the default type for that property or object.

For example, to get the value of the cell named “Employee” and store it in an AppleScript variable “emp,” you would write the following:

```
tell application "Informed Filler™"
    set emp to cell "Employee"
end tell
```

Note that this very short script implies, through the default containment, that the requested data is the value property of the cell named “Employee” of the current record of the front window.

The contents of any writeable property or object can be set by using either the “set” or “copy” commands. The following script gets the value of the cell "Counter," adds one, and then sets it back.

```
tell application "Informed Filler™"
    set theCount to cell "Counter"
    set theCount to theCount + 1
    set cell "Counter" to theCount
end tell
```

## Documents

This section shows how you can use AppleScript to open and close a document, quit Informed Filler, and test if an object exists.

### Opening a Document

To open a document, you use the “open” command with the file to open.

```
tell application "Informed Filler™"
    open file "HD:Test Form"
end tell
```

You can also use an alias to specify the file to open.

```
tell application "Informed Filler™"
  open alias "HD:Test Form"
end tell
```

### Note

Referring to a file as “alias "filename"” saves an alias to the file in the compiled script, as if you made an alias in the Finder. Referring to a file as “file "filename"” just saves the "filename" string. Using an alias will successfully locate the file in many more situations, such as when the file name changes, or it is moved to another folder. There are still many situations where an alias will not resolve, so you should be prepared to handle errors.

## Closing a Document

To close a document, use the “close” command with the window to close.

```
tell application "Informed Filler™"
  -- close the front window:
  close window 1

  -- close a specific document:
  close window "Test Form"

  -- close every document:
  close every window
end tell
```

## Quitting Informed Filler

To quit the application, you simply tell Informed Filler to quit.

```
tell application "Informed Filler™"
  quit
end tell
```

## Checking the Existence of an Object

Sometimes, you may need to test if some object exists. For example, you might want to test if a window is already opened before you tell Informed Filler to open it. To test if an object exists, use the “exists” command with a specifier for the object you want to test.

```
tell application "Informed Filler™"
  if not (exists window "Test Form") then
    Open file "HD:Test Form"
  end if
end tell
```

You can also test for an object's existence by trying to access the object, and catching any error in a try statement.

```
tell application "Informed Filler"
  try
    name of window "Test Form"
  on error
    open file "HD:Test Form"
  end try
end tell
```

### Note

The “try” statement asks AppleScript to try to perform the following commands. If an error occurs, AppleScript will resume execution of the script in the “on error” statements. This structure allows you to trap errors and perform intelligent error recovery

## Working With Records

This section describes how you can use AppleScript to work with individual records, and records in a collection.

### Setting the Current Collection

The current collection property of a window can be set to any record or group of records. The current collection can be thought of as a list of references to record objects. When you set the current collection to a different list of record references, you are affecting the records that will be displayed to the user. Setting the current collection to one or more records is equivalent to performing a find operation in Informed Filler that matches that set of records.

```
tell application "Informed Filler"
  -- find a single record
  set current collection of window 1 to record 1

  -- display all records:
  set current collection of window 1 to every record

  -- find some specific records
  set current collection of window 1 to (every record whose cell "Salary" ~
    > 35000) of window 1

  -- narrow an existing search:
  set current collection of window 1 to (every record whose cell "Supervisor" ~
    = "Fred") of current collection of window 1
end tell
```



## Omitting a Record from the Collection

Records can be omitted from the current collection with the “omit” command.

```
tell application "Informed Filler™"
  -- omit one record:
  omit record 1

  -- omit the current record:
  omit current record of window 1

  -- omit some specific records:
  omit (every record whose cell "City" = "Calgary") of current collection ~
    of window 1
end tell
```

Note that when omitting records, you should only try to omit records that are already in the current collection. If you try to omit a record that is not in the current collection, an error will result.

## Looping Through Each Record

Many tasks will require you to perform the same function on every record in the current collection. There are several approaches to doing this.

One approach is to count the number of records in the collection, and then loop through each record. Here is a sample script that totals up a bunch of invoice records.

```
tell application "Informed Filler™"
  set n to number of records in current collection of window 1
  set theTotal to 0
  repeat with i from 1 to n
    set thisValue to cell "Total" of record i of current collection of window 1
    set theTotal to theTotal + thisValue
  end repeat
end tell
```

Another approach is to request all of the data at once, and then loop through the results.

```
tell application "Informed Filler™"
  set theValues to cell "Total" of every record of current collection of window 1
  set theTotal to 0
  repeat with i in theValues
    set theTotal to theTotal + i
  end repeat
end tell
```

There is a certain amount of overhead involved in communicating with any application. Since the second approach only communicates with Informed Filler once, it will generally be the faster method.

## Making a New Record

The “make” command instructs the target application to make a new object of a given class. You can create a new “record” object in an existing window with the following script.

```
tell application "Informed Filler™"
    -- make a new record in the front window:
    make new record in window 1
end tell
result: record id 4836 of window id 24 of application "Informed Filler™"
```

The result of the “make” command is a reference to the newly created object. This reference can be used to access the new object. The following script makes a new record, then sets several cells in the new record.

```
tell application "Informed Filler™"
    set theRecord to (make new record in window 1)
    set cell "Name" of theRecord to "Fred"
    set cell "Era" of theRecord to "Jurassic"
end tell
```

## Deleting Records

The “delete” command is used to delete objects in the target application. Any record or set of records may be deleted.

```
tell application "Informed Filler™"
    -- delete the current record
    delete current record of window 1

    -- delete the first three records:
    delete records 1 thru 3 of current collection of window 1

    -- delete every record
    delete every record of window 1
end tell
```

The following script appears to loop through each record, deleting those that match a test.

```
tell application "Informed Filler™"
    set theCount to number of records in window 1
    repeat with i from 1 to theCount
        if cell "Supervisor" of record i of window 1 = "Fred" then
            delete record i of window 1
        end if
    end repeat
end tell
```

Unfortunately, there is a critical bug in this script. As each matching record is deleted, the index of all subsequent records is reduced by 1. Thus, as the script completes, the loop variable “i” becomes more and more inaccurate. The following script shows a better way to perform this operation.

```
tell application "Informed Filler™"
  set theIDs to ID of every record of window 1
  repeat with i in theIDs
    if cell "Supervisor" of record id i of window 1 = "Fred"
      delete record id i of window 1
    end if
  end repeat
end tell
```

Finally, this particular script can actually be simplified to the following single statement.

```
tell application "Informed Filler™"
  delete (every record whose cell "Supervisor" = "Fred") of window 1
end tell
```

## Duplicating Records

To duplicate a record, you can use the “duplicate” command.

```
tell application "Informed Filler™"
  duplicate current record of window 1
end tell
```

You can also duplicate a set of records. At the end of such an operation, the current record will be the duplicate of the last record specified.

```
tell application "Informed Filler™"
  duplicate (every record whose cell "Supervisor" = "Fred") of window 1
end tell
```

## Saving the Current Record

The current record is normally saved whenever the user presses the Enter key. If your script sets some cell values, when the script finishes, the last cell that was changed will be selected and will be the current cell. If you wish to save the record, and deactivate the current cell, you can use the “save” command.

### Note

In this context, “save” does not apply to the active window or data document. Instead, it corresponds to accepting the current record.

```
tell application "Informed Filler™"
  save current record of window 1
end tell
```

The “save” command can only be applied to the current record.

**Note**

If your record contains check calculations, these may fail, preventing you from saving the record, or even changing the current cell. If checks fail, Informed Filler will beep and reselect the offending value, just as if the user had pressed “Enter” or “tab”.

**Reverting the Current Record**

The “revert” command will revert a record to its last accepted state. This command can only be applied to the current record of a window.

```
tell application "Informed Filler™"
    revert current record of window 1
end tell
```

**Counting the Collection of Records**

Many of the examples given so far apply some task to every record in the current collection. If you need to know how many records are in the current collection, you can use the following script.

```
tell application "Informed Filler™"
    number of records in current collection of window 1
end tell
result: 6
```

This script counts the records in the current collection only. This number will be the same as the one displayed at the bottom left of the form window. You can also find out how many records are in the document, including the ones that are not displayed:

```
tell application "Informed Filler™"
    number of records in window 1
end tell
result: 14
```

The second script includes the 6 records in the current collection, plus 8 more that are not in the collection. If you then did a “Find All”, the number of records in the collection would be 14, the same as the number in the document:

```
tell application "Informed Filler™"
    set current collection of window 1 to every record
    number of record in current collection of window 1
end tell
result: 14
```

## Testing for a Record's Existence

In “Checking the Existence of an Object,” testing for existence of a window was described. You can also test for the existence of a given record, perhaps a record that matches some specific test.

```
tell application "Informed Filler™"
    -- Find Fred and set his salary to 35000
    set empName to "Fred"
    set newSalary to 35000

    -- Check to see if Fred already has a record
    if exists (first record whose cell "Employee" = empName) of window 1 then
        set current record of window 1 to (first record whose cell "Employee" = -
            empName) of window 1
    else
        -- Fred doesn't have a record yet, so make him one
        make new record in window 1
        set cell "Employee" of current record of window 1 to empName
    end if

    -- Set the salary
    set cell "Salary" of current record of window 1 to newSalary
end tell
```

## Cells and Elements

This section discusses cells, elements of cells, and cell values.

### Cell Values

Each element of a cell has a value. Single value cells, drawn with Informed Designer’s Field tool, have a single element, and thus a single value. Multiple value cells, drawn with Informed Designer’s Table tool, have one element for each row in the table, and thus one value for each row in the table.

References to a cell will, by default, refer to the value of the cell. This makes it easier to get and set the value of a cell. Each cell has a default data type, which is one of: text, number, boolean, date, picture, and signature.

Picture values represent a PICT image, and can only be copied to and from a picture cell.

The value of a signature cell is an AppleScript record containing all of the pieces of information in the digital signature. Here is an example of a digital signature from PowerTalk™, as returned by Informed Filler:

```
tell application "Informed Filler™"
    value of cell "Sig Cell"
end tell
result: {class:signature, valid:true, common name:"Wayne Peroit, organization:"",
serial number:"8673820673", certificate valid dates:{date "Friday, April 30, 1996
11:00:00 PM", date "Thursday, April 30, 1999 10:59:59 PM"}, signing time:date "Mon-
day, March 6, 1996 10:27:11 AM", distinguished name:{{attribute label:"country
```

```
code", attribute ID:0, writing code:0, attribute value:"US"}, {attribute
label:"postal code", attribute ID:8, writing code:0, attribute value:"00000"},
{attribute label:"common name", attribute ID:5, writing code:0, attribute
value:"Wayne Peroit}}, issuer:{{attribute label:"country code", attribute ID:0,
writing code:0, attribute value:"US"}, {attribute label:"organization", attribute
ID:1, writing code:0, attribute value:"RSA Data Security, Inc."}, {attribute
label:"organization unit", attribute ID:7, writing code:0, attribute value:"Unaf-
filiated User Certification Authority"}}
```

The information available when you verify a digital signature varies depending on which signing service you're using. The following table lists the parts of a PowerTalk digital signature:

#### Digital Signature Parts

Label	Description
class	The data type (always signature).
valid	Tells whether the signature is valid, that is, whether or not the signature was successfully verified.
common name	The common name, or everyday name of the user who owns the signature.
organization	The name of the organization that the user works for. Unaffiliated digital signatures have a blank organization.
serial number	The unique serial number for the digital signature. This serial number is associated with the certificate for that signature, not with the particular signing instance.
certificate valid dates	A list containing the start date and end date for the period during which this signature is valid.
signing time	The data and time that this digital signature instance was created.
distinguished name	A list of attributes that, together, uniquely distinguish the owner of this signature from all other individuals. Each attribute is a record with four parts: attribute label, attribute ID, writing code, and attribute value.
issuer	A list of attributes that, together, uniquely distinguish the issuing authority of this signature.
attribute label	A string label for one attribute of the distinguished name or issuer.
attribute ID	A numeric label for one attribute of the distinguished name or issuer.
writing code	A code representing the language system for one attribute of the distinguished name or issuer.
attribute value	The value of one attribute of a distinguished name or issuer.

The following table lists the possible attributes of the distinguished name and the issuer.

Distinguished Name/Issuer Attributes

Attribute Label	ID
"country code"	0
"organization"	1
"street address"	2
"state"	3
"Locality"	4
"common name"	5
"title"	6
"organization unit"	7
"postal code"	8

## Pictures and Signatures

Picture and signature values are binary values. That is, unlike text values, they are not represented as ASCII characters. Informed Filler allows you to read the value of a picture or signature cell in both binary and ASCII forms. The ASCII forms, however, are Informed-specific and are, therefore, unreadable by other applications. To obtain the ASCII form of a picture or signature cell, append the words “as text” to the cell identifier.

## Lists and Column Cells

When you access a column cell, you are by default accessing every element of that cell. If you request the value of a column cell, you will get back a list of values, with one item in the list for each row in the column cell.

You can also set the value of a column cell to a list of values. This will place one item from the list in each row of the column, starting at the top. If you set a column cell to a single value, that value will be placed into each row of the column.

## Working with Many Cells at the Same Time

Generally, to access several cell values, you must refer to each of the cells individually. The following script copies the value of four cells from one form to another.

```

tell application "Informed Filler™"
    set theName to cell "Name" of window "Employees"
    set theAddress to cell "Address" of window "Employees"
    set theCity to cell "City" of window "Employees"
    set theZip to cell "Zip" of window "Employees"

    set cell "Name" of window "Holidays" to theName
    set cell "Address" of window "Holidays" to theAddress
    set cell "City" of window "Holidays" to theCity
    set cell "Zip" of window "Holidays" to theZip
end tell

```

This script will take some time to execute, because it sends eight separate commands to Informed Filler.

Informed Filler supports a special method of referring to more than one cell at a time. Using this method, you give a list of cell names where you would normally put one cell name. The data that is returned from Informed Filler will be a list with the same number of items as names you provided, and each item will match the corresponding name.

The following script is a two line version of the above script, using lists of names.

```

tell application "Informed Filler™"
    set theData to cell {"Name", "Address", "City", "Zip"} of window "Employees"
    set cell {"Name", "Address", "City", "Zip"} of window "Holidays" to theData
end tell

```

The first line requests four cell values, and puts the resulting list into the variable “theData.” The second line in the script instructs Informed Filler to set the values of each of four cells to a list. Since the number of items in the list matches exactly the number of cells specified, Informed Filler will take the list apart and match each item in the list with the corresponding cell. This method is much faster, as it only sends two commands to Informed Filler.

The following script is even faster, since it combines both of those commands into a single command.

```

tell application "Informed Filler™"
    set cell {"Name", "Address", "City", "Zip"} of window "Holidays" to cell -
        {"Name", "Address", "City", "Zip"} of window "Employees"
end tell

```

## Menus

Menus and menu items are provided as scriptable objects in Informed Filler for cases where the other scriptable objects will not suffice. For example, if you created a script that read some data from another database and displayed that in the choice list for a cell, you might want to make sure the choice list was displayed at the end. Since the choices palette is not a scriptable object, the only way to display it is to test the menu item title and, if it is Show Choices, execute it to show the choices palette.



Here is a complete script that does a simple lookup using FileMaker Pro as a central database, sets the extra choices for the next cell, and displays the choices palette.

```
-- First, get the parts category value:
tell application "Informed Filler™"
    set partCat to cell "PartCategory"
end tell

-- Look up the list of part numbers from FileMaker Pro
tell application "FileMaker Pro"
    try
        Show (every record whose Cell "Category" = partCat) of Window "Parts DB"
        -- got at least one record...
        set partsList to Cell "PartNumber" of every record of Window "Parts DB"
    on error
        -- got an error, so no matches on the category
        set partsList to {}
    end try
end tell

-- Put the matching part numbers back into Informed:
tell application "Informed Filler™"
    set extra choices of cell "PartNum" to partsList

-- Make sure the choices list is displayed:
    if name of menu item 1 of menu "View" = "Show Choices" then execute menu item 1 of menu "View"
end if
end tell
```

You can also use the “enabled” status of menus to get some information about the status of Informed Filler. For example, if the “Sign” menu item of the “Signatures” menu is enabled, you know that the record is active and the current cell is a signature cell. You could use this information to write a general purpose script that gets the serial number of the current cell’s signature, and looks up some information from a central user database, such as the user’s signing authority. By testing the “Sign” menu item’s “enabled” property, you don’t have to build information about cell names into the script.

The following script checks if the current cell is a signature cell and, if so, looks up the serial number of the signature in a FileMaker Pro database. The signing authority of that user is then displayed in a dialog.

```

tell application "Informed Filler™"
    set serialNum to ""
    if menu item "Sign" of menu "Signatures" is enabled then
        set theSig to current cell of window 1
        if valid of theSig then
            set serialNum to serial number of theSig
        end if
    end if
end tell

tell application "FileMaker Pro"
    if serialNum <> ""
        try
            Show (every record whose Cell "SerNum" = serialNum ) of Window ↵
                "Sig DB"
            -- got at least one record...
            set auth to Cell "Authority" of first Record of Window "Sig DB"
            display dialog "The user's signing authority is " & auth & "."
        on error
            -- got an error, so no matches on the category
            display dialog "Couldn't find that serial number."
        end try
    end if
end tell

```

## Printing

Printing of records and record lists can be scripted with the “Print” command. This command allows for several options, which are listed in the following table.

### Optional Parameters for Print

Parameter	Description
from page <integer>	First page to print. The default is to print all pages.
to page <integer>	Last page to print. The default is to print all pages.
from part <integer>	First part to print. The default is to print all parts.
to part <integer>	Last part to print. The default is to print all parts.
as record list	Print the data as a record list. The default is to print the data as individual records.
print template <yes/no>	Control whether the template should be printed. The default is to print the template.
print data <yes/no>	Control whether the data should be printed. The default is to print the data.

The other option you have in printing is what selection of data to print. The options are listed in the following table.

#### Optional Parameters for Printing Data

Print What?	What gets Printed...
window	All records in the window.
collection	All records in the collection.
record	All of the records described by the record specifier.

Here are a few examples:

```
tell application "Informed Filler™"
  -- print everything
  Print window 1

  -- print just the current record
  Print current record of window 1

  -- print the current collection as a list
  Print current collection of window 1 as record list

  -- print some specific records
  Print (every record whose cell "Overdue Amt" > 0) of window 1

  -- print the current collection onto pre-printed forms
  -- (that is, print just the data)
  Print current collection of window 1 print template no
end tell
```

## Importing and Exporting

Data can be imported and exported between Informed Filler documents and various standard data formats. These capabilities are also scriptable using the “import” and “export” commands.

To export data from an Informed document, you use the “export” command in AppleScript. This command allows you to specify a set of records to export, the file to export to, and the data format. The specified data is exported to a new file with the name you have chosen.

The following table lists the data formats you can export to.

#### Export Formats

Format	Description
Informed Interchange	The standard Informed interchange format, which includes non-text data such as pictures and digital signatures, and style information for text data.
tab delimited text	The standard text only data format, with field values delimited by the tab character.
comma delimited text	The standard text only data format, with field values delimited by the comma character.

You can also choose what data to export. The export command always exports all cells in the form, and it replaces the target file, if such a file already exists. The following table lists the options you have, and the resulting selections of data.

#### Export Selections

Export What?	
window	All records in the window.
collection	All records in the collection.
record	All of the records described by the record specifier.

Here are a few example AppleScript fragments that export data from an Informed document to one of the standard data formats.

```
tell application "Informed Filler™"
  -- export everything to tab delimited text
  export window 1 to file "HD:Data File" as tab delimited text

  -- export the collection as Informed interchange
  export current collection of window 1 to file "HD:Data File" as Informed ~
    interchange

  -- export the current record
  export current record of window 1 to file "HD:Data File" as tab delimited text

  -- export some specific records
  export (every record whose cell "Overdue Amt" > 0) of window 1 to file ~
    "HD:Deadbeats" as Informed interchange
end tell
```

You can also import data from a file to an Informed document. The file must be in a format that is understood by Informed Filler.

To import a file, you simply specify the file to import and which window the data should be imported into.

```

tell application "Informed Filler™"
  -- Simple import of existing data file
  import file "HD:Data File" into window 1

  -- Move data to another window using import and export
  set current collection of window 1 to every record of window 1
  set current collection of window 1 to (every record whose cell "Overdue -
    Amt" >0) of window 1
  export current collection of window 1 to file "HD:Data File:Deadbeats" -
    as Informed Interchange
  import file "HD:Data File:Deadbeats" into window "Deadbeats File"
end tell

```

## Mail

Mail is an important aspect of electronic forms. Informed Filler supports many electronic mail systems. Using your mail system, you have the capability to send forms to other users in a variety of formats.

Sending a form to another user is scriptable in Informed Filler by using the “send” command. You can send an entire document, the current collection of records, the current record, or a selection of records. The following table lists the objects you can send, and what data will be included.

### Send Selections

Send What?	What gets Sent...
window	All records in the window.
collection	All records in the collection.
record	All of the records described by the record specifier.

The “send” command also has many optional parameters, which are listed below.

### Send Command Parameters

Parameter	Description
using	Which mail system to use. The options are listed in a subsequent table.
recipients	One or more principal recipients.
cc recipients	One or more recipients to receive a copy.
bcc recipients	One or more recipients to receive a blind copy.
reply recipients	One or more recipients to be replied to.
sender	An alternative identity for the sender (may require authentication of your identity).
confirm	yes - confirm the setting with the user before sending. no - send immediately without a confirmation dialog.
format	The format to send the data in. The options are listed in a subsequent table.
subject	The subject for the mail message.
priority	The priority to send the message at.

The following table lists the mail system options available for the “using” parameter.

#### Mail System Selectors

Mail System	Description
PowerTalk	PowerTalk
QuickMail	QuickMail
Microsoft Mail	Microsoft Mail
cc:Mail	cc:Mail
mail system name	Search for the mail plug-in with the given name, and use that mail system.

The following table lists the data formats available for use with the “format” parameter.

#### Send Format Selectors

Format	Description
Informed data	Send the data alone in Informed data format
Informed package	Send the data with the template in a single integrated package.
Informed Interchange letter	Send the data alone in Informed Interchange format. Send the document in AppleLetter format (PowerTalk only).
Tab delimited text	Send the data alone in tab delimited text format.
Comma delimited text	Send the data alone in comma delimited text format.
format name	Search for a plug-in with the given name, and send the document in that format.

Each of the parameters described above is optional. In general, if some information is missing, the user will be prompted to complete the mail specification. If you specify “confirm no”, and some critical information is missing (such as who the recipients are), the command may result in an error being reported to AppleScript.

When the command completes, the result will be an AppleScript record that contains values for all of the known information. This may include the list of recipients, the data format, priority, subject, and so on. What information is returned, and its exact format, is dependent on the mail system chosen, and on the version of the mail plug-in in use.

# 13

## Informed 4D External

In this chapter:

- Overview 13-2
- Using the INF\_FILL External 13-7
- Using the INF\_NS\_Client External 13-27

# 13

## Informed 4D External

Informed 4D External for 4th DIMENSION is a highly technical product designed for use by trained 4th DIMENSION programmers. This chapter assumes that you're an experienced 4th DIMENSION programmer and that you're familiar with Informed Designer and Informed Filler. Experience with Informed Number Server (now included with Informed Designer) is also helpful if you intend to use this product with your 4th DIMENSION application.

### Overview

Informed 4D External contains a set of powerful 4th DIMENSION external that provide seamless integration of Informed's forms processing capabilities with any 4th DIMENSION database. Based on the IAC (inter-applications communications) of the Mac OS, one of the included external makes it easy to look up information from a 4th DIMENSION database while filling out forms with Informed Filler. Once a form has been completed, the information can be easily inserted—or submitted—directly into the 4th DIMENSION database, therefore eliminating the need to manually export and import data.

Also included is support for Informed Number Server. By making simple modifications to your 4th DIMENSION application, it too can obtain new form numbers from Informed Number Server. That way, users can fill out forms using either your 4th DIMENSION application or Informed Filler on the Mac OS and still maintain unique form numbering among all users.

With Informed Filler acting as the tool for filling out forms, users can avoid running 4th DIMENSION to do their work. For users that currently use your 4th DIMENSION application only for filling out forms, Informed Filler can be used in its place. Users benefit from the higher performance and reduced memory requirements of Informed Filler when compared to a 4th DIMENSION runtime.

Since Informed forms remain separate documents, changes to a form's design can be made easily without affecting your 4th DIMENSION application. Even major form revisions can be made without the need to recompile your application.

If your 4th DIMENSION application is currently used in a single-user environment, Informed can offer a practical way to expand your system to support multi-user data entry without the need to install additional 4th DIMENSION runtime applications. With one single-user 4th DIMENSION runtime available on a networked Mac OS compatible computer, multiple Informed Filler users can fill out and submit forms.



## How it Works

Chapter 1, “Adding Intelligence to Your Forms”, explains how you can configure lookups, auto-incrementing cells, and form submission. Each of these features allows you to select different methods of accessing data sources. One such method is Apple events, an IAC (inter-application communications) capability available on any computer running version 7 or later of the Mac OS. The Informed 4D External<sup>s</sup> rely on this method of communication.

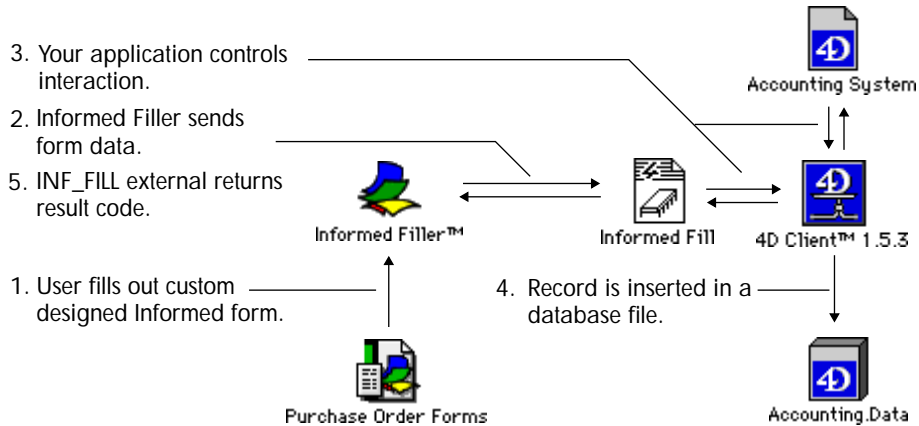
An Apple event is a message that one application sends to another. The message can contain information, or it can request that a certain command be performed. For example, when Informed Filler sends a completed form to a different application, it can send an Apple event containing the information on the form. To perform a lookup, an Apple event that requests a particular value can be sent instead. The other application performs a search and sends back a reply containing the information found.

Informed Designer, Informed Filler, and Informed Number Server use a particular set of Apple events to interact with other applications. Without the Informed 4D External<sup>s</sup>, your 4th DIMENSION application cannot understand these Apple events. This product includes two 4th DIMENSION external<sup>s</sup>. The external named “INF\_FILL” acts as an Apple event handler that can interpret and process any Apple events that are received from Informed Designer or Informed Filler. The external named “INF\_NS\_CLIENT” allows your 4th DIMENSION application to send Apple events to Informed Number Server.

### Form Submission

Form submission normally marks the end of the forms process. Once a form is complete, its contents are accepted by an information system for further processing. The submission of a form often triggers other business procedures that process the information. For example, submission of an approved expense form may trigger the accounting procedure that issues a payment.

The INF\_FILL external enables form submission into any 4th DIMENSION database. Custom form templates are designed with Informed Designer and linked to a file in the 4th DIMENSION database using Informed Designer’s Configure Submit command. Forms filled out with Informed Filler can then be submitted directly into the 4th DIMENSION database by choosing a single menu command. The following figure illustrates this process.



Linking a form template involves mapping each cell on the template to a corresponding field in a table of the 4th DIMENSION database. When a completed form is submitted using Informed Filler, the linking information ensures that each of the cell values on the template are entered in the correct fields in the 4th DIMENSION database file. Templates can be linked to both flat and relational files. For relational files, the external will automatically create the appropriate number of records in each of the related files.

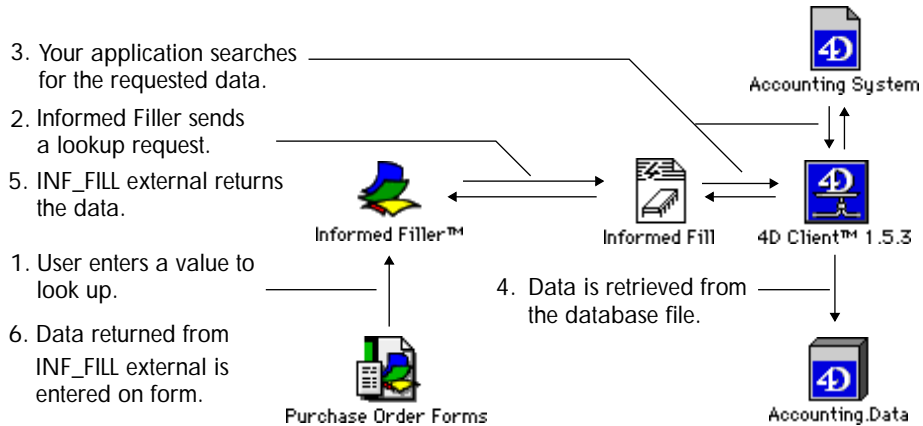
Since the link between Informed Filler and the INF\_FILL external is real time, forms submission occurs interactively. When a form is submitted, your 4th DIMENSION application can examine and validate the data before it's accepted, and messages can be sent back to Informed Filler if errors are detected. That way, users are notified of mistakes immediately rather than later when it is often inconvenient and more costly to correct them.

For detailed information on configuring form submission, see "Form Submission" in Chapter 1, "Adding Intelligence to Your Forms."

## Lookups

Forms often contain information that already exists electronically. Rather than retyping this information to fill out a form, Informed allows you to configure cells to lookup information in other forms or databases.

With the INF\_FILL external, lookups can extract information from a 4th DIMENSION database. Using Informed Designer, you link the cells on a form that are to be looked up with fields in a file of your 4th DIMENSION database. When the user filling out a form enters a value in a particular cell, a lookup request is sent to the 4th DIMENSION application. Your application can search for and return the requested information to Informed Filler where it's automatically entered on the form.

**Note**

The INF\_FILL external is capable of returning multiple records to Informed Filler. If a lookup is performed and multiple matching records are found, Informed Filler will prompt the user with a scrolling list displaying all matches. A single record can then be selected by the user and entered on the form. This feature is helpful in situations where the user doesn't know in advance exactly which information is being looked up. The user may, for example, enter only the first few letters of a name to lookup a customer record.

For detailed information on configuring lookups, please see “Using Lookups” in Chapter 1, “Adding Intelligence to Your Forms.”

## Supporting Informed Number Server

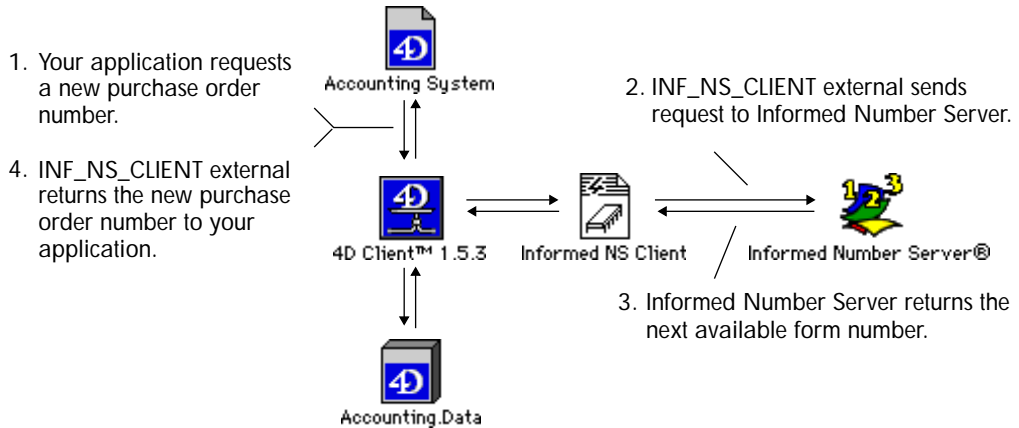
Unique form numbering is important for many types of forms. Form numbers provide a way to uniquely identify any form. Invoice numbers and purchase order numbers are examples of form numbers.

“Auto-incrementing Cells” in Chapter 1, explains how you can link an auto-incrementing cell to a variety of data sources. One such method of linking uses Apple events as a means of communication. This is the method used to link with Informed Number.

Informed Number Server was designed to automate the assignment of form numbers as different people fill out different forms. Each time a new form is filled out, Informed Filler sends an Apple event to Informed Number Server requesting a form number for the type of form being filled out. Informed Number Server replies with the next available form number.

In order to maintain unique consecutive numbering, form numbers must be obtained from a single source. Support for Informed Number Server in your 4th DIMENSION application is important if users are to fill out forms using both 4th DIMENSION and Informed Filler. Although Informed Filler may be your users' primary tool for filling out forms, you may have duplicated certain form designs in your 4th DIMENSION application as a convenience for those who use 4th DIMENSION

more often. The following figure illustrates how your 4th DIMENSION application interacts with Informed Number Server.



If your 4th DIMENSION application is already designed to assign unique form numbers to new forms, making the necessary modifications to take advantage of Informed Number Server is simple and straightforward. Rather than obtaining new form numbers the way you normally do, you'll change your application to request new numbers from Informed Number Server instead. This is done by making calls to the Informed NS Client external to specify where the Informed Number Server is running and the type of form for which numbers are requested.

## System Requirements

The Informed 4D External for 4th DIMENSION depend on specific versions of system software and applications. They are:

- System software version 7.0 or greater
- Informed Designer version 1.3 or greater
- Informed Filler version 2.0 or greater (or Informed Manager version 1.3 or greater)
- 4th DIMENSION version 2.2.1 or greater
- CallProcs.Ext external (required for use with 4th DIMENSION 2.2)

The externals work with both versions 2.2 and 3.x of 4th DIMENSION. They also work with the 4D SERVER. Due to certain changes and enhancements in version 3.x and the 4D SERVER, simple changes must be made to your application and how it interacts with the INF\_FILL external. These changes are explained in the following section, "Using the INF\_FILL external."

## Using the INF\_FILL External

In order to protect data integrity, the INF\_FILL external is built on an open architecture that gives you, the 4th DIMENSION developer, full control over how Informed interacts with your application. Once an Apple event has been received, you make calls to the external to process the event in a controlled manner. That way, you can reject invalid data, deny access to certain files, and ensure that the integrity of your 4th DIMENSION application and database is protected.

Certain minimal changes must be made to your 4th DIMENSION application in order to use it with Informed. This requirement is imposed intentionally to prevent unauthorized access to a 4th DIMENSION application and database. It prevents Informed users from obtaining access to a 4th DIMENSION database without the application developer enabling this capability. This chapter describes how you can modify your application to take advantage of form submission and lookups using Informed. An explanation of the INF\_NS\_CLIENT external can be found later in this chapter.

## Using 4th DIMENSION Version 3.x and the 4D SERVER

The INF\_FILL external works with both versions 2.2 and 3.x of 4th DIMENSION, as well as the 4D SERVER. With the exception of the mechanism used to poll the INF\_FILL external's event queue, a 4th DIMENSION application interacts with the external in much the same way, regardless of which version of 4th DIMENSION you're using.

As explained in "Accepting Apple Events" later in this chapter, Informed Filler sends Apple events to your application whenever the user submits a form or triggers a lookup. Your application is expected to poll a queue and process any Apple events that are received. For 4th DIMENSION version 2.2, ACIUS's CallProcs.Ext external is required. This external allows you to configure 4th DIMENSION to repeatedly call a specified polling procedure during idle time.

With 4th DIMENSION version 3.x and the 4D SERVER, the new *processes* capability offers a more appropriate polling mechanism. A process is like a procedure that executes concurrently in its own 4th DIMENSION environment independent of other processes. Since each process has its own current selection and current record information, its operation doesn't interfere with the 4th DIMENSION application or any other processes that might be running.

For more information about processing Apple events, please see "Accepting Apple Events" and "Processing Apple Events" later in this chapter.

## Installing the External

You install 4th DIMENSION externals using the 4D External Mover application provided with 4th DIMENSION. The INF\_FILL external can be found in the file ‘Informed Fill.’

The INF\_FILL external can be installed in any copy of your interpreted or compiled database application, or in your CallProc.Ext file.

Since the interaction between Informed Filler and your 4th DIMENSION database occurs through a preselected 4th DIMENSION development or runtime application, the INF\_FILL external must be available to that application. When you select which 4th DIMENSION development or runtime application to link forms to, you should choose the one that can be running whenever Informed Filler users may be filling out forms.

Informed 4D Externals comes with two versions of a sample 4th DIMENSION database application (one for 4th DIMENSION version 2.2 and the other for version 3.x and the 4D SERVER), and an Informed template that’s already configured for lookups and form submission. You can use these items to experiment with the capabilities of the INF\_FILL external.

## Accepting Apple Events

When an Informed Filler user triggers a lookup or sends a completed form, an Apple event is sent to your 4th DIMENSION application. The external accepts the event and places it in a queue where it waits to be processed. Your 4th DIMENSION application is expected to periodically check for events in the queue by calling the procedure `InfAECCount`. This procedure returns a single integer parameter indicating the number of events in the queue. Its declaration is shown below.

```
InfAECCount ($theCount)
```

If an event is found in the queue, you then call the external procedure `InfAERead` to read the event. This procedure returns a parameter indicating the type of the event.

```
InfAERead ($theType)
```

For insert or submit events, the value of `$theType` will be 1 whereas lookup events are identified by the value 2. Other event types are processed internally and should be ignored. These are events that Informed Designer generates while you link a form or configure a lookup.

Once you’ve read an event and you know its type, you should then process the event in an appropriate manner. Event processing is explained in the section “Processing Apple Events” later in this chapter.

Apple events can be sent to your 4th DIMENSION application from multiple users connected to the same network. Each Apple event, however, is read and queued sequentially. As a result, your 4th DIMENSION application processes incoming Apple events one at a time, thereby avoiding potential problems caused by concurrent access by multiple users.

## Polling the Apple Event Queue

In order to process Apple events, your application must continually poll the Apple event queue. If you're using 4th DIMENSION version 2.2, we recommend that you use ACIUS's CallProcs.Ext external. With the CallProcs.Ext external installed, 4th DIMENSION can be configured to call one or more global procedures during idle time. The call shown below configures 4th DIMENSION to automatically call the global procedure ProcessAllAEs once every 60 ticks (60 ticks = 1 second).

```
AddProc ("ProcessAllAEs"; 60)
```

If you're using version 3.x of 4th DIMENSION or the 4D SERVER, polling is accomplished by invoking a new process rather than using the CallProcs.Ext external. Since a process executes independently, processing Apple events using this mechanism ensures that your 4th DIMENSION application and other processes are not affected. The call shown below creates a new process named AEPollProcess. The value of 96000 should be adequate for most applications.

```
pid := New Process ("AEPollProcess";96000;"CallProAllAEs")
```

The process' procedure should be written to repeatedly check for and process any Apple events in the event queue, or call another procedure that does so. This is essentially what the CallProcs external accomplishes for applications based on version 2.2 of 4th DIMENSION. The process procedure shown below uses an endless repeat loop to call the ProcessAllAEs procedure. The 'Delay Process' call is necessary so that 4th DIMENSION allows other processes time to execute. 60 Ticks is adequate for most applications.

```
PROCEDURE AEPollProcess
While (True)
  ProcessAllAEs
  Delay Process (current process;60)
End while
```

The ProcessAllAEs procedure should be written to check for Apple events waiting in the event queue and process those that are found. Below is an example procedure that's provided with the Informed 4D External. This procedure can be used unmodified in your 4th DIMENSION application.

```
PROCEDURE ProcessAllAEs

C_INTEGER ($theCount)
C_INTEGER ($theType)

InFAECount ($theCount)
While (Not ($theCount = 0))
  InFAERead ($theType)
  Case of
    :($theType = 1)
      ProcessInsert
    :($theType = 2)
      ProcessLookup
  End case
  InFAECount ($theCount)
End while
```

In addition to the above procedure, you must also write the procedures ProcessInsert and ProcessLookup. These procedures should be written to process incoming Apple events in a manner that's appropriate for your application. Information about processing Apple events can be found later in this chapter.

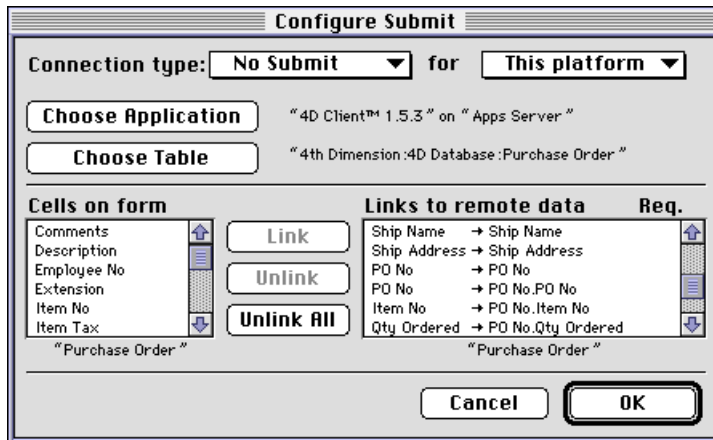
## Linking Forms

Forms are linked to your 4th DIMENSION application using Informed Designer. Linking is required both to configure lookups and to link forms for form submission.

The linking process involves mapping cells on the Informed form to fields in a file of a 4th DIMENSION database. For detailed information on Informed Designer's Lookup and Configure Submit commands, see Chapter 1, "Adding Intelligence to Your Forms." The issues related specifically to Informed and 4th DIMENSION are discussed here.

## Required and Unique Fields

When you link a form to a file in your 4th DIMENSION database, the Configure Submit dialog box displays a list of cells on the form and a list of the fields in the database file.



The 'Req.' column next to the list of database fields is intended to indicate which fields require values when a completed form is submitted. This column does not reflect 4th DIMENSION's required field attribute of each field.

You should not rely on 4th DIMENSION's required and unique field attributes to prevent the entry of blank or duplicate values. Instead, you should check for blank or duplicate values when you examine the contents of a form for errors. See "Validating Data" later in this chapter for more information.



## Relational Files

Lookups can retrieve information from a single file in your 4th DIMENSION database. A form, however, can be configured for submission to multiple related files. The submission of a single form can result in the creation of records in each of the files.

Related files are often used to reduce the amount of redundant data stored in a database. For example, a purchase order may be stored in two files, one containing header information and the other containing line item information. Each purchase order would consist of one header record and multiple line item records, one for each item ordered. By dividing the purchase order into two files, the header information (that is, the purchase order number, date, terms, and so on) is stored only once rather than redundantly with each line item record. Using 4th DIMENSION terminology, the header file is called the 'one file.' The line item file is called the 'many file.'

Purchase Order Header File			
PO Number	Vendor Number	Date	Terms
1009	V9061	Jan 17/92	Net 30

Line Items File				
PO Number	Item Number	Quantity	Price	
1009	15112	25	150.00	
1009	13207	150	35.50	
1009	13382	10	49.00	
1009	18711	34	275.00	

The 4th DIMENSION programming interface does not allow an external to determine the relations that exist in the structure of your application. In order to make this information available to the INF\_FILL external, your application should call the external procedure InfAddRelation once for each pair of related files on startup.

```
InfAddRelation ($oneFile; $relatedField; $manyFile)
```

\$oneFile is the file number that identifies the one file (the header file in the purchase order example). \$manyFile is the file number of the related file. \$relatedField is the field number of the field in the one file that binds the related records in the two related files. In the purchase order example, the related field would be the purchase order number field. This field is stored in both the header file and with each line item record. The call shown below would correctly describe the related files.

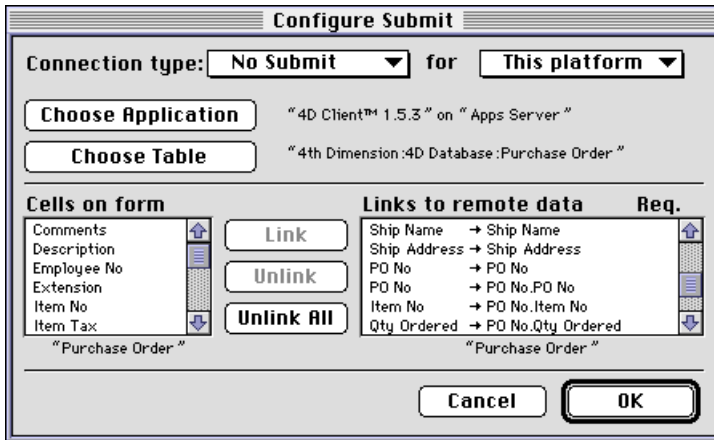
```
InfAddRelation (File (>[PO Header]); Field (>[PO Header]PO Number); File (>[Line Items]))
```

By calling InfAddRelation to describe related files, Informed Designer will allow you to link a single form to multiple related files.

**Note**

Your application must call `InfAddRelation` before any Apple events are accepted. This initialization should occur in your application's `STARTUP` procedure. After your application processes the first Apple event, calls to `InfAddRelation` will be ignored.

When you select a file in your 4th DIMENSION database to submit a form to, Informed Designer will list all fields of both the selected file as well as any related files on the Configure Submit dialog box. The field names of the fields contained in a related many file will be prefixed by the name of the related field.



Fields in a many file should be linked to multi-value column cells on your form. For example, fields such as the item number, quantity, and price in the line items file of a purchase order should be linked to the corresponding column cells on your purchase order form. When the Informed Filler user submits a completed purchase order form, the `INF_FILL` external will automatically insert one line item record for each value in the linked column cells. If you link a single-value cell to a field in a many file, the cell's value will be stored in each of the multiple records created.

**Subfields**

4th DIMENSION offers an alternate method for storing relational data. In addition to standard data types such as text, integer, and date, a single field can store multiple *subfile* records. Each subfile contains one or more fields called *subfields*. With subfiles, relational data can be stored in a single file. A purchase order, for example, could be stored in a single file containing fields for header information and a subfile for the line items.

**Note**

The `INF_FILL` external allows you to insert records into files containing subfiles. You cannot, however, lookup information that's stored in subfiles.

When you configure submission to a file containing one or more subfiles, each subfield can be linked to an individual cell on your form. On the Configure Submit dialog box, Informed Designer will prefix each subfield with the name of its containing subfile.

**Configure Submit**

Connection type: **No Submit** for **This platform**

**Choose Application** "4D Client™ 1.5.3" on "Apps Server"

**Choose Table** "4th Dimension:4D Database:Purchase Order"

Cells on form	Links to remote data	Req.
Comments	Ship Name → Ship Name	
Description	Ship Address → Ship Address	
Employee No	PO No → PO No	
Extenson	PO No → ItemsSub.Item No	
Item No	Item No → ItemsSub.Qty Ordered	
Item Tax	Qty Ordered → ItemsSub.Price	
"Purchase Order"	"Purchase Order"	

**Cancel** **OK**

Subfields should be linked to multi-value column cells on your form. When Informed Filler sends a completed form to your 4th DIMENSION application, the INF\_FILL external will automatically create a new subfile record for each set—or row—of values in the column cells that are linked to the subfields. If you link a single-value cell to a subfield, the cell's value will be stored in each of the multiple subfile records created.

## Data Types

Informed supports a rich set of data types and formatting options. The data types include text, character, number, name, date, time, boolean, picture, and signature. Although 4th DIMENSION supports most of these data types, its formatting capabilities are not as robust. For example, 4th DIMENSION does not allow you to enter a date such as 'Monday, June 24, 1996,' whereas Informed will accept this format.

Cells that are linked to fields in your 4th DIMENSION database must be formatted according to the constraints of 4th DIMENSION. The following table lists each Informed data type along with the compatible 4th DIMENSION data type and any formatting constraints.

Informed and 4th DIMENSION data types

Informed Data Type	4th DIMENSION Data Type	Required Format
Text	Text	No formatting constraints
Character	Alpha	Format characters must be stripped
Name	Text	No formatting constraints
Number	Number	No formatting constraints
Date	Date	MM/DD/YY
Time	Time	HH:MM:SS
Boolean	Boolean	No formatting constraints
Picture	Picture	No formatting constraints
Signature	Picture	No formatting constraints

If you want to use a cell format that's not supported by 4th DIMENSION, you can work around this limitation by creating an additional cell with an acceptable format. This additional cell would be calculated to be equal to the original cell and linked to the 4th DIMENSION field in its place. You might place this cell on the work page.

Like Informed, 4th DIMENSION supports character formatting for telephone numbers and other fixed format text values. Format characters such as dashes or parentheses are automatically entered for you. With 4th DIMENSION, format characters must be omitted when you enter a value. When Informed Filler sends a value to the INF\_FILL external, format characters are included. Therefore, a telephone number such as '(403) 463-3330' might be interpreted incorrectly by 4th DIMENSION as '((40) 3) -463-' if the value is displayed using a telephone output formatter. As explained above, you can work around this incompatibility by creating an additional cell and calculating its value as the formatted value with the format characters removed.

## Processing Apple Events

Once an Apple event has been accepted, your application should then call a procedure to process the event. Processing an event involves either inserting data in the case of an insert event, or searching for data in the case of a lookup event. Processing is completed by unloading the current records of all affected files and sending a reply back to Informed Filler indicating either that the event was processed successfully, or that an error occurred.

### Processing Insert Events

When the Informed Filler user completes and submits a form, an Apple event containing the form information is sent to your 4th DIMENSION application according to the linking configuration of the form. Your application is expected to read the Apple event, insert or reject the data, then send a reply back to Informed Filler.

Depending on the nature of the form, you may or may not want to validate the data before accepting it. For example, you may want to check sensitive accounting data on financial forms to ensure that accounting errors are not made. You may also want to control more tightly the integrity of your 4th

DIMENSION application by saving and restoring the current selection of any affected files, or by completing a transaction through additional entries in the database. For example, processing an invoice form may involve adjusting the customer's balance and the inventory quantities.

## Determining the Form Type

Once your application has read an insert event by calling `InfAERead` (see “Accepting Apple Events” earlier in this chapter), the data to be inserted is held in a special buffer maintained internally by the external. At this point you can determine which file or files the data is being inserted into by calling the two external procedures shown below.

```
InfNewRecCount ($theCount)
InfNewRecInfo ($theIndex; $theFileNum; $theRecNum)
```

`InfNewRecCount` returns an integer value in `$theCount` indicating the number of new records that will be created. For forms that map to a single file in your 4th DIMENSION database, `$theCount` will return the value 1. For forms that are linked to related files, `$theCount` may return a value greater than 1. For example, suppose that your 4th DIMENSION application stores information for a single invoice in two related files, one containing the invoice header fields (that is, the invoice number, date, terms, and so on), and the other containing line item information, one record for each item. If an invoice with three line items were sent to your application, the value of `$theCount` would be 4, indicating one header record and three line item records. (For more information about linking forms to related files, see “Relational Files” earlier in this chapter.)

To find out the file or files in which new records will be created, call the external procedure `InfNewRecInfo`. The value of `$theIndex` should be between 1 and the number of records to be created. In `$theFileNum`, `InfNewRecInfo` will return the file number of the file in which the record will be inserted. The value of `$theRecNum` will be -1 and should be ignored at this point.

## Controlling Access to Files

By knowing which file a record is being inserted into before the record is inserted, you can programmatically prevent the creation of data in particular files of your database. For example, suppose that of the ten files in your 4th DIMENSION database, you're allowing remote submission into only two files. After determining which file or files records will be inserted into, you can proceed with processing the event only if insertion into that file is allowed.

Below is an example procedure that processes insert Apple events. In this example, remote forms submission is allowed only for the files identified by file numbers 4 and 5.

```
PROCEDURE ProcessInsert

C_BOOLEAN ($allowInsert)
C_INTEGER ($theCount)
C_INTEGER ($theFileNum)
C_LONGINT ($theRecNum)

InfNewRecCount ($theCount)
$allowInsert := True
While ($theCount # 0)
```

```

InfNewRecInfo ($theCount, $theFileNum, $theRecNum)
If (Not (($theFileNum = 4) | ($theFileNum = 5)))
    $allowInsert := False
    InfErrorMsg ("You cannot submit that type of form.")
    $theCount := 1
End if
$theCount := $theCount - 1
End while

If $allowInsert
    InfDoInsert
    UnloadRecords
End if
InfSendReply

```

This procedure calls `InfNewRecInfo` for each record about to be inserted to determine which files will be affected. If the file number is determined to be either 4 or 5 for at least one of the records, the insert event is not processed.

If you're not concerned about preventing access to certain files, much of the code in the previous example is unnecessary. Your `ProcessInsert` procedure could, therefore, be as simple as the following procedure.

```

PROCEDURE ProcessInsert

InfDoInsert
UnloadRecords
InfSendReply

```

The above procedure inserts the data from the sent form and sends a confirmation reply back to Informed Filler. The procedure doesn't check which file data is being inserted into, nor does it check the inserted data for errors.

After calling `InfDoInsert` (see "Inserting the Data"), the procedure `UnloadRecords` is called to unload the current records of any affected files. Unloading records is the responsibility of your application. See "Unloading Current Records" later in this section for more information.

## Inserting the Data

The information from the submitted form remains in a special buffer until your application calls the external procedure `InfDoInsert`. This procedure creates the necessary record or records in the database files and fills their contents with the form information. The following example demonstrates the use of the `InfDoInsert` external procedure.

```

PROCEDURE ProcessInsert

If (FileAllowed)
    InfDoInsert
    ValidateData
    UnloadRecords
Else
    InfErrorMsg ("You cannot submit that type of form.")
End if
InfSendReply

```

The function FileAllowed is assumed to return false if insertion into the requested file is not allowed. After calling InfDoInsert, you can then call InfNewRecInfo to obtain the record numbers of the records inserted. This allows you to examine the contents of the inserted records in order to check for errors (see “Validating Data” for more information).

## Validating Data

Data validation is often critical in preventing errors from being accepted. After calling InfDoInsert, your application can examine the contents of the new record (or records) by first calling the external procedures InfNewRecCount and InfNewRecInfo. InfNewRecCount returns the number of new records created and InfNewRecInfo returns the record numbers of the new records. By calling 4th DIMENSION’s GOTO RECORD procedure, you can examine the data for each new record, then either proceed normally or return an appropriate error message. The following example shows a function that checks new purchase orders for blank or duplicate purchase order numbers.

```

FUNCTION Validate_PO

C_BOOLEAN ($theResult)
C_STRING (20;$thePONum)
C_INTEGER ($theFileNum)
C_LONGINT ($theRecNum)
C_INTEGER ($theCount)

$theResult := True
InfNewRecInfo (1; $theFileNum; $theRecNum)

GOTO RECORD ([PO Header]; $theRecNum)
$thePONum := [PO Header]PO No

` Check for blank purchase order number
If ($thePONum = "")
    $theResult := False
    InfErrorMsg ("You must enter the purchase order number.")
Else
    ` Check for duplicate purchase order number
    ALL RECORDS ([PO Header])
    SEARCH ([PO Header];[PO Header]PO No; =;$thePONum)
    If (Records in selection ([PO Header]) > 1)
        $theResult := False
        InfErrorMsg ("A purchase order with that number already exists")
    End if
End if

```

```

` If an error was detected, delete the inserted records
If (Not ($theResult))
  InfNewRecCount ($theCount)
  While (Not ($theCount = 0))
    InfNewRecInfo ($theCount, $theFileNum, $theRecNum)
    GOTO RECORD (File ($theFileNum)»; $theRecNum)
    DELETE RECORD (File ($theFileNum)»)
    $theCount := $theCount - 1
  End while
End if

$0 := $theResult

```

This example assumes that purchase orders are stored in two files, one containing the header information, and the other containing one record for each item ordered. The name of the purchase order header file is ‘PO Header’ and the field in this file containing the purchase order number is named ‘PO No.’ For information about error reporting, please see “Sending Errors Back to Informed Filler” later in this chapter.

For forms that are mapped to multiple related files, you can assume that calling `InfNewRecInfo` with a value of 1 for `$theIndex` will return the file and record numbers of the single record inserted into the ‘one file.’ In the example shown above, there’s no need to call `InfNewRecCount` before the first call to `InfNewRecInfo` since we know that passing 1 in `$theIndex` will return the record number for the record inserted into the purchase order header file. For more information about how forms are mapped to related files, see “Relational Files” earlier in this chapter.

## Unloading Current Records

After calling `InfDoInsert`, your application is expected to unload the current records for any files that were affected by the insert. Unloading these records ensures that they’re unlocked for use in other transactions. The procedure shown below is written generically and can be used in your application without modification.

```

PROCEDURE UnloadRecords

C_INTEGER ($theCount;$theFileNum;$theRecNum)

InfNewRecCount ($theCount)
While ($theCount # 0)
  InfNewRecInfo ($theCount, $theFileNum, $theRecNum)
  UNLOAD RECORD (File ($theFileNum)»)
  $theCount := $theCount - 1
End while

```

If your application is aware of which files records were inserted into, you can avoid calling `InfNewRecCount` and `InfNewRecInfo` and simply call `UNLOAD RECORD` with the appropriate file number.



## Using Transactions

If processing the insertion of a single form involves making several entries in the database, you may want to start a 4th DIMENSION transaction before calling InfDoInsert. By starting a transaction, you can more easily roll back submission of the form if an error is detected.

Suppose that processing an invoice involves adjusting inventory quantities and customer balances in addition to inserting the invoice header and line item records. After inserting the new records, your application may check for errors such as a duplicate invoice number or insufficient customer credit. If an error is detected, the new records must be deleted and any other changes to the database reverted. If your application starts a transaction before processing the event, rolling back or canceling can be done simply by calling the 4th DIMENSION procedure CANCEL TRANSACTION. Below is an example.

```
PROCEDURE ProcessInsert

C_BOOLEAN ($dataOK)

If (FileAllowed)
  START TRANSACTION (*)
  InfDoInsert
  ValidateData ($dataOK)
  If $dataOK = True
    VALIDATE TRANSACTION
  Else
    CANCEL TRANSACTION
  End if
Else
  InfErrorMsg ("You cannot submit that type of form.")
End if
UnloadRecords
InfSendReply
```

Even for simple forms, the use of transactions is highly recommended if your 4th DIMENSION application is used in a multi-user environment. Using transactions will ensure that the changes made to a database as a result of processing a submitted form will not be available to other users until processing has completed and the transaction has been validated. However, if you're using version 2.2 of 4th DIMENSION, care must be taken to ensure that the processing of Apple events doesn't interfere with other transactions that may be invoked directly by your application. For more information, see "Coordinating Interaction" later in this chapter.

## Processing Lookup Events

A lookup is triggered when the Informed Filler user types a value in a lookup cell. The value typed is looked up either in another form or by another application, and the information retrieved is entered on the form.

If a lookup is linked to your 4th DIMENSION application, Informed Filler will send it an Apple event when the lookup is triggered. After accepting the event (see "Accepting Apple Events" earlier in this chapter), your application is expected to search for the requested record and make it the cur-

rent selection of the appropriate file. Later when you send a reply back to Informed Filler, the external will transfer the contents of the current selection. Informed Filler will then enter the field values from your 4th DIMENSION database in the appropriate cells on the form.

## Performing the Lookup

Once you've read an Apple event (via `InfAERead`) and determined it to be a lookup event, your application should call the external procedure `InfDoLookup` to perform the lookup. This procedure takes a single string parameter.

```
InfDoLookup ($theSearchProcedure)
```

The value of `$theSearchProcedure` should be the name of a global procedure in your application which performs the search and leaves the data found in the current selection of the appropriate file. `InfDoLookup` passes four parameters to the search procedure. They are:

- \$1: the file number of the file for which the lookup has been requested
- \$2: the field number of the field on which the lookup is based
- \$3: the match option of the lookup as configured by the form designer  
1 = do not lookup; 2 = use next value
- \$4: the value to be searched for as entered by the Informed Filler user

Suppose, for example, that an invoice form were configured to look up the part number and price of an item when the Informed Filler user enters the item's description. `InfDoLookup` would call your search procedure passing the file number of the inventory file in \$1, the field number of the description field in \$2, the lookup match option in \$3, and in \$4, the value of the description entered by the Informed Filler user. The following figure shows a simple search procedure called 'MySearch' as well as a procedure which your application might call to process lookup events

```
PROCEDURE MySearch

C_INTEGER ($1;$2;$3)
C_STRING (80;$4)

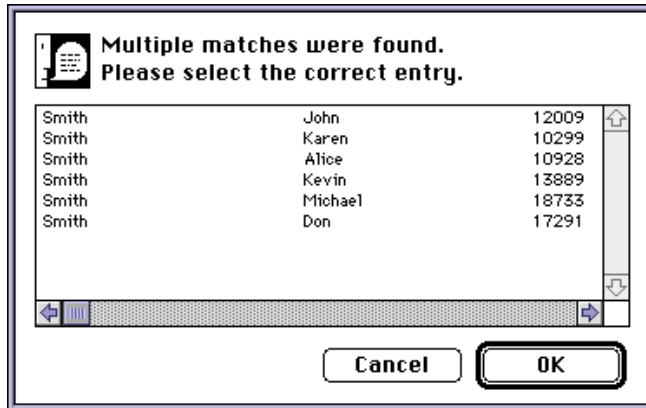
ALL RECORDS (File ($1)»)
SEARCH (File ($1)»; Field ($1; $2)» = $4)

PROCEDURE ProcessLookup

InfDoLookup ("MySearch")
InfSendReply
```

The search procedure searches for an exact match of the search value. If the search fails, the current selection of the file will be left empty. If one or more records are found, they will be available in the file's current selection following the call to `SEARCH`. Their field values will be included with the reply that's sent back to Informed Filler when your application calls `InfSendReply`.

If your search procedure returns a single record selection, Informed Filler will enter the field values of that record in the appropriate cells on the form. If the selection contains more than one record, a dialog box will display allowing the Informed Filler user to browse through the list of records and select the correct one. The browsing dialog box is shown below.



Lookup browsing is useful when there are multiple records containing the requested search value. For example, suppose that an invoice form is configured to lookup customer information when the customer's surname is entered on the form. If there are two or more customers with the same surname, Informed Filler will display those customers in the lookup browsing dialog box. The user can select the correct customer based on related customer information then click 'OK' to enter the information on the form.

The number of records that the external can return to Informed Filler is limited by the maximum size of an Apple event. The size of the combined records cannot exceed 20K. You can further limit the number of records that can be returned to Informed Filler by calling the external procedure `InfMaxRecords`. By default, the limit is set to 50 records.

```
InfMaxRecords ($numRecords)
```

This setting should be based on network performance and cost. The slower the network, the longer it takes to transfer the data from your 4th DIMENSION application to Informed Filler. Users should be encouraged to enter complete values before triggering lookups. That way, the number of matching records is kept to a minimum.

## Match Options and Wildcards

When a lookup is configured, the form designer chooses one of two match options. The match option determines what happens if an exact match is not found when the lookup is performed.

When InfDoLookup calls your 4th DIMENSION search procedure, it passes the lookup's match option as the third parameter according to the values shown below.

1 = Do not lookup  
2 = Use next value

You can control the interpretation of the match option by changing the lookup search procedure, or you can ignore the match option entirely if it's not relevant to your application. The sample search procedure below returns the record containing the next higher search value if an exact match is not found.

```
PROCEDURE MySearch

C_LONGINT ($recCount)
C_INTEGER ($1;$2;$3)
C_STRING (80;$4)

ALL RECORDS (File ($1)»)
SEARCH (File ($1)»; Field ($1; $2)»; "="; $4)
SORT SELECTION (File ($1)»; Field ($1; $2)»; ">")
$recCount := Records in selection (File ($1)»)
If ($recCount = 0) And ($3 = 2))
    ALL RECORDS (File ($1)»)
    SEARCH (File ($1)»; Field ($1; $2)» >= $3)
    $recCount := Records in selection (File ($1)»)
    If ($recCount > 0)
        SORT SELECTION (File ($1)»; Field ($1; $2)»; ">")
        ` Make the first record the current record.
        FIRST RECORD (File ($1)»)
        ` Remove other records from the current selection.
        ONE RECORD SELECT (File ($1))
    End if
End if
```

4th DIMENSION allows you to use wildcard symbols for partial matches. The Informed Filler user could, for example, find all customers whose surname starts with 'Smi' by entering the search value 'Smi@.' This feature is useful if the user doesn't know the exact spelling of the value being looked up.

## Unloading Current Records

If your 4th DIMENSION application is used in a multi-user environment, you should be sure to unload the current records of any files that were affected by a lookup. Records are unloaded by calling the 4th DIMENSION procedure UNLOAD RECORD. If you do not unload the current records after performing a lookup, users of your application may encounter 'File in use' errors.

## Sending Errors Back to Informed Filler

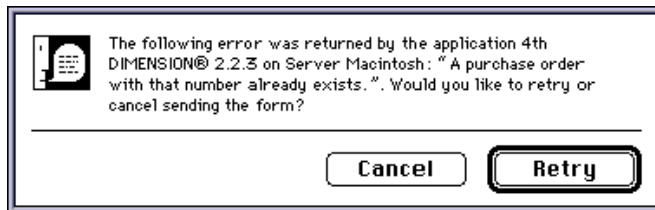
Once an Apple event has been processed, your application is expected to send a reply back to Informed Filler by calling the external procedure `InfSendReply`. Sending a reply confirms that processing has completed.

Your application can call one of two external procedures to report errors back to the Informed Filler user. Before calling `InfSendReply`, simply call either `InfErrorCode` or `InfErrorMsg` to set the error code or message, respectively.

```
InfErrorCode ($theCode)
InfErrorMsg ($theMessage)
```

The value of `$theCode` is an integer and is defined by your application. The value of `$theMessage` is a string that allows for a maximum length of 255 characters.

Each time an Apple event is accepted, the `INF_FILL` external automatically clears the error code and message. If an error is detected while processing an insert event, your application should call either `InfErrorCode` or `InfErrorMsg` before calling `InfSendReply`. The dialog box that Informed Filler displays when your application calls `InfSendReply` will show the error code or message.



In practice, error codes generally do not provide sufficient information for the Informed Filler user to understand a problem and, therefore, are best suited for debugging purposes.

## Coordinating Interaction

With version 2.2 of 4th DIMENSION, the interaction between Informed Filler and the 4th DIMENSION database occurs through your 4th DIMENSION client application. Care must be taken to prevent the processing of Apple events from interfering with the application's normal use. With version 3.x of 4th DIMENSION and the 4D SERVER, the `INF_FILL` external interacts directly with an independent process, therefore eliminating any possible interference with your client application.

If your v2.2 4th DIMENSION application is used in a multi-user environment, you can avoid the concern of coordinating Apple event processing altogether. Instead of relying on a user's copy of 4th DIMENSION for processing insert and lookup events, you could link forms to an additional 4th DIMENSION runtime that's dedicated for this purpose.

The precautions described in the following sections apply only if you're using version 2.2 of 4th DIMENSION and only if forms are linked to a 4th DIMENSION development application or runtime which is also being shared with user.

## Current Selection and Current Record

Since processing a lookup event changes the file's current record and current selection, you may want to save these attributes before searching and restore them afterwards. If your application maintains global variables for attributes such as the number of records in a file, you may have to update these variables as well.

Certain 4th DIMENSION actions such as printing a report or editing a layout rely on a consistent selection. Calling commands that change the current selection during these operations can generate errors. Your application, therefore, must ensure that the processing of Apple events doesn't interfere with commands that may be invoked due to user actions.

## Transactions

If you're using 4th DIMENSION version 2.2.x, and the procedures that you've written to process Apple events use transactions, additional precautions must be taken to avoid potential conflicts. By definition, 4th DIMENSION allows no more than one active transaction per user. Your application, therefore, must delay the processing of an Apple event that modifies the database if a transaction is already active. If processing the Apple event relies on transactions, an error will occur when the transaction is initiated if another transaction is already active. If processing occurs without the use of transactions, the database entries made as a result of processing the Apple event could be lost if an active transaction is later cancelled.

## Quick Start

The open architecture of the INF\_FILL external provides control and flexibility to your 4th DIMENSION application. As explained earlier, you can write global procedures to validate data, control access to the different files in your database, and customize the processing that occurs when completed forms are submitted or lookups requested. If these issues are not important to your application, the modifications necessary to enable forms submission and lookups are simple and straightforward.

To enable forms submission and lookups with Informed, simply add the global procedures listed below to your application. These procedures can be found in the text file named "Fill Ext. Procs.TEXT".

---

**Procedure: STARTUP**

This code should be added to your application's STARTUP procedure. For 4th DIMENSION v2.2, it calls AddProc to ensure that the global procedure ProcessAllAEs is called once every 60 ticks during idle time. For 4th DIMENSION v3.x or the 4D SERVER, the procedure invokes a new process which continually calls ProcessAllAEs. The global variable ErrorFound is also initialized to False.

For 4th DIMENSION version 2.2:

```
AddProc ("ProcessAllAEs";60)
```

For 4th DIMENSION version 3.x or the 4D SERVER:

```
C_LONGINT (pid)
pid := New Process ("AEPollProcess";96000;"CallProAllAEs")
```

---

**Procedure: AEPollProcess**

This procedure is required only if you're using version 3.x of 4th DIMENSION or the 4D SERVER.

```
While (True)
    ProcessAllAEs
    Delay Process (current process;60)
End while
```

---

**Procedure: ProcessAllAEs**

This procedure is called continually to process any Apple events in the queue by calling either ProcessInsert or ProcessLookup repeatedly until the queue is empty.

```
C_INTEGER ($I;$theType)
InfAECOUNT ($I)
While (Not($I=0))
    InfAERead ($theType)
    Case of
        : ($theType=1)
            ProcessInsert
        : ($theType=2)
            ProcessLookup
    End case
    InfAECOUNT ($I)
End while
```

---

**Procedure: ProcessInsert**

This procedure is called to process an insert Apple event. The call to PreInsert is intended to start a transaction and ensure that access to the file or files in which records are being inserted is allowed.

The call to `ValidateIns` is intended to examine that inserted data for any errors. If no errors are detected, `ErrorFound` is cleared to `False`.

```
C_BOOLEAN (ErrorFound)
ErrorFound:=True
If (PreInsert)
  InfDoInsert
  If (ValidateIns)
    ErrorFound:=False
  End if
End if
PostInsert
UnloadRecords
InfSendReply
```

---

#### Function: **PreInsert**

This function is called by `ProcessInsert` before any records are created. A transaction is initiated and the function returns `True`. If you want to deny access to certain files, you should insert code here that calls `InfNewRecCount` and `InfNewRecInfo` to determine which file or files records will be created in. To prevent the insert from occurring, return `False` instead of `True`.

```
START TRANSACTION (*)
$0:=True
```

---

#### Function: **ValidateIns**

This function is called by `ProcessInsert`. It is intended to validate the inserted data. If an error is detected, you should return a `False` result.

```
$0:=True
```

---

#### Procedure: **PostInsert**

This procedure is called by `ProcessInsert` after the data has been inserted. If the insert event was processed successfully, the transaction is validated. Otherwise, the transaction is cancelled.

```
If (ErrorFound)
  CANCEL TRANSACTION
Else
  VALIDATE TRANSACTION
End if
```

---

#### Procedure: **ProcessLookup**

This procedure is called by `ProcessAllAEs` to process a lookup event.

```
InfDoLookup ("DoSearch");
InfSendReply
```



---

**Procedure: DoSearch**

This procedure is called by InfDoLookup. Its parameters specify the file for which the lookup has been requested (\$1), the field in this file to search (\$2), the lookup's match option (\$3), and the search value (\$4). It performs the search and leaves the results in the file's current selection. If you want to deny access to certain files, insert code that examines the file number parameter (\$1) and calls InfErrorMsg or InfErrorCode accordingly.

```
C_INTEGER ($1;$2;$3)
C_STRING (80;$4)

ALL RECORDS(File($1)»)
SEARCH(File($1)»;Field($1;$2)»=$4)
```

---

**Procedure: UnloadRecords**

```
C_INTEGER ($theCount;$theFileNum;$theRecNum)

InfNewRecCount ($theCount)
While ($theCount # 0)
    InfNewRecInfo ($theCount, $theFileNum, $theRecNum)
    UNLOAD RECORD (File ($theFileNum)»)
    $theCount := $theCount - 1
End while
```

## Using the INF\_NS\_Client External

Informed Number Server is an application that generates unique form numbers at the request of other applications. A single Informed Number Server application can be configured to generate form numbers for several different types of forms.

Using Informed Designer, you can link a 'form number' cell on a form to the Number Server application. When the Informed Filler user adds a new form or manually requests a new number, Informed Filler sends a message to the Number Server application requesting the next available form number. The Number Server application replies with the next available number.

If users fill out forms using both Informed Filler and your 4th DIMENSION application, you'll need to modify your application so that it too obtains new form numbers from the Number Server application. The INF\_NS\_CLIENT external contains four external functions that your application can call to select a Number Server application, determine which form numbers are available, and request a new form number.

## Installing the External

Like any 4th DIMENSION external, you install the INF\_NS\_CLIENT external using the 4D External Mover application provided with 4th DIMENSION. The INF\_NS\_CLIENT external can be found in the file ‘Informed NS Client.’

The INF\_NS\_CLIENT external can be installed in any copy of your interpreted or compiled 4th DIMENSION application, in any copy of a 4th DIMENSION development or runtime application, or in your Proc.Ext file.

## Modifying Your 4th DIMENSION Application

Informed Number Server uses Apple events, an IAC (inter-application communications) capability of the Mac OS (version 7 or later) to communicate with other applications. In addition to the computer on which the Number Server application is running, each computer running your 4th DIMENSION application must also be using system software version 7.0 or later.

Integrating your 4th DIMENSION application with Informed Number Server is a simple and straightforward exercise. Rather than generating new form numbers from within your application, a new number is obtained by calling a function in the INF\_NS\_CLIENT external. The external sends an Apple event message over the network to the Number Server application. The Number Server application replies with the next available form number. Other external functions allow your application to select the Number Server application and find out which form numbers are available.

### Identifying the Number Server Application

When your application requests a new form number, information that identifies the computer on which the Number Server application is running must be provided. This information includes the name of the Number Server application, the name of the computer on which it’s running, and the name of the network zone to which the computer is connected.

Although you may already know the names of these items, your application can obtain them by calling the external function ChooseNS.

```
ChooseNS ($theAppName, $theMacName, $theZoneName)
```

ChooseNS displays the Program Linking dialog box. This dialog box contains three scrolling lists from which you can select the correct network zone, computer, and Number Server application.

When you click ‘OK,’ ChooseNS will return the names of the selected items in the three string parameters \$theAppName, \$theMacName, and \$theZoneName. These names are then passed to the external function GetNSNextValue to obtain a new form number (see “Requesting a New Form Number” later). If your network contains only a single zone, the Program Linking dialog box will contain two lists instead of three. The zone name of a single-zone network is always the asterisk character (\*).

ChooseNS is an external function that returns a long integer result. A result of zero indicates that a Number Server application was successfully selected. Non-zero results correspond to the various errors that can occur. For information about error handling, please see “Error Codes” later in this chapter.

## Determining the Available Form Numbers

A single Number Server application can generate form numbers for several different types of forms. Form numbers are configured using the Number Server application. Each form number has a name (‘invoice number,’ or ‘purchase order number,’ for example). When your application requests a new form number, the name of the form number must be provided.

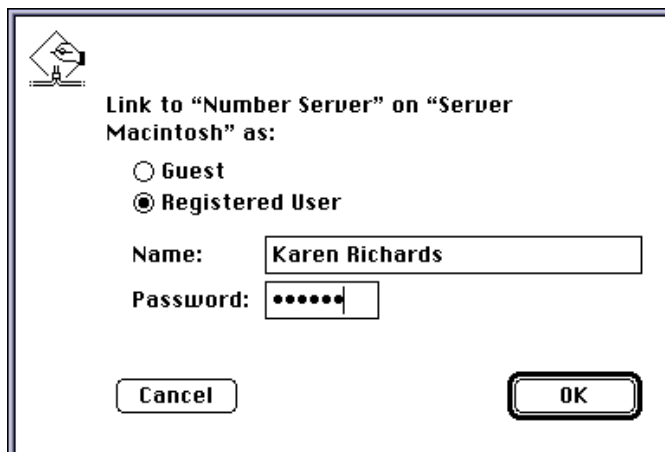
To obtain the names of the form numbers available from a particular Number Server application, your 4th DIMENSION application can call the external functions GetNSNameList and GetNSName.

```
GetNSNameList ($theAppName, $theMacName, $theZoneName, $theCount)
```

```
GetNSName ($theIndex, $theName)
```

GetNSNameList connects to the Number Server application identified by the string parameters \$theAppName, \$theMacName, and \$theZoneName, then requests the names of all available form numbers. If your network consists of a single zone only, pass the asterisk character (‘\*’) in \$theZoneName. Otherwise, pass the zone name. GetNSNameList will return the number of names in the list in the integer parameter \$theCount. The list of names itself is held in the external’s memory area.

If the Number Server application is not running on the same computer as your 4th DIMENSION application (which is commonly the case), the user of your application might be requested to connect to the Number Server application.



As part of the Informed Number Server installation process, the administrator must be sure to enable program linking privileges for the appropriate users. These settings determine if each user is required to connect as a registered user or if guest access is permitted. To connect as a registered user, entry of a name and password is required.

After calling `GetNSNameList`, you can then call `GetNSName` to obtain the name of a specific form number. `$theIndex` specifies which form number; its value should be between 1 and the number returned by `GetNSNameList` in `$theCount`. The name of the form number is returned in the string parameter `$theName`.

`GetNSNameList` and `GetNSName` are external functions that return long integer results. A result of zero indicates that the function completed successfully. Non-zero results correspond to the various errors that can occur. For information about error handling, please see “Error Codes” later in this chapter.

## Requesting a New Form Number

Whenever a new form number is needed, your 4th DIMENSION application should call the external function `GetNSNextValue`.

```
GetNSNextValue ($theAppName, $theMacName, $theZoneName,
                $theNumberName, $increment, $theValue)
```

`$theAppName`, `$theMacName`, and `$theZoneName` identify the Number Server application and the computer on which it’s running. If your network consists of a single zone only, pass the asterisk character (`*`) in `$theZoneName`. Otherwise, pass the zone name.

`GetNSNextValue` will connect to the specified Number Server application and request the next available number for the form number identified by the string parameter `$theNumberName`. Only if you pass 1 in `$increment`, the Number Server application will actually increment the corresponding form number. By passing 0, you can obtain the next available form number without advancing it. This feature is useful for testing purposes. The form number is returned in the string parameter `$theValue`.

Like `GetNSNameList`, calling `GetNSNextValue` may require that the user of your application connect to the Number Server application. Please see the explanation of `GetNSNameList` for more information.

`GetNSNextValue` is an external function that returns a long integer result. A result of zero indicates that the function completed successfully. Non-zero results correspond to the various errors that can occur. For information about error handling, please see “Error Codes” later in this chapter.

## Error Codes

Each of the four Informed Number Server external functions return a long integer result code. A non-zero result can occur due to invalid input parameters or problems with network or operating system components or with the Number Server application itself. The following table lists some of the more common errors, each with a brief explanation.

### Common Error Codes

Error Code	Description
0 No error	The function or procedure completed successfully
-128 userCanceledErr	The user clicked Cancel on the Program Linking dialog box.
-1708 errAEEventNotHandled	An Apple event was not handled by the target application. This error normally indicates that the application specified by the parameters to either GetNSNameList or GetNSNextValue is not a Number Server application.
-1712 errAETimeout	The Number Server application is not responding and a time out occurred
-906 destPortErr	The errors -906, -908, and -915 will occur if the Number Server application is not running on the computer specified by the parameters \$theAppName, \$theMacName, and \$theZoneName.
-908 noSessionErr	
-915 noResponseErr	
-1	The form number specified in \$theNumberName when calling GetNSNextValue does not exist.
-1000 kIndexRangeErr	The value of \$theIndex passed to the GetNSName function is out of range.

.

.



## Appendix A

### Name Prefixes and Suffixes



## Appendix A - Name Prefixes and Suffixes

As described in Chapter 1, “Adding Intelligence to Your Forms,” Informed allows you to store names using the name cell type. With the name cell type, Informed always displays a name according to the cell’s format, even if the person filling out the form enters a name differently. For more information, see “Name” in Chapter 1.

In order to identify the different parts of a name, Informed uses the list of prefixes and suffixes shown in the following tables.

### Prefixes

Full Prefix	Abbreviation	Full Prefix	Abbreviation
Abbot		Congresswoman	Cong.
Admiral	Adm.	Corporal	Cpl.
Airman	Amn.	Count	
Ambassador		Countess	
Archbishop		Dame	
Archdeacon		Doctor	Dr.
Army		Duke	
Assemblyman		Duchess	
Assemblywoman		Excellency	
Assistant		Ensign	
Associate		Father	
Attorney		First	
Baron		Fleet	
Baroness		General	Gen.
Baronet		Governor	Gov.
Bishop		Grade	
Brigadier	Brig.	Her	
Cadet	Cdt.	Highness	
Canon		His	
Cantor		Holiness	
Captain	Capt.	Honorable	Hon.
Cardinal		Judge	
Chairman		Justice	
Chairperson		King	
Cancellor		Knight	
Chaplain		Lady	
Chief		Lieutenant	Lt.
Colonel	Col.	Lord	
Commander	Comm.	M.	
Commodore		Madame	
Congressman	Cong.	Majesty	



## Prefixes (continued)

Full Prefix	Abbreviation	Full Prefix	Abbreviation
Major	Maj.	Rabbi	
Marchioness		Rear	
Marquess		Representative	Rep.
Mayor		Reverend	Rev.
Midshipman		Right	Rt.
Minister		Royal	
Miss		Seaman	
Mister	Mr.	Second	
Most		Secretary	Sec.
Mother		Senator	Sen.
Mrs.		Sergeant	Sgt.
Ms.		Sir	
Navy		Sister	
Of		Specialist	Spec.
Officer		The	
Patriarch		Third	
Pope		Under	
President	Pres.	Very	
Prime		Vice	V.
Prince		Viscount	
Princess		Viscountess	
Private	Pvt.	Warrant	
Professor	Prof.	Yeoman	
Queen			

## Suffixes

Full Suffix	Abbreviation	Full Suffix	Abbreviation
Junior	Jr.	D.D.S.	
Senior	Sr.	D.V.M.	
Esquire	Esq.	J.M.	
First	I	LL.B.	
Second	II	M.A.	
Third	III	M.B.A.	
Fourth	IV	M.D.	
Fifth	V	M.Ed.	
B.A.		M.S.	
B.Comm.		P.Eng.	
B.Ed.		Pharm.	
B.Sc.		Ph.D.	
C.A.		R.E.T.	
C.P.A.		R.N.	





## Appendix B

### Built-in Commands



## Appendix B - Built-in Commands

Built-in commands correspond to the commands and settings that are built into Informed Filler. As described in Chapter 3, “Customizing Menus,” you can add or remove menu items that invoke built-in commands to help create a simpler and more familiar environment for the Informed Filler user. You can also configure buttons to trigger built-in commands. See Chapter 4, “Using Buttons” for more information.

The following table lists each of Informed’s built-in commands and explains which menu items or actions that they correspond to.

### Built-in Commands

Built-in Command	Corresponding Menu Item or Action
About Informed Filler	About command under the Help (Windows) or Apple (Mac OS) menu
Actual Size	Actual Size command under the View menu
Add Columns	Add Columns command under the List menu
Add Record	Add Record command under the Database menu
Add To Record List	Add To Record List command under the Cell menu
Align Column Center	Center menu item in Alignment submenu under List
Align Column Left	Left menu item in Alignment submenu under List
Align Column Right	Right menu item in Alignment submenu under List
Assign Next Value	Assign Next Value command under the Cell menu
Attach File	Attach command under the File menu
Cascade Windows	Cascade command under the Window menu
Check Current Record	Check Current Record item in Spelling submenu under Edit
Check Collected Records	Check Collected Records item in Spelling submenu under Edit
Check Selection	Check Selection item in Spelling submenu under Edit
Clear	Clear command under the Edit menu
Clear Record	Clear Record command under the Database menu
Close	Close command under the File menu
Column Title	Column Title command under the List menu
Copy	Copy command under the Edit menu
Cut	Cut command under the Edit menu
Duplicate Record	Duplicate command under the Database menu
Enlarged Size	Enlarged Size command under the View menu
Export	Export command under the File menu
Extra Choices	Extra Choices command under the Cell menu
Extract Attachment	Extract command under the File menu
Find	Find command under the Database menu
Find Again	Find Again command under the Database menu
Find All	Find All command under the Database menu
First Record	First item in the Go To submenu under Database

## Built-in Commands (continued)

Built-in Command	Corresponding Menu Item or Action
Go To Record	Record command in the Go To submenu under Database - same as double-clicking the record information box to display the Change Record dialog box
Help	Help command under the Cell menu
Import	Import command under the File menu
Insert Date	Insert Date command under the Edit menu
Insert File	Insert File command under the Cell menu
Insert Row	Insert Row command under the Edit menu
Last Record	Last item in the Go To submenu under Database
Log Off Service	Log Off Service command in the Signatures submenu under Edit
Memorize	Memorize command under the Cell menu
New Document	New Document command under the File menu
Next Record	Next item in the Go To submenu under Database - same as clicking the right arrow in the record information box
Omit	Omit command under the Database menu
Omit Others	Omit Others command under the Database menu
Open	Open command under the File menu
Open Mail	Open Mail command under the File menu - available only for Informed mail plug-ins that allow opening mail from within Informed Filler
Page Setup	Page Setup command under the File menu
Paste	Paste command under the Edit menu
Place Note	Place Note command under the Edit menu
Preferences	Preferences command under the Edit menu
Previous Record	Previous item in the Go To submenu under Database - same as clicking the left arrow in the record information box
Print	Print command under the File menu
Quit	Exit (Windows) or Quit (Mac OS) command under the File menu
Record Information	Record Information command under the View menu
Reduced Size	Reduced Size command under the View menu
Register	Register command under the Help (Windows) or Apple (Mac OS) menu
Remove Record	Remove command under the Database menu
Remove Column	Remove Column command under the List menu
Remove Format	Remove Format command in the Formats submenu under List
Remove Row	Remove Row command under the Edit menu
Remove Tag	Remove Tag command in the Tags submenu under Database
Revert	Revert command under the Database menu
Revision Status	Revision Status command under the View menu
Save	Save command under the File menu
Save As	Save As command under the File menu
Save Format	Save Format command in the Formats submenu under List

Built-in Commands (continued)

<b>Built-in Command</b>	<b>Corresponding Menu Item or Action</b>
Select All	Select All command under the Edit menu
Send	Send command under the File menu
Set Value	Set Value command under the Cell menu
Show Attachments	Show/Hide Attachments command under the Window menu
Show Choices	Show/Hide Choices command under the View menu
Show Clipboard	Show/Hide Clipboard command under the Edit menu (Mac OS only)
Show Record List	Show/Hide Record List command under the Window menu
Show Signed Cells	Show/Hide Signed Cells command in the Signatures submenu under Edit
Show Totals	Show/Hide Totals command under the List menu
Sign	Sign command in the Signatures submenu under Edit
Sort	Sort command under the Database menu
Submit	Submit command under the File menu
Tag Records	Tag Records command in the Tags submenu under Database
Template Information	Template Information command under the View menu
Tile Windows	Tile command under the Window menu
Totals	Totals command under the List menu
Tracking Status	Tracking Status command under the View menu
Undo	Undo command under the Edit menu
Verify Signature	Verify command in the Signatures submenu under Edit
Verify Template	Verify Template command in the Signatures submenu under Edit

# 10 Using Functions

In this chapter:

- Overview 10-2
- Function Parameters 10-3
- Function Results 10-6
- Function References 10-7

# 10 Using Functions

One way Informed helps you design intelligent forms is by letting you create cells with calculated values. Cells with calculated values are filled in automatically when you fill out a form with Informed Filler or the test mode of Informed Designer (we'll refer to these common capabilities as features of Informed). A cell's value can be calculated based on the values of other cells. In addition, the cells you fill in can be checked for errors, and the tabbing order of a form can be based on conditions.

## Overview

You calculate or check a cell's value, or specify a cell's next tab position with a formula. Formulas use operators to combine constant values, cell values, and functions to give a single value as a result. Operators are special symbols that produce a new value from other values. The other values (called operands) can be constant values that you enter directly, cell values, or the result of a function. For a complete discussion of formulas and how they work, see Chapter 9.

A *function* performs a predefined calculation using a given set of values, called *parameters*. Functions return a single value called a *result*. For example, you can use the 'Max' function to find the maximum number in a group of numbers.

```
Max (Cell1, Cell2, Cell3)
```

Without the 'Max' function, you'd have to create a formula that compares the numbers in Cell1, Cell2, and Cell3 individually. You can use a function, such as 'Max,' in a formula anywhere you can use a constant or a cell value, provided the type of the function's result is appropriate.

Function name

Choose (Cell1, "hello", 99)

Parameters

Commas

Parentheses

In this chapter you'll learn about functions, function parameters, and function results. A summary of Informed's functions, grouped into general categories, is followed by a detailed description of each function. For information about entering functions in formulas, see "Calculations," "Check Formulas," and "Conditional Tabbing" in Chapter 1.



## Function Parameters

When you use a function, you must supply it the values that it needs to calculate its result. These values are called *parameters*. A function's parameters always appear immediately after the function's name. You enclose parameters in parentheses and separate them with commas. For example, the following function calculates the sum of three parameters: 2, 4, and 8.

```
Sum (2, 4, 8)
```

The number of parameters that you supply to a function depends on the particular function. Some functions don't require any parameters. Other functions use a specific number of parameters. For example, the 'Today' function, which returns the current date, doesn't have any parameters, whereas the 'AddDays' function requires two parameters.

```
Today  
AddDays (PurchaseDate, 7)
```

If you use a function that doesn't have any parameters, you enter only the function's name. The parentheses are not required.

Some functions allow optional parameters. For example, the 'PMT' function uses an optional parameter to indicate when payments are made on an investment (either at the beginning or end of each period). If you don't supply an optional parameter, the function will calculate its result using a default parameter value.

In the formula below, the parameters to the 'Sum' function are all constant, or unchanging, values.

```
Sum (2, 4, 8)
```

You can use other kinds of parameters, such as cell names, other functions, or complete formulas with operators and operands. Consider the function shown below.

```
Sum (Cell1, 99 + Cell2, MakeList(Cell3), Sqrt(Cell4))
```

This function adds the following parameters:

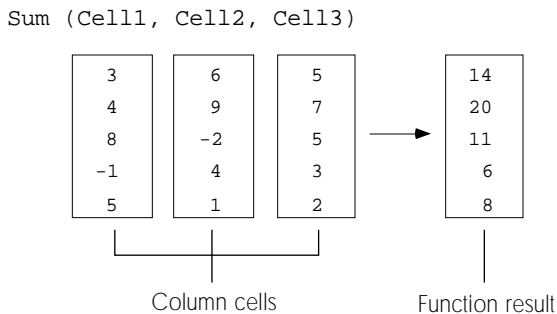
- the value in 'Cell1'
- a formula that adds 99 to the value in 'Cell2'
- the values in the column cell, 'Cell3'
- the result of the 'SQRT' function.

When you use a cell, formula, or function as a parameter, Informed Filler first calculates its value and then uses that result as the parameter value.

The type of a parameter value should match the type expected by the function. For example, all parameters to the 'Sum' function should be numbers. If the type of a parameter's value is not appropriate, Informed Filler will try to convert the value to the correct type. If the value can't be converted, the empty value is used instead. For more information about type compatibility, see "Type Conversion" in Chapter 9.

## Column cell parameters

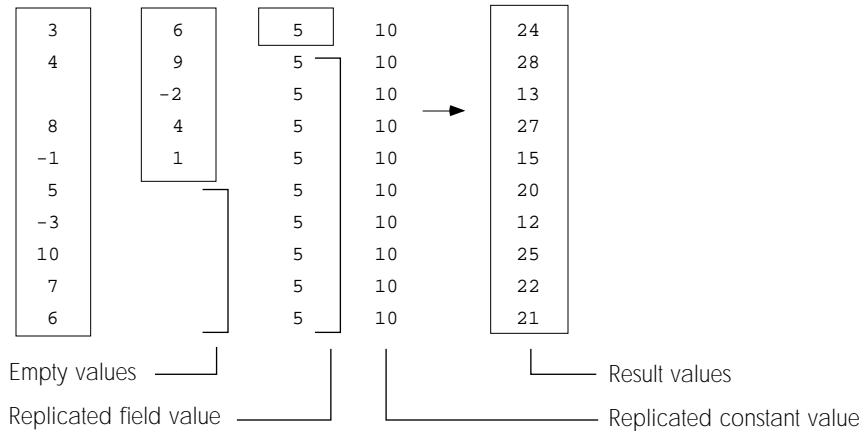
When you use a column cell as a function parameter, the function's result will also be a column. That is, the result will consist of multiple values. For example, if the SUM function were used to add four column cells, each containing five rows, the function's result would also be a column consisting of seven rows. Each row would contain the sum of the corresponding rows in the four column cells.



If the column cells in the above example were of different heights, the function's result would contain as many values as the longest column. Informed would insert empty values in the missing rows of the shorter column cells.

If you use column cells along with field cells or constants as parameters, the field cell and constant values are applied to each row of the column cells. The following figure illustrates how the 'Sum' function would add two column cells with a field cell and a constant.

```
Sum (Column1, Column2, Field1, 10)
```



If you want the values in the rows of a column cell to be treated as though they were supplied as individual parameters, use the 'MakeList' function. This function separates the values in each row of a column cell into separate parameter values. You can use 'MakeList' only with functions that allow any number of parameters. For example, to calculate the sum of the rows in a column cell called 'Extension' plus the shipping charge in 'Shipping,' you could use the following formula.

```
Sum (MakeList (Extension), Shipping)
```

If 'Extension' contains four rows with the values 4.99, 5.95, 2.99, and 10.00, and 'Shipping' contains the value 5.00, the above formula is equivalent to:

```
Sum (4.99, 5.95, 2.99, 10.00, 5.00)
```

If you didn't use the 'MakeList' function, the result of the 'Sum' function would be a column with four rows, each containing the sum of the corresponding row in 'Extension' plus the value of 'Shipping.' For more information about the 'MakeList' function, see the function reference later in this chapter.

If in a place where a list of values is normally expected, a single column cell is given instead, Informed will automatically apply the 'MakeList' function to turn the column cell into a list of individual values. The functions below each accept a list of values. In both cases, 'Column1' is a column cell.

```
Sum (Column1)
Choose (Cell1, Column1)
```

The Choose function accepts two or more parameters. The first is a single value cell. This parameter is followed by one or more parameters forming a variable length list. If you include a column cell as one of these parameters, Informed applies the 'MakeList' function. Informed interprets these functions as though you had typed:

```
Sum (MakeList (Column1))  
Choose (Cell1, MakeList (Column1))
```

## Function Results

Each Informed function returns a result of a particular type. You can use a function in a formula anywhere you can use a constant or cell value with the same type. When you use a function in a formula, the function's result is calculated and then used as an operand when the formula is evaluated.

You can use a function as the single operand in a formula. The function's result is the result of the formula. For example, the formula below extracts the year from a date.

```
YearOf (ToDate ("March 3, 1991"))
```

You can also use a function as one of many operands in a formula, or as a parameter to another function. The formula below uses the 'SQRT' function as an operand to the multiplication (\*) operator.

```
(5 * Sqrt (Cell1))/100
```

The 'SQRT' function is evaluated and its result is then multiplied by 5.

In the following example, the 'SQRT' function is used to calculate the values of both parameters to the 'Sum' function.

```
Sum (Sqrt (Cell1), Sqrt (Cell2))
```

When you use a function in a formula or as a parameter to another function, the type of the function's result value must match the type expected by the formula or function parameter. For example, since the multiplication operator multiplies numbers, it expects number values as its operands. If the types don't match, Informed will try to convert the parameter or operand value to the correct type. For example, if you use the text constant '5' with the multiplication operator, Informed will convert the text value to the numeric value 5 before multiplying. For more information regarding type compatibility, see "Type conversion" in Chapter 9.

## Function Reference

This section summarizes the Informed functions and gives a detailed description of each function. Each function reference includes:

- the name of the function
- the function's parameters
- a description of the function's parameters and the calculation that the function performs
- example uses of the function
- any related functions.

In the function parameter list, optional parameters are enclosed in square brackets ([ ]). The square brackets are there only to tell you that the parameter is optional; don't type them. If a function allows any number of parameters, the parameter list will end with an ellipsis (...). All of the parameters that you supply must be separated with commas and they must be the appropriate type.

Each function description gives several examples of the function's use. The examples use different kinds of parameters: constant values, cell references, other function references, and formulas. When a cell reference is used in an example, the example gives the assumed value of the cell. The result of each example appears to the right of an arrow (→). Each result is shown in a common format that's appropriate for the function's result type.

The following lists show a summary of Informed's functions grouped by category. A detailed description of each function follows.

### Mathematical Functions

Fact (number)  
 Sign (number)  
 SQRT (number)  
 Inv (number)  
 Int (number)  
 Frac (number)  
 Round (number, decimals)  
 Ceiling (number)  
 Floor (number)  
 Abs (number)  
 Markup (cost, selling)  
 Margin (cost, selling)  
 Convert (number, fromUnit, toUnit)  
 ConvertTo (value, toUnit)  
 Trunc (number, decimals)  
 Sum (number1, number2, ...)

## Mathematical Functions (continued)

Min (number1, number2, ...)  
Max (number1, number2, ...)  
RunningTotal (startValue, columnCell)  
Random (min, max)  
NumForm (number, format, currency)

## Statistical Functions

Count (value1, value2, ...)  
Mean (number1, number2, ...)  
Median (number1, number2, ...)  
GMean (number1, number2, ...)  
HMean (number1, number2, ...)  
Range (number1, number2, ...)  
Var (number1, number2, ...)  
PVar (number1, number2, ...)  
StDev (number1, number2, ...)  
PStDev (number1, number2, ...)  
SumSq (number1, number2, ...)

## Trigonometry Functions

Sin (number)  
Cos (number)  
Tan (number)  
ASin (number)  
ACos (number)  
ATan (number)

## Logarithm Functions

Log (number)  
ALog (number)  
LogN (number, base)  
ALogN (number, base)  
Ln (number)  
Ln1 (number)  
Exp (number)  
Exp1 (number)

## Boolean Functions

AnyOf (boolean1, boolean2, ...)  
AllOf (boolean1, boolean2, ...)  
OneOf (boolean1, boolean2, ...)

## Boolean Functions (continued)

Choose (index, value1, value2, ...)  
Member (target, value1, value2, ...)  
WhichMember (target, value1, value2, ...)  
IsEmpty (cell)  
Between (value, startValue, endValue)  
Within (value, startValue, endValue)  
IFT (boolean, trueValue)  
IFTE (boolean, trueValue, falseValue)

## Text Functions

Length (text)  
Concat (text1, text2, ...)  
Mid (text, start, length)  
Left (text, length)  
Right (text, length)  
Insert (text, start, subText)  
Delete (text, start, count)  
Replace (text, start, count, subText)  
Pos (subText, text)  
BeginsWith (text, subText)  
EndsWith (text, subText)  
Contains (text, subText)  
Repeat (text, count)  
Upper (text)  
Lower (text)  
Trim (text)  
UpperFirst (text)  
UpperWords (text)  
Tokenize (text, delimiter)  
TransLiterate (text, source, destination)  
ASCIIChar (number)  
ASCIICode (character)  
CharForm (text, format, fromLeft, default)

## Date Functions

Today  
DayOf (date)  
MonthOf (date)  
YearOf (date)  
MonthName (monthNumber)  
MonthAbbrev (monthNumber)  
DayOfWeek (date)  
DayOfYear (date)

## **Date Functions (continued)**

WeekOfYear (date)  
DayName (dayNumber)  
DayAbbrev (dayNumber)  
AddDays (date, number)  
AddMonths (date, number)  
AddYears (date, number)  
MakeDate (day, month, year)  
LastDayOfMonth (date)  
WorkDays (date1, date2, mask)  
DateForm (date, format)

## **Time Functions**

Now  
HourOf (time)  
MinuteOf (time)  
SecondOf (time)  
AddSeconds (time, number)  
AddMinutes (time, number)  
AddHours (time, number)  
MakeTime (hour, minute, second)  
TimeSpan (date, time)  
TimeForm (time, format)

## **Name Functions**

PrefixCount (name)  
MiddleCount (name)  
MiddleInitialsOf (name)  
SuffixCount (name)  
PrefixOf (name, number)  
FirstOf (name)  
FirstInitialOf (name)  
MiddleOf (name, number)  
LastOf (name)  
LastInitialOf (name)  
SuffixOf (name, number)  
NameForm (name, format)

## **Spell Functions**

NumberTH (number)  
SpellNumber (number)  
SpellNumberTH (number)  
SpellCurrency (number, decimals)



## Amortization Functions

PV (fv, pmt, rate, term[, BEGIN or END])  
FV (pv, pmt, rate, term[, BEGIN or END])  
PMT (pv, fv, rate, term[, BEGIN or END])  
Rate (pv, fv, pmt, term[, BEGIN or END])  
Term (pv, fv, pmt, rate[, BEGIN or END])  
PPMT (pv, pmt, rate, term, per[, BEGIN or END])  
IPMT (pv, pmt, rate, term, per[, BEGIN or END])  
Principal (pv, pmt, rate, term, per[, BEGIN or END])

## Bond Functions

BondPrice (face, rate, ytm, pmts, yield)  
BondYield (price, face, rate, ytm, pmts)  
PBondPrice (face, rate, yield)  
PBondYield (price, face, rate)

## Depreciation Functions

SLD (cost, salvage, life)  
SLDValue (cost, salvage, life, term)  
SOYD (cost, salvage, life, term)  
SOYDValue (cost, salvage, life, term)  
DBal (cost, salvage, life, term[, factor])  
DBalValue (cost, salvage, life, term[, factor])  
MDBal (cost, salvage, life, term[, factor])  
MDBalValue (cost, salvage, life, term[, factor])

## Cash Flow Functions

NPV (rate, columnCell[, BEGIN or END])

## Choice Functions

Choices (cell)  
ValidChoice (value, cell)

## Page Functions

Page  
PageCount  
Part  
PartLabel (label1, label2, label3, ...)  
PartCount

## Table Functions

Row  
RowCount (columnCell)  
MakeList (value1, value2, ...)  
CollapseList (value1, value2, ...)  
MakeColumn (value1, value2, ...)  
CollapseColumn (value1, value2, ...)  
Column

## Type Conversion Functions

ToDate (value)  
ToBoolean (value)  
ToPicture (value)  
ToSignature (value)  
ToTime (value)  
ToName (value)  
ToNumber (value)  
ToText (value)

## Record Functions

Attachments  
CreationTime  
CreationDate  
ModifyTime  
ModifyDate  
SendTime  
SendDate  
PrintTime  
PrintDate

## Template Functions

TemplateName  
TemplateStatus  
TemplateID  
TemplateRevision  
AuthorName  
AuthorOrg

## General Functions

RegisteredCompany  
RegisteredName  
UserName

## General Functions (continued)

Application

External (externalName, value1, value2, ...)

Platform

---

### ABS (number)

The 'Abs' function returns the absolute value of *number*.

#### Examples

Abs (-4) → 4

Abs (0) → 0

Abs (101) → 101

#### Related Functions

None.

---

### ACOS (number)

The 'Acos' function returns the arccosine of *number*. The arccosine is the angle whose cosine is equal to *number*. *Number* must be in the range -1 to 1. ACOS returns the angle in radians between 0 and  $\pi$ .

#### Examples

Acos (-0.75) → -0.722734 radians

Acos (1) → 0 radians

#### Related Functions

'Acos' is the inverse of the 'Cos' function. 'Asin' and 'Atan' return the arcsine and arctangent, respectively, of a number.

---

### ADDDAYS (date, number)

### ADDMONTHS (date, number)

### ADDYEARS (date, number)

These functions calculate a new date based on an existing *date* and a desired increment or decrement. The 'AddDays' function returns the date calculated by adding *number* days to *date*. The 'AddMonths' function returns the date calculated by adding *number* months to *date*. The 'AddYears' function returns the date calculated by adding *number* years to *date*. For an explanation of dates and date constants, see *Date and Name, date, and time constants*.

**Examples**

```
AddDays ("September 1, 1939", -10) → August 22, 1939
AddMonths ("07/23/55", 12) → 07/23/56
AddDays ("Dec 25, 1999", 7) → January 1, 2000
AddYears ("Thurs, May 30, 1991", -31) → Monday, May 30, 1960"
```

If the current date is September 2, 1989 and *years*, *months*, and *days* contain the values 25, 4, and 8, respectively, then

```
AddYears (AddMonths (AddDays (Today, -days), -months), -years) → Apr 25, 1964
```

**Related Functions**

‘DayOf,’ ‘MonthOf,’ and ‘YearOf’ return the numeric value for the day, month, or year of a date. ‘DayOfWeek,’ ‘DayOfYear,’ and ‘WeekOfYear’ return the numeric value for the weekday, day of year, and week of year of a date.

**ADDDHOURS (time, number)****ADDMINUTES (time, number)****ADDSECONDS (time, number)**

These functions calculate a new time based on an existing *time* and a desired increment or decrement. The ‘AddHours’ function returns the time calculated by adding *number* hours to *time*. The ‘AddMinutes’ function returns the time calculated by adding *number* minutes to *time*. The ‘AddSeconds’ function returns the time calculated by adding *number* seconds to *time*.

**Examples**

```
AddSeconds ("6:45:01 PM", 45) → 6:45:46 PM
AddMinutes ("11:30", -100) → 09:50
AddHours ("23:00:00", 50.5) → 1:30:00
```

If *start* contains the value 8:30 AM and *hoursWorked* contains the value 8.5, then

```
AddHours (start, hoursWorked) → 5:00 PM
```

**Related Functions**

‘HourOf,’ ‘MinuteOf,’ and ‘SecondOf’ return the hour, minute, or second of a time value.

**ALLOF (boolean1, boolean2, ...)**

The ‘AllOf’ function returns the boolean value True if all of its boolean parameters are True. If one or more of its parameters are False, ‘AllOf’ returns False. The parameters can be any formula that returns a boolean value.

**Examples**

```

AllOf (True) → True
AllOf (1 + 2 = 3, True, 3 > 2) → True
AllOf (True, 0 > 1) → False

```

If  $x$  contains the value 1 or if *Cell1* is empty, then

```
AllOf (x ≠ 1, Not IsEmpty (Cell1)) → False
```

**Related Functions**

‘AnyOf’ returns True if any of its parameters are True. ‘OneOf’ returns True if exactly one of its parameters is True.

**ALOG (number)**

The ‘ALog’ function returns the base 10 antilogarithm of *number*. The antilogarithm is the number whose logarithm is *number*. A base 10 antilogarithm is equivalent to 10 raised to the power *number*.

**Examples**

```

Alog (2) → 100
Alog (0) → 1
Alog (-2) → 0.01
Alog (Log (5)) → 5

```

**Related Functions**

‘ALog’ is the inverse of the ‘Log’ function. AlogN’ returns a number’s antilogarithm for a given base.

**ALOGN (number, base)**

The ‘ALogN’ function returns the antilogarithm of *number* using the positive base *base*. The antilogarithm is the number whose logarithm is *number*. A base *base* antilogarithm is equivalent to *base* raised to the power *number*.

**Examples**

```

AlogN (2,10) → 100
AlogN (0,42) → 1
AlogN (-2,10) → 0.01

```

**Related Functions**

‘ALogN’ is the inverse of the ‘LogN’ function. ‘ALog’ returns the base 10 antilogarithm of a number.

---

### ANYOF (boolean1, boolean2, ...)

The ‘AnyOf’ function returns the boolean value True if one or more of its boolean parameters are True. If all its parameters are False, ‘AnyOf’ returns False. The parameters can be any formula that returns a boolean value.

#### Examples

```
AnyOf (False, False, False, True) → True
AnyOf (100 / 10 = 9, 4 * 2 = 7, 0 = 1) → False
AnyOf (1 = 1, False) → True
```

If *x* contains the value 1 or if *Cell1* is empty, then

```
AnyOf (x = 0, Allof (x = 1, False), IsEmpty (Cell1)) → True
```

#### Related Functions

‘Allof’ returns True if all of its parameters are True. ‘OneOf’ returns True if exactly one of its parameters is True.

---

## APPLICATION

The ‘Application’ function returns either “Informed Designer” or “Informed Filler.”

---

### ASCIICHAR (number)

The ‘ASCIIChar’ function returns the ASCII character represented by the value *number*. *Number* must be numeric in the range 0 to 255.

#### Examples

```
ASCIICode (65) → "A"
```

#### Related Functions

‘ASCIICode’ returns the number corresponding to an ASCII character.

---

### ASCIICODE (character)

The ‘ASCIICode’ function returns the number corresponding to the ASCII character given in *character*. If *character* evaluates to a text value with more than one character, ‘ASCIICode’ uses the first character.

### Examples

`ASCIICode ("A")` → 65

### Related Functions

'ASCIIChar' returns the ASCII character of a specified value.

---

## ASIN (number)

The 'ASin' function returns the arcsine of *number*. The arcsine is the angle whose sine is equal to *number*. *Number* must be in the range -1 to 1. 'ASin' returns the angle in radians between  $-\pi/2$  and  $\pi/2$ .

### Examples

`Asin (-0.25)` → -0.2527

`Asin (0.84147)` → 1

### Related Functions

'ASin' is the inverse of the 'Sin' function. 'ACos' and 'ATan' return the arccosine and arctangent, respectively, of a number.

---

## ATAN (number)

The 'ATan' function returns the arctangent of *number*. The arctangent is the angle whose tangent is equal to *number*. 'ATan' returns the angle in radians between  $-\pi/2$  and  $\pi/2$ .

### Examples

`Atan (-0.025)` → -0.02499

`Atan (100)` → 1.5608

`Atan (0)` → 0

### Related Functions

'ATan' is the inverse of the 'Tan' function. 'ACos' and 'ASin' return the arccosine and arcsine, respectively, of a number.

---

## ATTACHMENTS

The 'Attachments' function returns a column value with the names of every attachment for the current record.

---

## AUTHORNAME

The 'AuthorName' function returns the author name from the Template Information dialog box for the current template.

---

## AUTHORORG

The 'AuthorOrg' function returns the author organization from the Template Information dialog box for the current template.

---

## BEGINSWITH (text, subText)

The 'BeginsWith' function is a boolean function. It returns the value True if *text* begins with the group of characters in *subText*. The beginning characters of *text* must match the characters in *subText* exactly. If *text* does not begin with *subText*, 'BeginsWith' returns the value False. If *subText* contains more characters than *text*, 'BeginsWith' returns False.

### Examples

```
BeginsWith ("AAA Pizza", "A") → True
BeginsWith ("XYZ Vacuums", "Xyz") → False
BeginsWith ("tele", "television") → False
BeginsWith ("television", "tele") → True
```

If *title* contains the value "Dr." and *name* contains the value "Dr John Smith", then

```
BeginsWith (name, title) → False
```

### Related Functions

'EndsWith' returns True if a text value ends with a specified group of characters. 'Pos' returns the starting position of a group of characters in a text value. 'Contains' returns True if a text value contains a specified group of characters.

---

## BETWEEN (value, startValue, endValue)

The 'Between' function is a boolean function. It returns the value True if *value* is between *startValue* and *endValue*. All three parameters to the 'Between' function must be the same type. They can be numbers, dates, times, text, or boolean values. If *value* is less than or equal to *startValue*, or if *value* is greater than or equal to *endValue*, then 'Between' returns False; otherwise, 'Between' returns True.

Informed uses the standard comparison operators to compare *value* with *startValue* and *endValue*. See *Comparison operators* for information about how each data type is compared.



**Examples**

Between (5, -1, 99) → True  
 Between (ToDate ("Oct. 3, 1989"), ToDate ("01/01/90"),  
 ToDate ("Jan. 1, 1990")) → False  
 Between (ToTime ("1:30 PM"), ToTime ("15:15:00"),  
 ToTime ("2:30 PM")) → False  
 Between ("actuary", "gymnast", "sensible") → False  
 Between ("actuary", "Gymnast", "sensible") → False

**Related Functions**

'Within' returns True if a value is between two other values, inclusive.

**BONDPRIce (face, rate, ytm, pmts, yield)****BONDYIELD (price, face, rate, ytm, pmts)****PBONDPRIce (face, rate, yield)****PBONDYIELD (price, face, rate)**

Bond prices (*price*) and yields to maturity (*yield*) can be equated using the face value of the bond (*face*), the interest rate shown on the bond (*rate*), the number of years to maturity (*ytm*), and the number of interest payments per year (*pmts*). The equation is as follows:

$$Price = \frac{\frac{rate \times face}{pmts}}{\left(1 + \frac{yield}{pmts}\right)} + \frac{\frac{rate \times face}{pmts}}{\left(1 + \frac{yield}{pmts}\right)^2} + \dots + \frac{\frac{rate \times face}{pmts}}{\left(1 + \frac{yield}{pmts}\right)^{ytm \times pmts}} + \frac{face}{\left(1 + \frac{yield}{pmts}\right)^{ytm \times pmts}}$$

Given *face*, *rate*, *ytm*, *pmts*, and *yield*, the 'BondPrice' function returns the market *price* of a bond. Given *price*, *face*, *rate*, *ytm*, and *pmts*, the 'BondYield' function returns the effective *yield* to maturity of a bond.

The 'PBondPrice' and 'PBondYield' functions return the *price* and the *yield*, respectively, for perpetual bonds. A perpetual bond has no maturity date; payments are made forever. The *ytm* factor in the above equation is effectively infinite and the equation reduces to:

$$Price = \frac{rate \times face}{yield}$$

**Examples**

Suppose a 7% bond with a \$1,000 face value, ten years to maturity, two interest payments per year, and a current market price of \$1040 was sold. The 6.45% yield on the bond is calculated by using the 'BondYield' function as follows:

BondYield (1040.00, 1000.00, 0.07, 10, 2) → 0.0645

The selling price required to attain a desired yield of 8% for the same bond can be calculated using the 'BondPrice' function:

```
BondPrice (1000.00, 0.07, 10, 2, 0.08) → 932.05
```

The market price for a 5.0% perpetual bond with a face value of \$1,000 and a desired yield of 6.0% is:

```
PBondPrice (1000.00, 0.05, 0.06) → 833.33
```

If the same perpetual bond is offered at a price of \$900.00, the effective yield is:

```
PBondYield (900.00, 1000.00, 0.05) → 0.0556
```

### Related Functions

None.

---

## CEILING (number)

The ‘Ceiling’ function returns the next integer greater than or equal to *number*.

### Examples

```
Ceiling (0.0000001) → 1  
Ceiling (14.4) → 15  
Ceiling (-42.0001) → -42  
Ceiling (87.000) → 87
```

### Related Functions

‘Floor’ returns the next integer less than or equal to a number.

---

## CHARFORM (text, format, fromLeft, default)

The ‘CharForm’ function formats the text value *text* using the character format specified in the text parameter *format*. The parameters *format*, *fromLeft*, and *default* correspond directly to the settings on the Cell dialog for character cells. For information about the meaning and use of these parameters, see *Character*.

### Examples

```
CharForm ("1234567", "(###) ###-####", False, "(415)000-0000") → "(415) 123-4567"  
CharForm ("1234567", "(###) ###-####", True, "(415)000-0000") → "(123) 456-7000"  
CharForm ("k735", "AA###", False, "FN000") → "FK735"
```

### Related Functions

‘NumForm’ formats a number, ‘DateForm’ formats a date, ‘NameForm’ formats a name, and ‘TimeForm’ formats a time value.

---

## CHOICES (cell)

The 'Choices' function returns a column value containing the values of the choices for the cell called *cell*. *Cell* must be the name of a cell. You enter the choices for a cell using the Choices command. See "Choices" in Chapter 1 for more information.

### Examples

If the cell called 'Ship Method' has the choices "Federal Express", "UPS", and "US Mail"

```
Choices (Ship Method) → "Federal Express", "UPS", "US Mail"
Member ("UPS", MakeList (Choices (Ship Method))) → True
```

### Related Functions

'ValidChoices' returns True if a specified value is a valid choice.

---

## CHOOSE (index, value1, value2, ...)

The 'Choose' function returns one of its parameters. The parameter returned is selected based on the value of *index*. If *index* is 1, *value1* is returned; if *index* is 2, *value2* is returned, and so on. Any number of values can follow *index* and the values can be of any type. For a value to be returned, *index* must be an integer between 1 and the number of values. If *index* is not between 1 and the number of values, 'Choose' returns the empty value.

### Examples

```
Choose (3, "one", "two", "three", "four") → "three"
Choose (4, "five", 5.0, 3+2, 15/3, "six - one") → 5
Choose (1, True, False, 1, 0, "on", "off") → True
```

### Related Functions

'Member' returns True if a value is a member of a specified list of values.

---

## COLLAPSECOLUMN (value1, value2,...)

This function accepts any number of input parameters (either field or column cells), and outputs a column which includes all of the input values. Empty values are stripped from the output and the other values move up to fill empty rows. Blank text values are considered equivalent to empty values.

### Examples

If the column cell 'Prices' has the values 10.00, 3.50, 11.25, an empty row, and 2.50

```
CollapseColumn (Prices) → a column with values 10.00, 3.50, 11.25, 2.50
```

**Related Functions**

The 'MakeColumn' function.

---

**COLLAPSELIST (value1, value2, ...)**

This function accepts any number of input parameters(either field or column cells), and outputs a list which includes all of the input values. Empty values are stripped from the output. Blank text values are considered equivalent to empty values. The output list is appropriate for functions that accept variable numbers of inputs, such as SUM or MEAN.

**Examples**

If the column cell 'Prices' has the values 10.00, 3.50, 11.25, an empty row, and 2.50

`CollapseList (Prices)` → a list with values 10.00, 3.50, 11.25, 2.50

**Related Functions**

The 'MakeList' function.

---

**COLUMN**

The 'Column' function returns the index of the current cell in its owning table. If the current cell is a field cell, 'Column' returns null.

**Examples**

If 'Column' is the calculation formula for the second column in a table

`Column` → 2

**Related Functions**

The 'Row' function returns the corresponding list of row numbers for the column cell that uses this function.

---

**CONCAT (text1, text2, ...)**

The 'Concat' function returns the concatenation of its parameters. Any number of text parameters can be included. The return value is one text value created by joining the text values of all the parameters.

**Examples**

`Concat (3*5, " equals fifteen is ", 3*5 = 15)` → "15 equals fifteen is True"

`Concat (" 'Mac' has ", Length ("Mac"), " chars.")` → "'Mac' has 3 chars."

`Concat ("(", True, " or ", False, ")")` → "(True or False)"

If *firstName* contains the value “Barbara”, *initial* contains the value “J”, *lastName* contains the value “Pearce” and today’s date is May 1, 1991, then

```
Concat (firstName, " ", initial, " ", lastName) → "Barbara J Pearce"
Concat ("Today's date: ", Today) → "Today's date: 5/1/91"
```

### Related Functions

The concatenation operator (&) concatenates its operands; "A" & "B" is equivalent to ‘Concat’ ("A", "B").

---

## CONTAINS (text, subText)

The ‘Contains’ function is a boolean function. It returns the value True if *text* contains the group of characters in *subText*. The subset of characters in *text* must match the characters in *subText* exactly. If they do not, ‘Contains’ returns the value False. If *subText* contains more characters than *text*, ‘Contains’ returns False.

### Examples

```
Contains ("abcdefghijklmnopqrstuvwxy", "M") → False
Contains ("one two three four", "four") → True
Contains ("one two three four", " four ") → False
Contains ("1 2 3 4", 2*2) → True
```

### Related Functions

‘Pos’ returns the starting position of a specified group of characters in a text value. ‘EndsWith’ returns True if a text value ends with a specified group of characters. ‘BeginsWith’ returns True if a text value begins with a specified group of characters.

---

## CONVERT (number, fromUnit, toUnit)

The ‘Convert’ function converts *number* from one unit of measure to another. *Number* is converted from the unit described in the text value *fromUnit* to the unit described in the text value *toUnit*. *FromUnit* and *toUnit* must be similarly derived from compatible unit classes. The following table lists the units recognized by Informed.

Units of Measure

Unit Class	Name	Description
mass	ct	Carat
	g	Gram
	grain	Grain
	lb	Avoirdupois pound
	lbt	Troy pound
	oz	Ounce

---

## Units of Measure (continued)

Unit Class	Name	Description
mass	ozt	Troy ounce
	slug	Slug
	t	Metric ton
	ton	Short ton
	tonUK	Long ton
length	chain	Chain
	fath	Fathom
	ft	International foot
	ftUS	Survey foot
	in	Inch
	m	Meter
	mi	International mile
	mil	Mil
	miUS	US statute mile
	nmi	Nautical mile
	pt	Point (1/72 in)
	rd	Rod
	yd	Yard
	μ	Micron
area (length*length)	a	Are
	acre	Acre
volume (length*length*length)	bbl	Barrel
	bu	Bushel
	cu	US Cup
	fbm	Board foot
	gal	US gallon
	galC	Canadian gallon
	galUK	UK gallon
	l	Liter
	ozFl	US fluid ounce
	ozUK	UK fluid ounce
	pk	Peck
	pnt	Pint
	qt	Quart
	tbsp	Tablespoon
	tsp	Teaspoon
pressure (mass/length*time*time)	atm	Atmosphere
	bar	Bar
	inHg	Inches of mercury
	inH2O	Inches of water

## Units of Measure (continued)

Unit Class	Name	Description
time	mmHg	Millimeters of mercury
	Pa	Pascal
	psi	Pounds per square inch
	torr	Torr
	d	Day
	h	Hour
	min	Minute
	s	Second
speed (length/time)	yr	Year
	knot	Knot
	kph	Kilometers per hour
angle	mph	Miles per hour
	deg or °	Degrees
	rad or r	Radians
temperature	grad	Grades
	°C	Degree Celsius
	°F	Degree Fahrenheit
	°K	Degree Kelvin
	°R	Degree Rankine

You can convert a value of measure only between different units of the same class. However, in addition to converting between the units of each class listed in the previous table, you can also convert measurements between units that are derived by combining units from other classes. For example, not only can you convert a value from quarts to liters, you can also convert a value from cubic inches to cubic centimeters.

```
Convert (15, 'quart', 'liter')
Convert (15 'in * in * in', 'cm * cm * cm')
```

Since volume is a measurement of cubic length, you can convert any value between any units derived by multiplying three units of length.

**Examples**

```
Convert (-1, "ft/s", "in/s") → -12
Convert (5.5, "ft*ft*ft/s", "gal/min") → 2468.5714
Convert (2.3, "atm", "mmHg") → 1748
```

**Related Functions**

‘ConvertTo’ also converts a measurement value from one unit to another.

---

## CONVERTTO (value, toUnit)

The 'ConvertTo' function converts the number/unit pair *value* to another unit of measure. *Value* is a text value that combines a number and its unit. The number is converted to the unit described in the text value *toUnit*. The number's original unit and *toUnit* must be similarly derived from compatible base units. See 'Convert' for more information.

### Examples

```
ConvertTo ("-1 ft/s", "in/s") → -12
ConvertTo ("100 knot", "mi/h") → 115.1
ConvertTo ("5 slug", "lb") → 160.87
```

### Related Functions

'Convert' also converts a measurement value from one unit to another.

---

## COS (number)

The 'Cos' function returns the cosine of *number*. *Number* must be an angle in radians. The return value is in the range -1 to 1.

### Examples

```
Cos (0) → 1
Cos (Pi) → -1
Cos (9* π/2) → 0
```

### Related Functions

'ACos' is the inverse of the 'Cos' function. 'Sin' and 'Tan' return the sine and tangent, respectively, of a number.

---

## COUNT (value1, value2, ...)

The 'Count' function returns the number of non-empty values in its parameter list. Any number of values can appear in the parameter list.

### Examples

If *salary* is empty and *time* contains the time value 8:50, then

```
Count ("", time) → 2
Count (True, 4*5, salary, time) → 3
Count (salary) → 0
Count (time) → 1
```

If *attendees* is a column cell with 10 rows, 7 of which have been calculated or filled in, then

```
Count (attendees) → 7
```



**Related Functions**

None.

---

**CREATIONDATE**

The 'CreationDate' function returns the creation date for the current record.

---

**CREATIONTIME**

The 'CreationTime' function returns the creation time for the current record.

---

**DATEFORM (date, format)**

The 'DateForm' function formats the date *date* using the format specified in the text parameter *format*. The resulting text value is returned. For an explanation of date formats, see "Date" in Chapter 1.

**Examples**

```
DateForm ("01/01/89", "MONTH DD") → "JANUARY 01"
DateForm ("Thursday, November 2, 1989", "MM-DD-YY")
→ "11-02-89"
DateForm ("Feb 27, 1990", "Dy, Month D, YYYY")
→ "Tue, February 27, 1990"
DateForm ("01/08/89", "M/D/YYYY") → "1/8/1989"
```

**Related Functions**

'CharForm' formats a text value, 'NameForm' formats a name, 'NumForm' formats a number, and 'TimeForm' formats a time value.

---

**DAYABBREV (dayNumber)****DAYNAME (dayNumber)**

The 'DayAbbrev' and 'DayName' functions return the name of the weekday identified by *dayNumber*. *DayNumber* should be an integer between 1 and 7. The first day of the week is Sunday, the second is Monday, and so on. The 'DayName' function returns the full name of the day with the first letter capitalized. The 'DayAbbrev' function returns an abbreviated name consisting of the first three characters of the day name with the first letter capitalized. 'DayName' and 'DayAbbrev' are often used with the 'DayOfWeek' function to obtain the name of the day in a date.

**Examples**

```
DayName (1) → "Sunday"
DayAbbrev (1) → "Sun"
DayName (1+3) → "Wednesday"
DayAbbrev (5) → "Thu"
```

If *birthDate* is a date containing the value May 30, 1960, then

```
DayName (DayOfWeek (birthDate)) → "Monday"
```

### Related Functions

'MonthAbbrev' and 'MonthName' return the names of months. 'DayOfWeek' returns the number of the weekday in a date.

---

## DAYOF (date)

The 'DayOf' function returns the number of the day represented in *date*. The number returned is an integer in the range 1 to 31.

### Examples

```
DayOf (ToDate ("Saturday, April 25, 1964")) → 25  
DayOf (ToDate ("09/12/89")) → 12  
DayOf (ToDate ("12/26/89")) → 26  
DayOf (ToDate ("26/12/89")) → 26
```

If *startDate* contains the date value Aug 20, 1999 and *duration* contains the value 20, then

```
DayOf (AddDays (startDate, duration)) → 9
```

### Related Functions

'MonthOf' returns the number of the month in a date. 'YearOf' returns the number of the year in a date.

---

## DAYOFWEEK (date)

The 'DayOfWeek' function returns an integer identifying the weekday represented in *date*. The integer returned is in the range 1 to 7. Sunday is the first day of the week, Monday is the second, and so on. 'DayOfWeek' is often used with the 'DayName' and 'DayAbbrev' functions to calculate the name of the weekday in a date.

### Examples

```
DayOfWeek (ToDate ("June 17, 1992")) → 4  
DayOfWeek (ToDate ("Mon Apr 20, 1987")) → 2  
DayOfWeek (MakeDate (5, 6, 1889)) → 4
```

If the current year is 1990, then

```
DayOfWeek ("Nov 10") → 7
```

**Related Functions**

'DayOfYear' returns the day of the year in a date. 'DayName' and 'DayAbbrev' return the name of a weekday.

**DAYOFYEAR (date)**

The 'DayOfYear' function returns an integer identifying the day of year represented in *date*. The integer returned is in the range 1 to 366.

**Examples**

```
DayOfYear (ToDate ("01/01/90")) → 1
DayOfYear (ToDate ("Dec 31, 1984")) → 366
DayOfYear (ToDate ("Dec 31, 1985")) → 365
DayOfYear (MakeDate (8, 6, 1989)) → 159
```

If the current year is 1989, then

```
DayOfYear (ToDate ("Apr 25")) → 115
```

**Related Functions**

'DayOfWeek' returns the weekday in a date.

**DBAL (cost, salvage, life, term[, factor])****DBALVALUE (cost, salvage, life, term[, factor])****MDBAL (cost, salvage, life, term[, factor])****MDBALVALUE (cost, salvage, life, term[, factor])**

These functions calculate an asset's depreciation amount and value. The 'DBal' and 'DBalValue' functions use the declining balance depreciation method. Under the declining balance method, the depreciation amount for each period is a constant percentage of the depreciated value of the asset. Since the depreciated value of the asset declines over time, this creates larger depreciation amounts during the early years of the useful life of the asset. The percentage used to depreciate the asset is *factor* times the straight line depreciation rate. See SLD.

If *factor* is omitted, a value of 2.0 is used. *Cost* is the initial cost of the asset, *salvage* is the estimated value of the asset at the end of its useful life, and *life* is the number of time periods in the depreciation lifespan of the asset. *Term* identifies a particular depreciation period.

The 'DBal' function returns the depreciation amount for the period identified by *term*.

The 'DBalValue' function returns the depreciated value of the asset at the end of the period identified by *term*. *Term* should be between 0 and *life*. If *term* is equal to 0, 'DBalValue' returns *cost*. If *term* is equal to *life*, 'DBalValue' returns a value greater than *salvage*.

The ‘MDBal’ and ‘MDBalValue’ functions use the modified declining balance depreciation method. This method is a combination of the declining balance and the straight line depreciation methods. See also SLD. Depreciation is first calculated using the declining balance method. When the straight line depreciation amount on the remaining depreciable cost is less than the declining balance amount, a switch is made to the straight line depreciation method.

The ‘MDBal’ function returns the depreciation amount for the period identified by *term*.

The ‘MDBalValue’ function returns the depreciated value of the asset at the end of the period identified by *term*. *Term* should be between 0 and *life*. If *term* is equal to 0, ‘MDBalValue’ returns *cost*. If *term* is equal to *life*, ‘MDBalValue’ returns *salvage*.

### Examples

If a machine costs \$36,000 and is expected to last 12 years with a \$1,500 salvage value, then the double rate (*factor* equal to 2) declining balance depreciation amounts and values for years 3 and 9 are:

```
DBal (36000, 1500, 12, 3) → 4166.66
DBal (36000, 1500, 12, 9, 2.0) → 1395.408
DBalValue (36000, 1500, 12, 3, 2.0) → 20833.33
DBalValue (36000, 1500, 12, 9) → 6977.04
```

and the corresponding modified declining balance depreciation amounts and values are:

```
MDBal (36000, 1500, 12, 3, 2.0) → 4167
MDBal (36000, 1500, 12, 9) → 1709
MDBalValue (36000, 1500, 12, 3, 2.0) → 20833
MDBalValue (36000, 1500, 12, 9) → 6629
```

### Related Functions

‘SLD’ and ‘SLDValue’ return an asset’s depreciation amount and depreciated value using the straight line method. ‘SOYD’ and ‘SOYDValue’ return an asset’s depreciation amount and depreciated value using the sum-of-years method.

---

## DELETE (text, start, count)

The ‘Delete’ function removes *count* characters from *text* and returns the resulting text value. Characters are removed starting at *start* characters from the beginning of *text*. If *count* is greater than the number of characters from *start* to the end of *text*, all characters from *start* onward are deleted.

### Examples

```
Delete ("Hello there", 2, 5) → "Hthere"
Delete ("Goods and services", 6, 99) → "Goods"
Delete ("(413) 555-1234", 1, 6) → "555-1234"
```

**Related Functions**

'Insert' inserts a group of characters into a text value. 'Replace' replaces a group of characters in a text value with another group of characters.

**ENDSWITH (text, subText)**

The 'EndsWith' function is a boolean function. It returns the value True if *text* ends with the group of characters in *subText*. The ending characters of *text* must match the characters in *subText* exactly. If *text* does not end with *subText*, 'EndsWith' returns the value False. If *subtext* contains more characters than *text*, 'EndsWith' returns False.

**Examples**

```
EndsWith ("Have a nice day.", "day") → False
EndsWith ("tele", "television") → False
EndsWith ("television", "vision") → True
EndsWith ("Number of participants: 15", 3*5) → True
```

**Related Functions**

'BeginsWith' returns True if a text value begins with a specified group of characters. 'Pos' returns the starting position of a group of characters in a text value. 'Contains' returns True if a text value contains a specified group of characters.

**EXP (number)  
EXP1 (number)**

The 'Exp' function returns the value of e raised to the power *number*. e is 2.7182818...; it is the base of the natural logarithm. Like 'Exp,' the 'Exp1' function returns the exponential of *number*; but 'Exp1' returns a more accurate result when *number* is close to zero.

**Examples**

```
Exp (1) → 2.7182818
Exp (-9.87) → 0.0000517
Exp (Ln (12.34)) → 12.34
Exp1 (0.0245) → 0.024803
Exp1 (-0.1) → -0.95162582
```

**Related Functions**

'Exp' and 'Exp1' are the inverses of the natural logarithm functions, 'Ln' and 'Ln1.' The exponentiation (^) operator raises an arbitrary number to a given power.

**EXTERNAL (externalName, value1, value2, ...)**

The 'External' function calls an external function. The name of the external function is given as the first parameter. The remaining parameters (any number) are passed to the external function.

**Examples**

The example below calls the external function named 'MyFunction' and passes the 'Now' function as the single parameter. The external function is evaluated and, in this example, returns the numeric value 15093.

```
External ("MyFunction", Now) → 15093
```

**Related Functions**

None.

**FACT (number)**

The 'Fact' functions returns the factorial of *number*. *Number* must be an integer greater than or equal to zero. The factorial of a number, *n*, is calculated as follows:

$$\text{Fact}(n) = n \times (n-1) \times (n-2) \times \dots \times 1 \quad \text{and} \quad \text{Fact}(0) = 1$$
**Examples**

```
Fact (0) → 1
Fact (11) → 39916800
Fact (6) → 720
```

**Related Functions**

None.

**FIRSTINITIALOF (name)**

The 'FirstInitialOf' function returns a text value containing the initial from the first name in *name*. If *name* contains a first name, the capitalized first letter of the first name is returned. If *name* does not contain a first name, the empty value is returned.

**Examples**

```
FirstInitialOf (ToName ("Mr. Anthony Marner")) → "A."
FirstInitialOf (ToName ("John Jacob Joseph Jones")) → "J."
FirstInitialOf (ToName ("e e cummings")) → "E."
```

If *theName* contains the name value Dr. Anderson, MD, then

```
FirstInitialOf (theName)
```

returns the empty value.

**Related Functions**

'LastInitialOf' returns the last initial of a name. 'MiddleInitialsOf' returns all middle initials of a name. 'FirstOf' and 'LastOf' return the first name and last name, respectively, of a name. 'MiddleOf' returns a given middle name of a name. 'PrefixOf' and 'SuffixOf' return a given prefix or suffix of a name.

---

**FIRSTOF (name)**

The 'FirstOf' function returns a text value containing the first name from *name*. If *name* does not contain a first name, 'FirstOf' returns the empty value.

**Examples**

```
FirstOf (ToName ("Dr. Susan Applehoff")) → "Susan"
FirstOf (ToName ("J S Bach")) → "J"
```

If *herName* contains the name value Ms. Pearce, then

```
FirstOf (herName)
```

returns the empty value.

**Related Functions**

'LastOf' returns the last name of a name. 'MiddleOf' returns a given middle name of a name. 'PrefixOf' and 'SuffixOf' return a given prefix or suffix of a name. 'FirstInitialOf' and 'LastInitialOf' return the initial of the first or last name, respectively, of a name. 'MiddleInitialsOf' returns the middle initials of a name.

---

**FLOOR (number)**

The 'Floor' function returns the next integer less than or equal to *number*.

**Examples**

```
Floor (12.999999) → 12
Floor (-42.01) → -43
Floor (0.001) → 0
Floor (-2.00) → -2
```

**Related Functions**

'Ceiling' returns the next integer greater than or equal to a number.

---

**FRAC (number)**

The 'Frac' function returns the fractional part of *number*.

**Examples**

Frac (-12.25) → -0.25

Frac (42) → 0

Frac (-0.0001) → -0.0001

Frac (3/4) → 0.75

**Related Functions**

‘Int’ returns the integer part of a number.

**FV (pv, pmt, rate, term[, Begin or End])**

The ‘FV’ function returns the future value of an investment for a given present value, payment amount, interest rate, and term. See ‘PV.’

**GMEAN (number1, number2, ...)**

The ‘GMean’ function returns the geometric mean of *number1*, *number2*, and so on. Any number of parameters can be specified. The geometric mean of *N* numbers is the product of the numbers to the  $1/N^{\text{th}}$  power.

Numbers that contain the value 0 are ignored.

**Examples**

GMean (2) → 2

GMean (2, 4, 8) → 4

GMean (30, 33, 33.66, 41.74) → 34.34199

If *entries* is a column cell with three non-empty rows that contain the values 47.8, 0.0001, and 9998, then

GMean (entries) → 3.62894

**Related Functions**

‘Mean’ and ‘Median’ return the arithmetic mean and median, respectively, of a group of numbers. ‘HMean’ returns the harmonic mean of a group of numbers.

**HMEAN (number1, number2, ...)**

The ‘HMean’ function returns the harmonic mean of *number1*, *number2*, and so on. Any number of parameters can be specified. The harmonic mean of *N* numbers is *N* divided by the sum of the reciprocals of the numbers.

$$HMean = \frac{N}{\frac{1}{number\ 1} + \frac{1}{number\ 2} + \dots + \frac{1}{number\ N}}$$



Numbers that contain the value 0 are ignored.

### Examples

```
HMean (2) → 2  
HMean (30, 50) → 37.5  
HMean (0.01, 100, 55.345) → 0.02999
```

If *entries* is a column cell with four non-empty rows that contain the values 2.0, 2.0, 3.0, and 0.5, then

```
HMean (entries) → 1.2
```

### Related Functions

‘Mean’ and ‘Median’ return the arithmetic mean and median, respectively, of a group of numbers. ‘GMean’ returns the geometric mean of a group of numbers.

---

## HOUROF (time)

The ‘HourOf’ function returns the number of the hour represented in *time*. The value returned by ‘HourOf’ is an integer between 0 and 23.

### Examples

```
HourOf ("6:45:01 AM") → 6  
HourOf ("23:00:01") → 23  
HourOf ("00:59:59") → 0  
HourOf ("1:00:00") → 1  
HourOf ("10:34 PM") → 22
```

### Related Functions

‘MinuteOf’ returns the number of the minute in a time value. ‘SecondOf’ returns the number of the second in a time value.

---

## IFT (expr1, expr2)

The ‘IFT’ function is a shortcut to using the ‘If’ statement and returns different results based on different conditions. If *expr1* is True, then *expr2* is returned, otherwise the empty value is returned. The ‘IFT’ function can also be used as an operand in a formula. For more information on the ‘If’ statement, see “The If Statement” in Chapter 9, “Using Formulas.”

### Examples

```
IFT (12>10, "Hello There") → "Hello There"
```

**Related Functions**

The 'IFTE' function.

---

**IFTE (expr1, expr2, expr3)**

The 'IFTE' function is a shortcut to using the 'If-Then-Else' statement and returns different results based on different conditions. If *expr1* is True, *expr2* is returned, otherwise *expr3* is returned. The 'IFTE' function can also be used as an operand in a formula. For more information on the 'If-Then-Else' statement, see "The If Statement" in Chapter 9, "Using Formulas."

**Examples**

```
IFTE (12>10, "Hello There", "GoodBye") → "Hello There"
```

**Related Functions**

The 'IFT' function.

---

**INSERT (text, start, subText)**

The 'Insert' function inserts characters from *subText* into *text* and returns the resulting text value. Characters are inserted immediately after the character that is *start* characters from the beginning of *text*. If *start* is greater than the number of characters in *text*, 'Insert' appends *subText* to the end of *text* and returns the resulting text value.

**Examples**

```
Insert ("Received:", 1, "Omitted:") → "ROmitted:eceived:"
Insert ("pencils", 7, " and pens") → "pencils and pens"
Insert ("555-6945", 0, "(413) ") → "(413) 555-6945"
```

If *description* contains the value "The car is " and *color* contains the value "blue", then

```
Insert (description, Length (description) + 1, color)
→ "The car is blue"
```

**Related Functions**

'Delete' deletes a group of characters from a text value. 'Replace' replaces a group of characters in a text value with another group of characters.

---

**INT (number)**

The 'Int' function returns the integer part of *number*. No rounding is done; *number* is truncated to obtain its integer part.

### Examples

Int (-12.75) → -12

Int (42) → 42

Int (-0.0001) → 0

Int (7/4) → 1

### Related Functions

'Frac' returns the fractional part of a number.

---

## INV (number)

The 'Inv' function returns the inverse of *number*. The inverse of a number, *n*, is  $1/n$ . *Number* can be any number except 0. If *number* is 0, the empty value is returned.

### Examples

Inv (1.0) → 1.0

Inv (23) → 0.04348

Inv (-0.003) → -333.333

Inv (3/4) → 1.333

### Related Functions

None.

---

## IPMT (pv, pmt, rate, term, per[, Begin or End])

The 'IPMT' function returns the interest payment amount on an investment for a given present value, payment amount, interest rate, and term. See 'PV.'

---

## ISEMPTY (value)

The 'IsEmpty' function returns a boolean value. *Value* can be any cell or formula. If *value* is empty, 'IsEmpty' returns True. When you fill out a form, a cell is empty until its value is entered or calculated.

### Examples

If the text cell *destination* is blank and hasn't been filled in or calculated, then

IsEmpty (destination) → True

### Related Functions

None.

---

## LASTDAYOFMONTH (date1)

The 'LastDayOfMonth' function returns a date value which is the last day of the month represented in the date value 'date1.'

### Examples

```
LastDayOfMonth (ToDate("Dec 7, 1996")) → December 31, 1996
```

---

## LASTINITIALOF (name)

The 'LastInitialOf' function returns a text value containing the initial of the last name in *name*. If *name* contains a last name, the capitalized first letter of the last name is returned. If *name* does not contain a last name, the empty value is returned.

### Examples

```
LastInitialOf (ToName ("Mr. Anthony Marner, MD")) → "M."
LastInitialOf (ToName ("John Jacob Joseph Jones")) → "J."
LastInitialOf (ToName ("e e cummings")) → "C."
```

If *theName* contains the name value Bill Smith, then

```
LastInitialOf (theName) → "S."
```

### Related Functions

'FirstInitialOf' returns the first initial of a name. 'MiddleInitialsOf' returns all middle initials of a name. 'FirstOf' and 'LastOf' return the first name and last name, respectively, of a name. 'MiddleOf' returns a given middle name of a name. 'PrefixOf' and 'SuffixOf' return a given prefix or suffix of a name.

---

## LASTOF (name)

The 'LastOf' function returns a text value containing the last name of *name*. If *name* doesn't contain a last name, 'LastOf' returns the empty value.

### Examples

```
LastOf (ToName ("Dr. Susan Applehoff")) → "Applehoff"
LastOf (ToName ("J S Bach")) → "Bach"
```

If *hisName* contains the name value Bill and *herName* contains the name value Brenda Wright, R.N., then

```
LastOf (herName) → "Wright"
LastOf (hisName) → "Bill"
```

**Related Functions**

‘FirstOf’ returns the first name of a name. ‘MiddleOf’ returns a given middle name of a name. ‘PrefixOf’ and ‘SuffixOf’ return a given prefix or suffix of a name.

---

**LEFT (text, length)**

The ‘Left’ function returns the first *length* characters in *text*. *Length* must be greater than or equal to 0. If *length* is greater than or equal to the number of characters in *text*, ‘Left’ returns *text*. If *length* is zero, a text value with no characters (“”) is returned.

**Examples**

```
Left ("Macintosh", 3) → "Mac"
Left ("orange", 10) → "orange"
Left ("Forms Design, A Primer", 0) → ""
Left (2 = 2.0, 1) → "T"
```

If *theName* is a name value containing Cecil Featherstone-Haugh and *maxLength* contains the value 15, then

```
Left (LastOf (theName), maxLength) → "Featherstone-Ha"
```

**Related Functions**

‘Right’ returns the last characters in a text value. ‘Mid’ returns a group of characters from the middle of a text value.

---

**LENGTH (text)**

The ‘Length’ function returns the number of characters in *text*.

**Examples**

```
Length ("John Q. Public") → 14
Length ("") → 0
Length (True) → 4
Length (3*4) → 2
```

If *theName* is a name value containing “Cecil Featherstone-Haugh” and *maxLength* contains the value 15, then

```
Length (LastOf (theName)) <= maxLength → False
```

**Related Functions**

‘Trim’ removes extra spaces from the start and end of a text value.

---

**LN (number)****LN1 (number)**

The 'Ln' function returns the natural logarithm of *number*. The natural logarithm of a number is the base e logarithm of the number (e is 2.7182818...). *Number* must be greater than 0. Like 'Ln,' the 'Ln1' function returns the natural logarithm of *number*; but 'Ln1' returns a more accurate result when *number* is close to zero.

**Examples**

```
Ln (e^2) → 2
Ln1 (0.034) → 0.033434776
Ln (8976.46) → 9.1024
Ln1 (0.6) → 0.47003629
Ln (Exp (-3.4)) → -3.4
```

**Related Functions**

'Ln' and 'Ln1' are the inverses of the 'Exp' and 'Exp1' functions. 'Log' returns the base 10 logarithm of a number. 'LogN' returns a number's logarithm for a given base.

---

**LOG (number)**

The 'Log' function returns the base 10 logarithm of *number*. *Number* must be greater than 0. If *number* is less than or equal to 0, the empty value is returned.

**Examples**

```
Log (1) → 0
Log (100) → 2
Log (0.0432) → -1.3645
Log (ALog (3.5)) → 3.5
```

**Related Functions**

'Log' is the inverse of the 'ALog' function. 'LogN' returns a number's logarithm for a given base. 'Ln' and 'Ln1' return the natural logarithm of a number.

---

**LOGN (number, base)**

The 'LogN' function returns the base *base* logarithm of *number*. Both *number* and *base* must be greater than 0. If *number* or *base* is less than or equal to 0, the empty value is returned.

**Examples**

```
LogN (1, 10) → 0
LogN (64, 4) → 3
LogN (0.0657, 6) → -1.51954
LogN (ALogN (3.5, 23.3), 23.3) → 3.5
```

### Related Functions

'Log' is the inverse of the 'ALog' function. 'Log' returns the base 10 logarithm of a number. 'Ln' and 'Ln1' return the natural logarithm of a number.

---

### LOWER (text)

The 'Lower' function converts all letters in *text* to lower case. The resulting text value is returned.

#### Examples

```
Lower ("Serial #: 146A889X") → "serial #: 146a889x"  
Lower ("") → ""  
Lower (True) → "true"  
Lower ("tHiS iS hArD tO rEaD!!") → "this is hard to read!!"
```

### Related Functions

'Upper' converts a text value to upper case. 'UpperFirst' converts the first letter of the first word of a text value to upper case. 'UpperWords' converts the first letter of all words of a text value to upper case.

---

### MAKECOLUMN (value1, value2, ...)

The 'MakeColumn' function returns a column value that includes all inputs.

#### Examples

If 'Names' is a column cell with the values "Tom", "Sally", and "Glenn"

```
MakeColumn (Names, "Scott", "Joan") →  
a column with the values "Tom", "Sally", "Glenn", "Scott", "Joan"
```

### Related Functions

The 'CollapseColumn' function.

---

### MAKEDATE (day, month, year)

The 'MakeDate' function returns the date represented by the integers *day*, *month*, and *year*.

#### Examples

```
MakeDate (1, 1, 1981) → January 1, 1981  
MakeDate (2*5, 5, 1651) → May 10, 1651  
MakeDate (29, 2, 1988) → February 29, 1988
```

If *startDay* contains the value 3, *startMonth* contains the value 10, *startYear* contains the value 1980, and *term* contains the value 5, then

```
MakeDate (startDay, startMonth, startYear + term)
→ October 3, 1985
```

### Related Functions

‘AddDays,’ ‘AddMonths,’ and ‘AddYears’ create a new date by adding a specified number of days, month, or years to a date. ‘ToDate’ converts a text value to a date.

### Note

When using the ‘MakeDate’ function be sure to add the complete year. For example, type 1, 1, 1993 not 1, 1, 93.

---

## MAKELIST (value1, value2, ...)

This function accepts any number of input parameters (either field or column cells) and outputs a list that includes all of the input values. Empty values are preserved. The output list is appropriate for functions that accept a variable number of inputs, such as SUM or MEAN.

### Examples

If ‘Names’ is a column cell with the values “Tom”, “Sally”, and “Glenn”

```
MakeList (Names, "Scott", "Joan") →
a list with the values "Tom", "Sally", "Glenn", "Scott", "Joan"
```

### Related Functions

The ‘CollapseList’ function.

---

## MAKETIME (hour, minute, second)

The ‘MakeTime’ function returns the time represented by the integers *hour*, *minute*, and *second*.

### Examples

```
MakeTime (23, 59, 59) → 23:59:59
MakeTime (0, 0, 0) → 00:00:00
MakeTime (8, 34, 0) → 08:34:00
MakeTime (1, 1, 1) → 01:01:01
```

### Related Functions

‘AddHours,’ ‘AddMinutes,’ and ‘AddSeconds’ create a new time by adding a specified number of hours, minutes, or seconds to a time.



---

## MARGIN (cost, selling)

The 'Margin' function calculates the profit margin for an item. *Cost* is the original cost of the item and *selling* is the final selling price of the item. The selling price must not be 0. If *selling* price is 0, the empty value is returned. The profit margin for an item is calculated as follows:

$$\text{margin} = \frac{\text{selling} - \text{cost}}{\text{selling}}$$

Multiplying the return value of MARGIN by 100 gives the percentage profit margin for the item.

### Examples

```
Margin (80.00, 100.00) → 0.20
Margin (3.45, 3.00) → -0.15
Margin (0, 2*2.5) → 1
Margin (2345.45, 2345.45) → 0
```

If *finalPrice* contains the value 5.00 and *cost* contains the value 5.50, then

```
Margin (cost, finalPrice) < 0 → True
```

### Related Functions

'Markup' returns the markup for an item, given its cost and selling price.

---

## MARKUP (cost, selling)

The 'Markup' function calculates the markup for an item. *Cost* is the original cost of the item and *selling* is the final selling price of the item. The cost must not be 0. If *cost* is 0, the empty value is returned. The markup for an item is calculated as follows:

Multiplying the return value of 'Markup' by 100 gives the percentage markup for the item.

### Examples

```
Markup (80.00, 100.00) → 0.25
Markup (3.45, 3.00) → -0.13
Markup (2*2.5, 0) → -1
Markup (2345.45, 2345.45) → 0
```

If *finalPrice* contains the value 5.00, *cost* contains the value 4.00, and *minMarkup* contains the value 0.10, then

```
Markup (cost, finalPrice) > minMarkup → True
```

### Related Functions

'Margin' returns the profit margin for an item, given its cost and selling price.

---

### MAX (number1, number2, ...)

The 'Max' function returns the largest number in its list of parameters. Any number of parameters can be specified.

#### Examples

Max (-12, 0, 34.5, -23.43, 99) → 99

Max (1, 1.0, 2/2, -5+6) → 1

Max (-100, -99.5) → -99.5

#### Related Functions

'Min' returns the smallest number in a group of numbers. 'Median' returns the median number of a group of numbers.

---

### MDBAL (cost, salvage, life, term[, factor])

The 'MDBal' function uses the modified declining balance method to calculate an asset's depreciation allowance for a particular period. See 'DBal.'

---

### MDBALVALUE (cost, salvage, life, term[, factor])

The 'MDBalValue' function uses the modified declining balance method to calculate the book value of an asset after a specified number of depreciation periods. See 'DBal.'

---

### MEAN (number1, number2, ...)

The 'Mean' function returns the arithmetic mean of *number1*, *number2*, and so on. Any number of parameters can be specified. The arithmetic mean of *n* numbers is the sum of the numbers divided by *n*.

#### Examples

Mean (2) → 2

Mean (30, 50) → 40

Mean (0.01, 100, 55.345) → 51.785

Mean (2.0, -2.0, 0, -0.5) → -0.125

#### Related Functions

'HMean' and 'Median' return the harmonic mean and median, respectively, of a group of numbers. 'GMean' returns the geometric mean of a group of numbers.

---

## MEDIAN (number1, number2, ...)

The 'Median' function returns the median from *number1*, *number2*, etc. Any number of parameters can be specified. The median of a group of numbers is the value in the group that has as many numbers less than it as it has numbers greater than it. If the number of values in the group is even, the median is the arithmetic mean of the two middle values.

### Examples

```
Median (2, 3, -2) → 2
Median (3, -12, -1, -23, -87, 4, 5) → -1
Median (2, 18, 34, 20) → 19
Median (0, 10) → 5
```

### Related Functions

'HMean' and 'Mean' return the harmonic mean and arithmetic mean, respectively, of a group of numbers. 'GMean' returns the geometric mean of a group of numbers.

---

## MEMBER (target, value1, value2, ...)

The 'Member' function returns a boolean value. If *target* equals any of *value1*, *value2*, and so on, 'Member' returns True. Any number of values can be specified.

### Examples

```
Member (True, 1=4, 5>8, 9<3) → False
Member ("5", 1*5, "6", "8") → True
Member (Pi, 3.1543, π, 4.3522, 3.1234) → True
```

If *answer* is 7, then

```
Member (answer, 1, 2, 3, 5, 7, 11, 13, 17) → True
```

### Related Functions

'Choose' selects a value from a specified list of values.

---

## MID (text, start, length)

The 'Mid' function returns a group of *length* characters from *text* starting at *start* characters from the beginning of *text*. *Length* must be greater than or equal to 0. If *length* is greater than or equal to the length of *text* minus *start*, 'Mid' returns all characters from *start* onward. If *length* is zero, a text value with no characters ("") is returned. If *start* is less than or equal to 0, or greater than the length of *text*, the empty value is returned.

**Examples**

```
Mid ("Macintosh", 1, 3) → "Mac"
Mid ("orange", 2, 10) → "range"
Mid ("Forms Design, A Primer", 12, 0) → ""
Mid (2 = 2.1, 1, 4) → "False"
```

If *phone* contains the value “(408) 555-1616”, then

```
Mid (Phone, Length (Phone) - 7, 3) → "555"
```

**Related Functions**

‘Right’ returns the last characters in a text value. ‘Left’ returns the first characters in a text value.

**MIDDLECOUNT (name)**

The ‘MiddleCount’ function returns the number of middle names in *name*. If *name* contains no middle names, ‘MiddleCount’ returns 0.

**Examples**

```
MiddleCount (ToName ("Harry S Truman")) → 1
MiddleCount (ToName ("J. S. Bach")) → 1
MiddleCount (ToName ("Dr. Wendy George")) → 0
MiddleCount (ToName ("Philip Arthur Charles George Windsor,
B.A. ")) → 3
```

**Related Functions**

‘PrefixCount’ returns the number of prefixes in a name. ‘SuffixCount’ returns the number of suffixes in a name. ‘MiddleOf’ returns a given middle name of a name.

**MIDDLEINITIALSOF (name)**

The ‘MiddleInitialsOf’ function returns a text value containing the initials from the middle names in *name*. The value returned contains the capitalized first letters of all middle names. The returned initials are separated by spaces. If *name* doesn’t contain any middle names, the empty value is returned.

**Examples**

```
MiddleInitialsOf (ToName ("Mr. Anthony Frank Marner")) → "F."
MiddleInitialsOf (ToName ("John Jacob George Jones")) → "J. G."
MiddleInitialsOf (ToName ("e e cummings")) → "E."
```

If *theName* contains the name value Dr. Roger Anderson, MD, then

```
MiddleInitialsOf (theName)
```

returns the empty value.

**Related Functions**

'LastInitialOf' returns the last initial of a name. 'FirstInitialOf' returns the first initial of a name. 'FirstOf' and 'LastOf' return the first name and last name, respectively, of a name. 'MiddleOf' returns a given middle name of a name. 'PrefixOf' and 'SuffixOf' return a given prefix or suffix of a name.

---

**MIDDLEOF (name, number)**

The 'MiddleOf' function returns a text value containing a middle name of *name*. The value of *number* determines which middle name is returned. If *number* is 1, the first middle name is returned; if *number* is 2, the second middle name is returned, and so on. If *number* is 0, then all middle names, separated by spaces, are returned. If *name* does not contain the middle name identified by *number*, 'MiddleOf' returns the empty value.

**Examples**

```
MiddleOf (ToName ("Ms. Susan Patricia Anderson"), 1) → "Patricia"
MiddleOf (ToName ("J S M Bach", 2) → "M"
MiddleOf (ToName ("Philip Arthur Charles George Windsor, B.Sc."), 3) → "George"
MiddleOf (ToName ("Philip Arthur Charles George Windsor"), 0)
    → "Arthur Charles George"
```

If *herName* contains the name value Sandra Susan Joan Wright, R.N., then

```
MiddleOf (herName, MiddleCount (herName)) → "Joan"
```

**Related Functions**

'LastOf' returns the last name of a name. 'FirstOf' returns the first name of a name. 'PrefixOf' and 'SuffixOf' return a given prefix or suffix of a name.

---

**MIN (number1, number2, ...)**

The 'Min' function returns the smallest number in its list of parameters. Any number of parameters can be specified.

**Examples**

```
Min (-12, 0, 34.5, -23.43, 99) → -23.43
Min (1, 1.0, 2/2, -5+6) → 1
Min (-100, -99.5) → -100
```

**Related Functions**

'Max' returns the largest number in a group of numbers. 'Median' returns the median number from a group of numbers.

---

## MINUTEOF (time)

The ‘MinuteOf’ function returns the number of the minute represented in *time*. The number returned is an integer in the range 0 to 59.

### Examples

```
MinuteOf ("6:45:01 AM") → 45  
MinuteOf ("23:00:01") → 0  
MinuteOf ("00:59:59") → 59  
MinuteOf ("10:01 PM") → 1
```

### Related Functions

‘HourOf’ returns the number of the hour in a time value. ‘SecondOf’ returns the number of the second in a time value.

---

## MODIFYDATE

The ‘ModifyDate’ function returns the date the current record was last modified.

---

## MODIFYTIME

The ‘ModifyTime’ function returns the time the current record was last modified.

---

## MONTHABBREV (monthNumber)

## MONTHNAME (monthNumber)

The ‘MonthAbbrev’ the ‘MonthName’ functions return the name of the month identified by *monthNumber*. *MonthNumber* must be an integer between 1 and 12. The ‘MonthName’ function returns the full name of the month with the first letter capitalized. The ‘MonthAbbrev’ function returns an abbreviated name consisting of the first three characters of the month name with the first letter capitalized. ‘MonthName’ and ‘MonthAbbrev’ are often used with the ‘MonthOf’ function to obtain the name of the month in a date.

### Examples

```
MonthName (1) → "January"  
MonthAbbrev (1) → "Jan"  
MonthName (1+3) → "April"  
MonthAbbrev (5) → "May"
```

If *birthDate* is a date containing the value December 30, 1965, then

```
MonthAbbrev (MonthOf (birthDate)) → "Dec"
```

**Related Functions**

'DayName' and 'DayAbbrev' return the name of a day. 'MonthOf' returns the number of the month in a date.

---

**MONTHOF (date)**

The 'MonthOf' function returns the number of the month represented in *date*. The number returned is an integer in the range 1 to 12.

**Examples**

```
MonthOf (ToDate ("Saturday, April 25, 1964")) → 4
MonthOf (ToDate ("09/12/89")) → 9
MonthOf (ToDate ("12/26/89")) → 12
MonthOf (ToDate ("26/01/89")) → 1
```

If *startDate* contains the date value Aug 20, 1999, then

```
MonthOf (startDate) → 8
```

**Related Functions**

'DayOf' returns the number of the day in a date. 'YearOf' returns the number of the year in a date.

---

**NAMEFORM (name, format)**

The 'NameForm' function formats the name *name* using the format specified in the text value *format*. The resulting text value is returned. The value of *format* describes the format of the name returned. The letters 'P,' 'F,' 'M,' 'L' and 'S' represent the name parts *prefix*, *first*, *middle*, *last*, and *suffix* in uppercase form. The corresponding lowercase letters represent the same parts in their abbreviated form. If *format* contains one of these letters, the corresponding name part will be included in the function result. The position of 'L' or 'l' in *format* determines if the surname comes first or last. If this letter is the first character of *format*, the surname comes first, otherwise the surname comes last. For an explanation of name formats, see *Name*.

**Examples**

```
NameForm ("Dr. Pat A. Smith", "PFL") → "Doctor Pat Smith"
NameForm ("Mr. William Jones, Esq.", "LpFMs") → "Jones, Mr. William, Esq."
NameForm ("Susan Anderson", "PfMLS") → "S. Anderson"
NameForm ("A. F. Houston", "FL") → "A. Houston"
```

**Related Functions**

'CharForm' formats a text value. 'DateForm' formats a date. 'NumForm' formats a number. 'TimeForm' formats a time value.

---

## NOW

The ‘Now’ function returns the current time. The current time is returned whenever a cell calculation, default formula, or check formula containing the ‘Now’ function is calculated.

### Examples

If the current time is 1:45:01 PM, then

Now → 13:45:01

HourOf (Now) → 13

MinuteOf (Now) → 45

### Related Functions

The ‘Today’ function returns the current date.

---

## NPV (rate, columnCell[, BEGIN or END])

The ‘NPV’ function calculates the net present value of future cash flows generated from an investment project. The present value of cash to be received in the future is the amount that, if it was invested today, would grow to equal the future sum at the future date. Net present value is the sum of the present values of all cash flows generated (or expended) during a project.

*Rate* is the interest rate that’s used to calculate the present value of the future cash flows. The rows in the column cell called *columnCell* contain the projected cash flows for the project. The first row contains the initial cash flow (usually an expenditure), the second row contains the cash flow for the first period, and so on. Net outflow of cash during a period is represented by a negative number in the corresponding row and net inflow is represented by a positive number. There can be any number of rows in *columnCell*. Rows that contain empty values are not ignored; they’re assumed to contribute no net cash flow for the corresponding period.

‘NPV’ calculates the net present value using the following formula, where *N* is the number of rows in *columnCell* and each *row* represents the value contained in the corresponding row of *columnCell*:

$$NPV = \frac{\text{row 1}}{(1 + \text{rate})^0} + \frac{\text{row 2}}{(1 + \text{rate})^1} + \dots + \frac{\text{row N}}{(1 + \text{rate})^{N-1}}$$

If the word BEGIN (or begin) appears as the last parameter to a function, payments are assumed to occur at the beginning of each period. If END (or end) appears as the last parameter, payments are assumed to occur at the end of each period. If neither BEGIN nor END appear, payments are assumed to occur at the end of each period.



**Examples**

If *discountRate* contains the value 0.10 (i.e. 10%) and *cashFlows* is a column cell with rows that contain the values -10000, 4000, 5000, and 6000, then

```
NPV (discountRate, cashFlows) → 2069.53
```

**NUMBERTH (number)**

The ‘NumberTH’ function returns a text value that consists of *number* followed by the appropriate ordinal suffix. *Number* should be an integer.

**Examples**

```
NumberTH (11) → "11th"
NumberTH (-32) → "-32nd"
NumberTH (21) → "21st"
```

**Related Functions**

‘SpellNumber’ spells out a number. ‘SpellNumberTH’ spells out the ordinal value of a number.

**NUMFORM (number, format, currency)**

The ‘NumForm’ function formats the number *number* using the format specified in the text value *format*. *Currency* is a boolean parameter. If it’s value is True, ‘NumForm’ includes the currency symbol in the formatted number. The resulting text value is returned. The number is formatted according to the formatting rules and options for the number type. For more information, see “Number” in Chapter 1.

**Examples**

```
NumForm (128.89, "0", False) → "129"
NumForm (2754.7, "#,##0.00", False) → "2,754.70"
NumForm (2754.7, "#,##0.00", True) → "$2,754.70"
NumForm (-.56, "##,##0.00 Cr;##,##0.00 Dr", False) → "0.56 Dr"
NumForm (4381.99, "The price is #,##0.00", True) → "The price is $4,381.99"
```

**Related Functions**

‘CharForm’ formats a text value, ‘DateForm’ formats a date, ‘NameForm’ formats a name, and ‘TimeForm’ formats a time value.

**ONEOF (boolean1, boolean2, ...)**

The ‘OneOf’ function returns the boolean value True if exactly one of its boolean parameters is True. If no parameter is True or if more than one parameter is True, ‘OneOf’ returns False. The parameters can be any formula that returns a boolean value.

**Examples**

```

OneOf (True) → True
OneOf (1 + 2 = 3, False, 3 > 2) → False
OneOf (True, 0 > 1) → True
OneOf (5 < 3, 1 + 1 = 6, 8 - 3 = 4) → False

```

If *x* contains the value 1 and if *Cell1* is empty, then

```
OneOf (x ≠ 1, IsEmpty (Cell1)) → True
```

**Related Functions**

‘AnyOf’ returns True if any of its parameters are True. ‘AllOf’ returns True if all of its parameters are True.

**PAGE**

When used in a cell’s formula, the ‘Page’ function returns the number of the page on which the cell appears. The number returned is in the range 1 to 99. Page numbering starts at the first numbered page in a form. The work and master pages are not included. ‘Page’ always returns the correct page number, even when pages are added and removed.

**Examples**

If Cell5 appears on the third page of a form, and is calculated as

```
Concat ("Page ", Page)
```

then the value of Cell5 will be “Page 3”.

**Related Functions**

‘PageCount’ returns the total number of pages in a form. ‘Part’ returns the part number of a printed page. ‘PartLabel’ returns a text label according the part of a page currently printing. ‘PartCount’ returns the total number of parts for a page on which a cell appears.

**PAGECOUNT**

The ‘PageCount’ function returns the total number of pages in the form. The number returned is in the range 1 to 99. The work and master pages are not included in the count.

**Examples**

If a form contains 5 numbered pages, then for any cell on any page,

```
Concat ("Total pages: ", PageCount) → "Total pages: 5"
```

**Related Functions**

When used in a cell's formula, the 'Page' function returns the number of the page on which the cell appears. 'Part' returns the part number of a printed page. 'PartLabel' returns a text label according to the part of a page currently printing. 'PartCount' returns the total number of parts for a page on which a cell appears.

---

**PART**

The 'Part' function returns the part number of the page currently being printed. When you print a form, Informed changes the value of 'Part' each time a different part of a page is printed. 'Part' is used to calculate a cell's value based on the part of a page. The number returned is in the range 1 to 99. For a description of the Multipart command and multipart pages, see *Multipart pages*.

**Examples**

If a page of a form contains three parts: the original, a customer copy, and an accounting copy, you can calculate a cell using the 'Part' function as shown below.

```
Choose (Part, "Original", "Customer Copy", "Accounting Copy")
```

**Related Functions**

'PartLabel' returns a text label according to the part of a page currently printing. 'PartCount' returns the total number of parts for a page on which a cell appears. When used in a cell's formula, the 'Page' function returns the number of the page on which the cell appears. 'PageCount' returns the total number of pages in a form.

---

**PARTCOUNT**

When used in a cell's formula, the 'PartCount' function returns the total number of parts for the page containing the cell. The number returned is in the range 1 to 99. For a description of the Multipart command and multipart pages, see *Multipart pages*.

**Examples**

If the following formula is used to calculate a cell on a page that contains three parts, and the second part is currently printing, then

```
Concat ("Part ", Part, " of ", PartCount) → "Part 2 of 3"
```

**Related Functions**

'Part' returns the part number of a printed page. 'PartLabel' returns a text label according to the part of a page currently printing. When used in a cell's formula, the 'Page' function returns the number of the page on which the cell appears. 'PageCount' returns the total number of pages in a form.

---

### **PARTLABEL (label1, label2, label3, ...)**

When used in a cell's formula, the 'PartLabel' function returns the text value which corresponds to the current part that's printing. If part 1 is printing, *label1* is returned; if part 2 is printing, *label2* is returned, and so on. 'PartLabel' is used to calculate a cell's value based on the part of a page. For a description of the Multipart command and multipart pages, see *Multipart pages*.

#### **Examples**

If a page of a form contains three parts: the original, a customer copy, and an accounting copy, you can calculate a cell using the 'PartLabel' function as shown below.

```
PartLabel ("Original", "Customer Copy", "Accounting Copy")
```

#### **Related Functions**

'Part' returns the part number of a printed page. 'PartLabel' returns a text label according the part of a page currently printing. 'PartCount' returns the total number of parts for a page on which a cell appears. When used in a cell's formula, the 'Page' function returns the number of the page on which the cell appears. 'PageCount' returns the total number of pages in a form.

---

### **PBONDPRICE (face, rate, yield)**

The 'PBondPrice' function returns the price of a perpetual bond with face value *face*, interest rate *rate*, and yield *yield*. See 'BondPrice.'

---

### **PBONDYIELD (price, face, rate)**

The 'PBondYield' function returns the yield from a perpetual bond with face value *face*, interest rate *rate*, and price *price*. See 'BondPrice.'

---

### **PLATFORM**

The 'Platform' function returns either "Windows" or "Mac OS."

---

### **PMT (pv, fv, rate, term[, BEGIN or END])**

The 'Pmt' function returns the payment amount on an investment for a given present value, future value, interest rate and term. See 'PV.'

---

### **POS (subText, text)**

The 'Pos' function returns the starting position of the first occurrence of *subText* in *text*. If no portion of *text* exactly matches the group of characters in *subText*, 'Pos' returns 0. The first character in

*text* numbered 1, the second is numbered 2, and so on. If *subText* contains no characters or if the length of *subText* is greater than the length of *text*, 'Pos' returns 0.

### Examples

```
Pos ("nice", "Have a nice day.") → 8
Pos (3*5, "Product Lifespan: 15 years") → 19
Pos ("for", "Informed for forms.") → 3
Pos ("television", "tele") → 0
```

If *adj* contains the value "Valued" and *salut* contains the value "Dear Customer", then

```
Insert (salut, Pos ("Customer", salut)-1, adj) → "Dear Valued Customer"
```

### Related Functions

'Contains' returns True if a text value contains a specified group of characters. 'Insert' inserts a group of characters into a text value. 'Replace' replaces one group of characters in a text value with another group of characters. 'Delete' deletes a group of characters from a text value.

## PPMT (pv, pmt, rate, term, per[, BEGIN or END])

The 'PPMT' function returns the principal payment amount on an investment for a given present value, payment amount, interest rate, term and period. See 'PV.'

## PREFIXCOUNT (name)

The 'PrefixCount' function returns the number of prefixes in *name*. If *name* contains no prefixes, 'PrefixCount' returns 0.

### Examples

```
PrefixCount (ToName ("Harry S Truman")) → 0
PrefixCount (ToName ("Hon. Mr. John S. Edgemont")) → 2
PrefixCount (ToName ("Dr. Wendy George")) → 1
PrefixCount (ToName ("Mr. Philip Arthur Charles George Windsor, B.Sc.)) → 1
```

### Related Functions

'MiddleCount' returns the number of middle names in a name. 'SuffixCount' returns the number of suffixes of a name.

## PREFIXOF (name, number)

The 'PrefixOf' function returns a text value containing a prefix from *name*. The value of *number* determines which prefix is returned. If *number* is 1, the first prefix is returned; if *number* is 2, the second prefix is returned, and so on. If *number* is 0, then all prefixes, separated by spaces, are returned. If *name* doesn't contain the prefix identified by *number*, 'PrefixOf' returns the empty value.

**Examples**

```
PrefixOf (ToName ("Ms. Susan Patricia Anderson"), 1) → "Ms."
PrefixOf (ToName ("Hon. Mr. John S. Edgemont"), 2) → "Mr."
```

If *herName* contains the name value Ms. Sandra Susan Joan Wright, R.N., then

```
PrefixOf (herName, PrefixCount (herName)) → "Ms."
```

**Related Functions**

‘LastOf’ returns the last name of a name. ‘FirstOf’ returns the first name of a name. ‘MiddleOf’ returns a given middle name of a name. ‘SuffixOf’ returns a given suffix of a name.

**PRINCIPAL (pv, pmt, rate, term, per[, BEGIN or END])**

The ‘Principal’ function returns the principal amount remaining on an investment for a given present value, payment amount, rate, term, and period. See ‘PV.’

**PRINTDATE**

The ‘PrintDate’ function returns the date the current record was last printed.

**PRINTTIME**

The ‘PrintTime’ function returns the time the current record was last printed.

**PSTDEV (number1, number2, ...)**

The ‘PSTDev’ function returns the population standard deviation,  $\sigma$ , of *number1*, *number2*, and so on. ‘PSTDev’ calculates the true standard deviation for data sets constituting entire populations of interest. The value returned is calculated as follows:

$$\sigma = \sqrt{\sigma^2}$$

It’s the positive square root of the population variance. See ‘PVar.’

Any number of parameters can be specified.

**Examples**

```
PStDev (3.74, 3.89, 4.00, 3.68, 3.69) → 0.125
PStDev (9.3455) → 0
PStDev (40, 30, 50, 15, 5) → 16.31
```

If *ages* is a column cell with rows that contain the values 28, 31, 27, 29, 45, and 24, then

$\text{PStDev}(\text{ages}) \rightarrow 6.749$

### Related Functions

‘PVar’ calculates the population variance of a group of numbers. ‘Var’ and ‘STDev’ calculate the sample variance and sample standard deviation, respectively, of a group of numbers. ‘Range’ calculates the range of a group of numbers.

**PV (fv, pmt, rate, term[, BEGIN or END])**

**FV (pv, pmt, rate, term[, BEGIN or END])**

**PMT (pv, fv, rate, term[, BEGIN or END])**

**RATE (pv, fv, pmt, term[, BEGIN or END])**

**TERM (pv, fv, pmt, rate[, BEGIN or END])**

**PPMT (pv, pmt, rate, term, per[, BEGIN or END])**

**IPMT (pv, pmt, rate, term, per[, BEGIN or END])**

**PRINCIPAL (pv, pmt, rate, term, per[, BEGIN or END])**

These functions calculate common financial parameters used in constant payment cash flow analysis. They calculate present value (PV), future value (FV), periodic payment amount (Pmt), interest rate per period (Rate), investment term (Term), principal payment amount (PPMT), interest payment amount (IPmt), and the principal amount remaining on an investment (Principal).

The parameters *pv*, *fv*, *pmt*, *rate*, and *term* correspond to the functions with the same names. *Per* is a number between 1 and *term* that specifies a particular period. If the word *BEGIN* (or *begin*) appears as the last parameter to a function, payments are assumed to occur at the beginning of each period. If *END* (or *end*) appears as the last parameter, payments are assumed to occur at the end of each period. If neither *BEGIN* nor *END* appear, payments are assumed to occur at the end of each period. All of the above functions assume that cash received is represented by a positive number and cash paid out is represented by a negative number.

The ‘PV,’ ‘FV,’ ‘Rate,’ ‘Pmt,’ and ‘Term’ functions use the following equations to calculate their return values:

$$pv = term \times pmt + fv$$

(if rate = 0)

$$pv = pmt \left[ \frac{1 - (1 + rate)^{-term}}{rate} \right] + fv [1 + rate]^{term}$$

(if rate  $\neq$  0)

The ‘Principal’ function calculates the principal amount remaining after *per* periods by finding the future value of the investment after *per* periods. The IPMT function calculates the interest payment

amount for period *per* by multiplying the interest rate per period *rate* by the principal amount remaining at the previous period. The 'PPmt' function calculates the principal payment for period *per* by subtracting the interest payment for the period from the total payment amount for the period, *pmt*.

### Examples

If a savings account contains \$5,000 and earns 12% annual interest (1% monthly interest) and, at the beginning of each month for the next 24 months, deposits of \$200 are made, the amount of money in the account after 24 months is:

FV (-5000, -200, 0.01, 24, BEGIN) → 11797.31

If a loan of \$60,000 is made at an annual interest rate of 18% (monthly interest rate of 1.5%), and it must be paid off in 4 years with monthly payments made at the end of each month, the monthly payment amount is:

PMT (60000, 0, 0.015, 48, END) → -1762.5

and the interest payment amount and principal payment amount for the 5<sup>th</sup> period, and principal amount remaining after the 5<sup>th</sup> period are:

IPMT (60000, -1762.5, 0.015, 48, 5, END) → -847.07

PPMT (60000, -1762.5, 0.015, 48, 5, END) → -915.43

Principal (60000, -1762.5, 0.015, 48, 5, END) → 55556.17

If an investor purchases a business for \$500,000 with the expectation of selling it for \$1,000,000 in 5 years, and the business makes a \$100,000 profit annually, the annual rate of return on the investment is:

Rate (-500000, 1000000, 100000, 5, END) → 0.3087

If the same investor wants a fixed annual return of 30% on the same investment, the required term (in years) before selling the business is:

Term (-500000, 1000000, 100000, 0.3) → 5.28

If the same investor wants a fixed annual return of 20% for a 5 year term on the same investment, the required purchase price is:

PV (1000000, 100000, 0.2, 5, END) → -700938.79

### Related Functions

None.

---

### PVAR (number1, number2, ...)

The 'PVar' function returns the population variance,  $\sigma$ , of *number1*, *number2*, and so on. 'PVar' calculates the true variance for data sets constituting entire populations of interest. The population variance of *N* numbers is calculated as follows:



$$\sigma^2 = \frac{(number\ 1 - \mu)^2 + (number\ 2 - \mu)^2 + \dots + (number\ N - \mu)^2}{N}$$

where  $\mu$  is the arithmetic mean of the numbers:

$$\mu = \frac{number\ 1 + number\ 2 + \dots + number\ N}{N}$$

Any number of parameters can be specified.

### Examples

PVar (3.74, 3.89, 4.00, 3.68, 3.69) → 0.0156

PVar (9.3455) → 0

PVar (40, 30, 50, 15, 5) → 266.0

If *ages* is a column cell with rows that contain the values 28, 31, 27, 29, 45, and 24, then

PVar (ages) → 45.556

### Related Functions

'PSTDev' calculates the population standard deviation of a group of numbers. 'Var' and 'STDev' calculate the sample variance and sample standard deviation, respectively, of a group of numbers. 'Range' calculates the range of a group of numbers.

---

## RANDOM (min, max)

The 'Random' function returns a random number between 'min' and 'max.' The result is a random floating point number that is greater than or equal to 'min' and less than 'max.' If you want to use the 'Random' function to generate a random integer within a range of values (inclusive), use this function:

Int (Random (min, max+1))

### Examples

Random (1, 10) → 6.36652

Int (Random (1, 10)) → 9

### Related Functions

None.

---

## RANGE (number1, number2, ...)

The 'Range' function returns the difference between the highest number and the lowest number in *number1*, *number2*, and so on. Any number of parameters can be specified.

### Examples

Range (-5, 0, 1) → 6

Range (65) → 0

Range (1.01, 0.98, 1.38, 1.24, 1.05, 0.96, 0.99) → 0.42

### Related Functions

'Var' and 'PVar' calculate the sample variance and population variance, respectively, of a group of numbers. 'STDev' and 'PSTDev' calculate the sample standard deviation and population standard deviation, respectively, of a group of numbers.

---

## RATE (pv, fv, pmt, term[, BEGIN or END])

The 'Rate' function calculates the interest rate per period on an investment for a given present value, future value, payment amount, and term. See 'PV.'

---

## REGISTEREDCOMPANY REGISTEREDNAME

The 'RegisteredName' and 'RegisteredCompany' functions return the names of the person and company, respectively, to which the application being used is registered. By using these functions in default formulas, you can have Informed Filler automatically fill in the user's name or company name on the form.

### Examples

Suppose that your form contains a cell called *Initiator* which is the name and company of the user filling out the form. You might use the default formula shown below to automatically fill in the person's name and company. Assume that the software being used is registered to a person by the name of 'John Smith' of the company 'ABC Company.'

Concat (RegisteredName, " of ", RegisteredCompany) → "John Smith of ABC Company"

By using the 'RegisteredName' and 'RegisteredCompany' functions (instead of typing the names directly in the default formula), the form can be filled out by different people using different applications provided that each person is using an application that was personally registered to him or her.

---

## REPEAT (text, count)

The ‘Repeat’ function returns a text value created by repeating *text*, *count* times. *Count* should be greater than zero. If *count* is zero, ‘Repeat’ returns a text value with no characters (“”).

### Examples

```
Repeat ("-- ", 10) → "-- -- -- -- -- -- -- -- -- "
Repeat (" ", 23) → " "
Repeat ("x", 3) → "xxx"
```

---

## REPLACE (text, start, count, subText)

The ‘Replace’ function replaces *count* characters from *text* with *subText*. Characters are replaced starting at *start* characters from the beginning of *text*. If *count* is greater than the number of characters from *start* to the end of *text*, all characters from *start* onward are replaced. If *start* is greater than the number of characters in *text*, ‘Replace’ appends *subText* to *text* and returns the resulting text value. If *count* is zero, *text* is returned.

### Examples

```
Replace ("Intramural", 0, 2, "Ex") → "Extramural"
Replace ("January", 4, 3, "itor") → "Janitory"
Replace ("Hello", 0, 99, "Goodbye") → "Goodbye"
```

If *firstName* contains the value “Walter” and *greeting* contains the value “Hello name,” then

```
Replace (greeting, 7, 4, firstName) → "Hello Walter,"
```

### Related Functions

‘Delete’ deletes a group of characters from a text value. ‘Insert’ inserts a group of characters into a text value. ‘Pos’ returns the position of a specified group of character in a text value.

---

## RIGHT (text, length)

The ‘Right’ function returns the last *length* characters from *text*. *Length* must be greater than or equal to 0. If *length* is greater than or equal to the number of characters in *text*, ‘Right’ returns *text*. If *length* is zero, a text value with no characters (“”) is returned.

### Examples

```
Right ("Macintosh", 3) → "osh"
Right ("orange", 10) → "orange"
Right ("Forms Design, A Primer", 0) → ""
Right (2 = 2.0, 2) → "ue"
```

If *phoneNumber* contains the value "(413) 555-6732" and *suffixLength* contains the value 4, then

```
Right (phoneNumber, suffixLength) → "6732"
```

**Related Functions**

'Left' returns the first characters in a text value. 'Mid' returns a group of characters from the middle of a text value.

---

**ROUND (number, decimals)**

The 'Round' function rounds *number* to *decimals* decimal places. *Decimals* should be greater than or equal to zero. If *Decimals* is greater than the number of digits in the fractional part of *number*, 'Round' returns *number*.

**Examples**

Round (12.234, 0) → 12  
Round (-12.235, 2) → -12.23  
Round (-12.234, 2) → -12.23  
Round (-4.1355, 3) → -4.135  
Round (-8.8, 0) → -9  
Round (4.1355, 5) → 4.1355

**Related Functions**

'Ceiling' returns the next integer greater than or equal to a number. 'Floor' returns the next integer less than or equal to a number.

---

**ROW**

The 'Row' function returns the corresponding list of row numbers for the column cell that uses this function (in its calculation, default, or check formula). Use the 'Row' function to fill in the row numbers in a column cell.

**Examples**

If the column cell called *Item* contains 5 rows, the 'Row' function will return the numbers 1, 2, 3, 4, and 5 if it's used in the calculation, default, or check formula of the cell.

**Related Functions**

'RowCount' returns the total number of rows in a column cell.

---

**ROWCOUNT (columnCell)**

The 'RowCount' function returns the total number of rows in the column cell called *columnCell*. *ColumnCell* must be the name of a column cell.

**Examples**

If the column cell called *Item* contains 10 rows, then

RowCount (Item) → 10

### Related Functions

'Row' returns the row numbers of a column cell.

## RUNNINGTOTAL (startValue, columnCell)

The 'RunningTotal' function automatically calculates the running total of a column cell on a form. The *columnCell* parameter must be a numeric column cell on your form. *StartValue* is the starting value of the running total. The result of 'RunningTotal' is a column of values which corresponds to the accumulative total of the column cell *columnCell* with an initial starting value of *startValue*.

### Examples

Suppose that you're creating a bank statement form to keep track of your bank account balance. This form might look like the one shown in the following illustration.

World National Bank		Statement of Account		
<b>N</b> <b>A</b> <b>M</b> <b>E</b>	John Smith	Statement Date		
	12345 - 123 Street	3/2/90		
	Somewhere, Some Place	Previous Balance		
	12345	24,991.70		
Account Details				
Date	Description	Debit	Credi	Balance
1/10/90	Check #110	219.90		24,771.80
1/24/90	Check #110	1,990.00		22,781.80
1/28/90	Deposit		1,570.15	24,351.95
2/7/90	Check #112	400.00		23,951.95
2/15/90	Deposit		1,030.54	24,982.49
2/28/90	Deposit		1,570.15	26,552.64
		Total Debits	Total Credits	Account Balance
		2,609.90	4,170.84	26,552.64

Each row in the balance column is calculated by subtracting the debits in the current and previous rows of the debit column from the credits in the current and previous rows of the credit column, then adding that result to the previous statement balance. Assuming that the cells on this form are named 'Previous Balance,' 'Credits,' and 'Debits,' respectively, the formula shown below would correctly calculate the balance column.

```
RunningTotal (Previous Balance, Credits-Debits)
```

The columnCell parameter (Credits-Debits) calculates the net change in balance for each row in the table. The running total is calculated based on this columnar result plus the Previous Balance amount.

---

## SECONDOF (time)

The 'SecondOf' function returns the number of the second represented in *time*. The number returned is an integer in the range 0 to 59.

### Examples

```
SecondOf (ToTime ("6:45:01 AM")) → 01  
SecondOf (ToTime ("23:00:00")) → 0  
SecondOf (ToTime ("00:59:59")) → 59  
SecondOf (ToTime ("10:01 PM")) → 0
```

### Related Functions

'HourOf' returns the number of the hour in a time value. 'MinuteOf' returns the number of the minute in a time value.

---

## SENDDATE

The 'SendDate' function returns the date the current record was last sent (mailed).

---

## SENDTIME

The 'SendTime' function returns the time the current record was last sent (mailed).

---

## SIGN (number)

The 'Sign' function returns 1 if *number* is positive, 0 if *number* is zero, or -1 if *number* is negative.

### Examples

```
Sign (2+3) → 1  
Sign (5-10) → -1  
Sign (-3+3) → 0  
Sign (-5^2) → 1
```

### Related Functions

None.

---

## SIN (number)

The ‘Sin’ function returns the sine of *number*. *Number* must be an angle in radians. The return value is in the range -1 to 1.

### Examples

```
Sin (0) → 0
Sin (- π/2) → -1
Sin (Pi) → 0
Sin (9* π/2) → 1
```

### Related Functions

‘ASin’ is the inverse of the ‘Sin’ function. ‘Cos’ and ‘Tan’ return the cosine and tangent, respectively, of a number.

---

## SLD (cost, salvage, life) SLDVALUE (cost, salvage, life, term)

These functions calculate an asset’s depreciation amount and value. They use the straight line depreciation method. This method spreads depreciation evenly over the useful life of the asset. *Cost* is the initial cost of the asset, *salvage* is the estimated value of the asset at the end of its useful life, and *life* is the number of time periods in the depreciation lifespan of the asset. The rate of depreciation is 1 over *life*.

The ‘SLD’ function returns the depreciation amount for a single period. It’s calculated as the wearing value of the asset (*cost* minus *salvage*) divided by the useful life of the asset (*life*).

The ‘SLDValue’ function returns the depreciated value of the asset at the end of the period identified by *term*. *Term* should be between 0 and *life*. If *term* is equal to 0, ‘SLDValue’ returns *cost*. If *term* is equal to *life*, ‘SLDValue’ returns *salvage*.

### Examples

If a machine costs \$5,000 and has an estimated value of \$500 at the end of 5 years, then

```
SLD (5000, 500, 5) → 900
```

and the depreciated value of the machine after 3 years is:

```
SLDValue (5000, 500, 5, 3) → 2300
```

### Related Functions

‘SOYD’ and ‘SOYDValue’ return an asset’s depreciation amount and depreciated value using the sum-of-years method. ‘DBal’ and ‘DBalValue’ return an asset’s depreciation amount and depreciated value using the declining balance method. ‘MDBal’ and ‘MDBalValue’ return an asset’s depreciation amount and depreciated value using the modified declining balance method.

**SOYD (cost, salvage, life, term)****SOYDVALUE (cost, salvage, life, term)**

These functions calculate an asset's depreciation amount and value. They use the sum-of-years depreciation method. This method creates larger depreciation amounts during the early years of the useful life of the asset. *Cost* is the initial cost of the asset, *salvage* is the estimated value of the asset at the end of its useful life, and *life* is the number of time periods in the depreciation lifespan of the asset. *Term* identifies a particular depreciation period.

The 'SOYD' function returns the depreciation amount for the period identified by *term*. It's calculated as a fraction of the wearing value of the asset (*cost* minus *salvage*). The denominator of the fraction is obtained by numbering the periods in *life* and adding. For example, if *life* is 5, the denominator is  $1 + 2 + 3 + 4 + 5 = 15$ . The numerator for the first period is *life*. For each subsequent period the numerator is reduced by 1. If *life* is 5, the fractions for the 5 periods are: 5/15, 4/15, 3/15, 2/15, and 1/15.

The 'SOYDValue' function returns the depreciated value of the asset at the end of the period identified by *term*. *Term* should be between 0 and *life*. If *term* is equal to 0, 'SOYDValue' returns *cost*. If *term* is equal to *life*, 'SOYDValue' returns *salvage*.

**Examples**

If a machine costs \$5,000 and has an estimated value of \$500 at the end of 5 years, then the wearing value of the machine is \$4,500 and the sum-of-years fraction for the fourth period is 2/15. The following formulas calculate the depreciation amount and the depreciated value for the first, second and fourth terms.

```
SOYD (5000, 500, 5, 1) → 1500
SOYD (5000, 500, 5, 2) → 1200
SOYD (5000, 500, 5, 4) → 600
SOYDValue (5000, 500, 5, 1) → 3500
SOYDValue (5000, 500, 5, 2) → 2300
SOYDValue (5000, 500, 5, 4) → 800
```

**Related Functions**

'SLD' and 'SLDValue' return an asset's depreciation amount and depreciated value using the straight line method. 'DBal' and 'DBalValue' return an asset's depreciation amount and depreciated value using the declining balance method. 'MDBal' and 'MDBalValue' return an asset's depreciation amount and depreciated value using the modified declining balance method.

**SPELLCURRENCY (number, decimals)**

The 'SpellCurrency' function returns a text value describing *number*. *Number* is described in monetary terms, using "dollars" and "cents". The number of decimal places to use in the description is given in *decimals*.



**Examples**

```
SpellCurrency (100, 2) → "One Hundred Dollars and 00 Cents"
SpellCurrency (45.99, 2) → "Forty Five Dollars and 99 Cents"
SpellCurrency (-10.01, 2) → "Ten Dollars and 01 Cents"
SpellCurrency (1000, 4) → "One Thousand Dollars and 00.00 Cents"
SpellCurrency (0, 2) → "No Dollars and 00 Cents"
```

**Related Functions**

‘SpellNumber’ spells out a number. ‘SpellNumberTH’ spells out the ordinal value of a number.

**SPELLNUMBER (number)**

The ‘SpellNumber’ function returns a text value describing *number*.

**Examples**

```
SpellNumber (101) → "One Hundred One"
SpellNumber (-54) → "Fifty Four"
SpellNumber (34.987) → "Thirty Four"
SpellNumber (-4.00) → "Four"
SpellNumber (0) → "Zero"
```

**Related Functions**

‘SpellCurrency’ spells out a number using “dollars” and “cents”. ‘SpellNumberTH’ spells out the ordinal value of a number.

**SPELLNUMBERTH (number)**

The ‘SpellNumberTH’ function returns a text value describing the ordinal value of *number*. *Number* should be an integer value. If *number* is not an integer, the fractional part is ignored.

**Examples**

```
SpellNumberTH (100) → "One Hundredth"
SpellNumberTH (1) → "First"
SpellNumberTH (-32.123) → "Thirty Second"
SpellNumberTH (0) → "Zeroth"
SpellNumberTH (1000000) → "One Millionth"
```

**Related Functions**

‘SpellCurrency’ spells out a number using “dollars” and “cents”. ‘SpellNumber’ spells out a number.

---

## SQRT (number)

The ‘SQRT’ function returns the positive square root of *number*. *Number* must be greater than or equal to zero.

### Examples

```
Sqrt (16) → 4
Sqrt (0.1) → 0.316227766
Sqrt (2) → 1.414213562
Sqrt (0) → 0
```

### Related Functions

The exponentiation operator (^) raises a number to a given exponent.

---

## STDEV (number1, number2, ...)

The ‘STDEV’ function returns the sample standard deviation, *s*, of *number1*, *number2*, and so on. ‘STDEV’ calculates the standard deviation for data sets constituting samples taken from populations of interest. The value returned is calculated as follows:

$$s = \sqrt{s^2}$$

It’s the positive square root of the sample variance. See ‘Var.’

Any number of parameters can be specified.

### Examples

```
StDev (3.74, 3.89, 4.00, 3.68, 3.69) → 0.1398
StDev (9.3455) → 0
StDev (40, 30, 50, 15, 5) → 18.235
```

If *ages* is a column cell with rows that contain the values 28, 31, 27, 29, 45, and 24, then

```
StDev (ages) → 7.393691004
```

### Related Functions

‘Var’ calculates the sample variance of a group of numbers. ‘PVar’ and ‘PSTDev’ calculate the population variance and population standard deviation, respectively, of a group of numbers.

‘Range’ calculates the range of a group of numbers.

---

## SUFFIXCOUNT (name)

The ‘SuffixCount’ function returns the number of suffixes in *name*. If *name* contains no suffixes, ‘SuffixCount’ returns 0.

**Examples**

```
SuffixCount (ToName ("Antonio Salieri, B.Mus. ")) → 1
SuffixCount (ToName ("Marsha")) → 0
SuffixCount (ToName ("A. E. Forrester, O.C., Q.C, M.Ed., B.Ed. "))
→ 4
```

**Related Functions**

'MiddleCount' returns the number of middle names in a name. 'PrefixCount' returns the number of prefixes in a name. 'SuffixOf' returns a given suffix of a name.

---

**SUFFIXOF (name, number)**

The 'SuffixOf' function returns a text value containing a suffix of *name*. The value of *number* determines which suffix is returned. If *number* is 1, the first suffix is returned; if *number* is 2, the second suffix is returned, and so on. If *number* is 0, then all suffixes, separated by spaces, are returned. If *name* doesn't contain the suffix identified by *number*, 'SuffixOf' returns the empty value.

**Examples**

```
SuffixOf (ToName ("Ms. Susan Patricia Anderson, O.C., Ph.D. "), 1)
→ "O.C."
SuffixOf (ToName ("Hon. Mr. Justice S. Edge, L.L.B., B.Sc. "), 2)
→ "B.Sc."
```

If *herName* contains the name value Ms. Sandra Susan Joan Wright, R.N., then

```
SuffixOf (herName, SuffixCount (herName)) → "R.N."
```

**Related Functions**

'LastOf' returns the last name of a name. 'FirstOf' returns the first name of a name. 'MiddleOf' returns a given middle name of a name. 'PrefixOf' returns a given prefix of a name.

---

**SUM (number1, number2, ...)**

The 'Sum' function returns the sum of the numbers in its list of parameters. Any number of parameters can be specified.

**Examples**

```
Sum (45) → 45
Sum (-3, 0, 4.5) → 1.5
Sum (-2, -5, -9) → -16
```

If *receipts* is a column cell with 5 rows that contain the values 998.00, 750.00, 515.50, 222.95, and 800.05, then

```
Sum (receipts) → 3286.50
```

**Related Functions**

‘SumSQ’ returns the sum of squares of a group of numbers.

---

**SUMSQ (number1, number2, ...)**

The ‘SumSQ’ function returns the sum of the squares of the numbers in its list of parameters. Any number of parameters can be specified.

**Examples**

SumSq (45) → 2025

SumSq (-3, 0, 4.5) → 29.25

SumSq (-2, -5, -9) → 110

If *rangeData* is a column cell with 5 rows containing the values 2.345, 1.239, 0.045, -1.852, and -0.099, then

SumSq (rangeData) → 10.4759

**Related Functions**

‘Sum’ returns the sum of a group of numbers.

---

**TAN (number)**

The ‘Tan’ function returns the tangent of *number*. *Number* must be an angle in radians.

**Examples**

Tan (0) → 0

Tan (Pi) → 0

Tan (3\* π/4) → -1

**Related Functions**

‘ATan’ is the inverse of the ‘Tan’ function. ‘Sin’ and ‘Cos’ return the sine and cosine, respectively, of a number.

---

**TEMPLATEID**

The ‘TemplateID’ function returns the unique template ID from the Template Information dialog box for the current template.

---

**TEMPLATENAME**

The ‘TemplateName’ function returns the template name from the Template Information dialog box for the current template.

---

## TEMPLATEREVISION

The 'TemplateRevision' function returns the revision number from the Template Information dialog box for the current template.

---

## TEMPLATESTATUS

The 'TemplateStatus' function returns the current status from the the Revision Options dialog box for the current template. The Informed Filler user can view this information by using the Revision Status command.

---

## TERM (pv, fv, pmt, rate[, BEGIN or END])

The 'Term' function returns the term of an investment given the present value, future value, payment amount, and interest rate. See 'PV.'

---

## TIMEFORM (time, format)

The 'TimeForm' function formats the time value *time* using the format specified in the text value *format*. The resulting text value is returned. For an explanation of time formats, see *Time*.

### Examples

```
TimeForm (ToTime ("4:6:59"), "0H:0M:0S") → "04:06:59"
TimeForm (ToTime ("23:34:56"), "H:MM AM") → "11:34 PM"
TimeForm (ToTime ("2:07 PM"), "H24:MM") → "14:07"
TimeForm (ToTime ("5:06:09"), "H:M:S") → "5:6:9"
```

### Related Functions

'CharForm' formats a text value, 'DateForm' formats a date, 'NameForm' formats a name, and 'NumForm' formats a number.

---

## TIMESPAN (date1, time1, date2, time2)

The 'TimeSpan' function returns the number of seconds between two date/time values. The example below shows the number of seconds between the current time today, and the same time one day later.

### Examples

```
TimeSpan (Today, Now, AddDays(Today, 1), Now) → 86400
```

---

## TOBOOLEAN (value)

The ‘ToBoolean’ function converts a text or numeric value to a boolean value. The following table shows which text values are converted to which boolean values. Upper and lower case is ignored when a text value is compared with those in the table.

Converting Text Values to Boolean Values

Text Value	Converts To
“True”	True
“T”	True
“False”	False
“F”	False
“Yes”	True
“Y”	True
“No”	False
“N”	False
“On”	True
“Off”	False

---

When you try to convert a text value not listed above to a boolean value, Informed will return the empty value.

When you convert a numeric value to a boolean value, the resulting boolean value will be True if the numeric value is non-zero; False otherwise. For more information about type conversion, see *Type conversion*. For a description of the boolean cell type, see *Boolean*.

### Examples

```
ToBoolean ("on") → True
ToBoolean (75.35) → True
ToBoolean ("F") → False
ToBoolean ("75.35") → Empty value
```

### Related Functions

The ‘ToText,’ ‘ToNumber,’ ‘ToDate,’ ‘ToTime,’ ‘ToName,’ ‘ToPicture,’ and ‘ToSignature’ functions convert values to text, number, date, time, name, picture, and signature values respectively.

---

## TODATE (value)

The ‘ToDate’ function converts a text value to a date value. The *value* parameter can be any text value that represents a valid date. The format of *value* can be any date format that Informed recognizes. If *value* doesn’t represent a valid date, ‘ToDate’ returns the empty value. For more information about type conversion, see “Type Conversion.” For a description of the date cell type, see “Date.”

**Examples**

ToDate ("1/1/90") → January 1, 1990  
 DayOf (ToDate ("May 23 80")) → 23  
 ToDate ("abcdefg") → Empty value

If the current year is 1991, then

ToDate ("Oct 3") → October 3, 1991

**Related Functions**

The 'ToText,' 'ToNumber,' 'ToBoolean,' 'ToTime,' 'ToName,' 'ToPicture,' and 'ToSignature' functions convert values to text, number, boolean, time, name, picture, and signature values respectively.

**TODAY**

The 'Today' function returns the current date. The current date is returned whenever the cell calculation, default formula, or check formula containing the 'Today' function is calculated.

**Examples**

If the current date is September 15, 1991, then

Today → September 15, 1991  
 MonthOf (Today) → 9  
 DayOf (Today) → 15

**Related Functions**

The 'Now' function returns the current time.

**TOKENIZE (text, delimiter)**

This function searches for the *delimiter* in the *text*, and breaks the input up into individual phrases. The phrases are collected and returned as a column value.

**Examples**

Tokenize ("123-456-789", "-") → {"123", "456", "789"}  
 Tokenize ("These are words", " ") → {"These", "are", "words"}

**Related Functions**

None.

---

## TONAME (value)

The ‘ToName’ function converts a text value to a name value. The *value* parameter can be any text value that represents a valid name. The format of *value* can be any name format that Informed recognizes. If the ‘ToName’ function is used to calculate the value of a name cell, the format of the displayed name depends on the format of the name cell. If *value* doesn’t represent a valid name, ‘ToName’ returns the empty value.

For more information about type conversion, see “Type Conversion.” For a description of the name cell type and name formatting options, see “Name.”

### Examples

```
ToName ("Mr. John Harold Smith") → Mister John Harold Smith
ToName ("Smith, John Harold") → John Harold Smith
FirstOf (ToName ("Dr. Susan Applehoff")) → "Susan"
```

### Related Functions

The ‘ToText,’ ‘ToNumber,’ ‘ToBoolean,’ ‘ToTime,’ ‘ToDate,’ ‘ToPicture,’ and ‘ToSignature’ functions convert values to text, number, boolean, time, date, picture, and signature values respectively.

---

## TONUMBER (value)

The ‘ToNumber’ function converts a text or boolean value to a numeric value. The *value* parameter can be any text value that represents a valid number, or any boolean value. ‘ToNumber’ converts the boolean values True and False to the numeric values 1 and 0 respectively.

If the ‘ToNumber’ function is used to calculate the value of a number cell, the format of the displayed number depends on the format of the number cell. If *value* is a text value that doesn’t represent a valid number, ‘ToNumber’ returns the empty value.

For more information about type conversion, see *Type conversion*. For a description of the number cell type and number formatting options, see *Number*.

### Examples

```
ToNumber ("514.234") → 514.234
SpellNumber (ToNumber ("101")) → "One Hundred One"
```

### Related Functions

The ‘ToText,’ ‘ToDate,’ ‘ToBoolean,’ ‘ToTime,’ ‘ToName,’ ‘ToPicture,’ and ‘ToSignature’ functions convert values to text, date, boolean, time, name, picture, and signature values respectively.



---

### TOPICTURE (value)

The 'ToPicture' function converts the text in *value* to a picture value. The text value must be formatted according to Informed's textual representation for pictures.

---

### TOSIGNATURE (value)

The 'ToSignature' function converts the text in *value* to a digital signature value. The text value must be formatted according to Informed's textual representation for signatures.

---

### TOTEXT (value)

The 'ToText' function converts a number, name, date, time, boolean, picture, or signature value to a text value. The *value* parameter can be any value of any type. The following table summarizes how 'ToText' converts a value of each type to a text value.

Converting Values to Text

Original Type	Convert to Text
number	using the General number format
name	using all name parts in full, in order with surname first
date	using the date format "M/D/YY"
time	using the time format "H:MM:SS AM"
boolean	using "True" for True and "False" for False
picture	using an Informed-specific text format
signature	using an Informed-specific text format

For information about each cell type and the available formatting options, see *Cell types*. For more information about type conversion, see *Type conversion*.

#### Examples

```
ToText (ToDate ("December 18, 1990")) → "12/18/90"
```

```
ToText (ToName ("Smith, John Harold")) → "John Harold Smith"
```

If *Insured* is a boolean cell containing the value False, then

```
ToText (Insured) → "False"
```

#### Related Functions

The 'ToDate,' 'ToNumber,' 'ToBoolean,' 'ToTime,' 'ToName,' 'ToPicture,' and 'ToSignature' functions convert values to date, number, boolean, time, name, picture, and signature values respectively.

---

## TOTIME (value)

The ‘ToTime’ function converts a text value to a time value. The *value* parameter can be any text value that represents a valid time. The format of *value* can be any time format that Informed recognizes. If *value* doesn’t represent a valid time, ‘ToTime’ returns the empty value. For more information about type conversion, see *Type conversion*. For a description of the time cell type, see *Time*.

### Examples

```
ToTime ("7 15") → 7:15:00
ToTime ("2:55:12 PM") → 14:55:12
ToTime ("abcdefg") → Empty value
```

### Related Functions

The ‘ToText,’ ‘ToNumber,’ ‘ToBoolean,’ ‘ToDate,’ ‘ToName,’ ‘ToPicture,’ and ‘ToSignature’ functions convert values to text, number, boolean, date, name, picture, and signature values respectively.

---

## TRANSLITERATE (text, source, destination)

The ‘TransLiterate’ function translates the input text in *text* on a character by character basis. Normally the source and the destination will be of identical length, and will form a mapping from the input text. If a character in *source* exists in *text*, the character in *text* will be replaced with the character in *destination* at the same character position as the character in *source*. If the *source* is longer than *destination*, then any extra characters in *source* are mapped to nothing (that is, if they exist in *text*, they are deleted).

### Examples

```
Transliterate ("encode me", "abcdefg", "ABCDEFGF") → "EnCoDE mE"
```

### Related Functions

The ‘Replace’ function.

---

## TRIM (text)

The ‘Trim’ function removes all leading and trailing blanks from *text* and returns the resulting text value. If all the characters in *text* are blanks, ‘Trim’ returns a text value with no characters (“”).

### Examples

```
Trim (" Greetings from us. ") → "Greetings from us."
Trim (" Time:") → "Time:"
Trim ("Invoiced amount: ") → "Invoiced amount:"
Trim ("") → ""
Trim (" ") → ""
```

**Related Functions**

'Delete' deletes a group of characters from a text value. 'Replace' replaces a group of characters in a text value with another group of characters.

---

**TRUNC (number, decimals)**

'Trunc' truncates a number by deleting any significant digits after a specified number of decimal places.

**Examples**

```
Trunc (Cell11,2) → 123.4567, 123.45
```

**Related Functions**

None.

---

**UPPER (text)**

The 'Upper' function converts all letters in *text* to upper case. The resulting text value is returned.

**Examples**

```
Upper ("Serial #: 146a889x") → "SERIAL #: 146A889X"  
Upper ("") → ""  
Upper (True) → "TRUE"  
Upper ("tHiS iS hArD tO rEaD!!") → "THIS IS HARD TO READ!!"
```

**Related Functions**

'Lower' converts a text value to lower case. 'UpperFirst' converts the first character of the first word of a text value to upper case. 'UpperWords' converts the first character of all words in a text value to upper case.

---

**UPPERFIRST (text)**

The 'UpperFirst' function converts the first character of each sentence in *text* to upper case, where a sentence is a number of words terminated by a period (.). The resulting text value is returned.

**Examples**

```
UpperFirst ("goods RECEIVED") → "Goods RECEIVED"  
UpperFirst ("2. Inventory") → "2. Inventory"  
UpperFirst ("x") → "X"
```

**Related Functions**

‘Upper’ converts a text value to upper case. ‘Lower’ converts a text value to lower case. ‘UpperWords’ converts the first character of all words in a text value to upper case.

---

**UPPERWORDS (text)**

The ‘UpperWords’ function converts the first character of each word in *text* to upper case. A word is any sequence of characters that starts *text* or that follows a space. The resulting text value is returned.

**Examples**

```
UpperWords ("goods RECEIVED") → "Goods RECEIVED"
UpperWords ("2. inventory") → "2. Inventory"
UpperWords ("x") → "X"
UpperWords ("John's car is a 240se") → "John's Car Is A 240se"
```

**Related Functions**

‘Upper’ converts a text value to upper case. ‘Lower’ converts a text value to lower case. ‘Upper-First’ converts the first character in a text value to upper case.

---

**USERNAME**

The ‘UserName’ function returns a text value containing the user name as set by the Chooser desk accessory. For information about the Chooser desk accessory and its use, see your *Macintosh Owner's Guide*.

**Examples**

If the current name in the Chooser desk accessory is “John Smith”, then

```
UserName → "John Smith"
```

**Related Functions**

None.

---

**VALIDCHOICE (value, cell)**

The ‘ValidChoice’ function returns a boolean result. If *value* matches a choice in the list of choices for the cell called *cell*, then ‘ValidChoice’ returns True; False otherwise. Informed ignores upper and lower case when it compares *value* with each choice in the choices list.

You enter a cell’s list of choices using the Choices command in the Settings menu. For a complete explanation of choices and the Choices command, see *Choices*.

**Examples**

If *Terms* is a cell that has the choices, “Cash”, “On account”, “Net 30 days”, and “Net 60 days”, then

```
ValidChoice ("cash", Terms) → True
ValidChoice ("free", Terms) → False
```

The most common use of ‘ValidChoice’ is to check if the value entered in a cell is in that cell’s choice list. To do this, pass the name of the cell in both *value* and *cell*.

```
ValidChoice (Terms, Terms)
```

The value in *Terms* is compared with the choices for the same cell. If a match is found, ‘ValidChoice’ returns True; False otherwise.

**Related Functions**

‘Choices’ returns a columnar text value containing the choices for a particular cell, one in each row.

**VAR (number1, number2, ...)**

The ‘Var’ function returns the sample variance,  $s^2$ , of *number1*, *number2*, and so on. VAR calculates the variance for data sets constituting samples from populations of interest. The population variance of  $N$  samples is calculated as follows:

$$s^2 = \frac{(number\ 1 - \bar{X})^2 + (number\ 2 - \bar{X})^2 + \dots + (number\ N - \bar{X})^2}{N - 1}$$

where  $\bar{X}$  is the arithmetic mean of the numbers:

$$\bar{X} = \frac{number\ 1 + number\ 2 + \dots + number\ N}{N}$$

Any number of parameters can be specified.

**Examples**

```
Var (3.74, 3.89, 4.00, 3.68, 3.69) → 0.01955
Var (9.3455) → 0
Var (40, 30, 50, 15, 5) → 332.5
```

If *ages* is a column cell with rows that contain the values 28, 31, 27, 29, 45, and 24, then

```
Var (ages) → 54.666666667
```

**Related Functions**

‘STDev’ calculates the sample standard deviation of a group of numbers. ‘PVar’ and ‘PSTDev’ calculate the population variance and population standard deviation, respectively, of a group of numbers. ‘Range’ calculates the range of a group of numbers.

---

**WEEKOFYEAR (date)**

The ‘WeekOfYear’ function returns an integer describing the week of the year represented in *date*. The integer returned is in the range 1 to 53.

**Examples**

```
WeekOfYear (ToDate ("01/01/90")) → 1
WeekOfYear (ToDate ("Dec 31, 1984")) → 53
WeekOfYear (ToDate ("Dec 31, 1985")) → 53
WeekOfYear (MakeDate (8, 6, 1989)) → 23
```

If the current year is 1989, then

```
WeekOfYear (ToDate ("Apr 25")) → 17
```

**Related Functions**

‘DayOfYear’ returns the day of year in a date. ‘DayOfWeek’ returns the weekday of a date. ‘LastDayOfMonth’ returns a date value which is the last day of the month represented in a date.

---

**WHICHMEMBER (target, value1, value2, ...)**

The ‘WhichMember’ function tries to match “target” against each of the other parameters. If a successful match is found, ‘WhichMember’ returns the index of the first matched value. If no match is found, ‘WhichMember’ returns null.

**Examples**

```
WhichMember (True, 1=4, 5>8, 9<3) → NULL
WhichMember ("5", 1*5, "6", "8") → 1
WhichMember (Pi, 3.1543, π, 4.3522, 3.1234) → 2
```

**Related Functions**

‘Member’ returns True if target equals any of value1, value2, and so on.

---

**WITHIN (value, startValue, endValue)**

The ‘Within’ function is a boolean function. It returns the value True if *value* is between *startValue* and *endValue*, inclusive. All three parameters to the ‘Within’ function must be the same type. They can be numbers, dates, times, text, or boolean values. If *value* is less than *startValue*, or if *value* is

greater than *endValue*, then ‘Within’ returns the value False; otherwise, ‘Within’ returns the value True.

Informed uses the standard comparison operators to compare *value* with *startValue* and *endValue*. See *Comparison operators* for information about how each type of value is compared.

### Examples

```
Within (-1, 99, 105) → False
Within (ToDate ("Oct. 3, 1989"), ToDate ("01/01/90"),
ToDate ("Oct. 5, 1990")) → False
Within (ToTime ("1:30 PM"), ToTime ("13:30:00"),
ToTime ("2:30 PM")) → True
Within ("actuary", "Sensible", "Tourist") → False
Within ("actuary", "actually", "sensible") → True
```

### Related Functions

‘Between’ returns True if a value is between two other values, non-inclusive.

---

## WORKDAYS (date1, date2, mask)

The ‘WorkDays’ function returns the number of working days between two dates *date1* and *date2*. *Mask* indicates which of the days of the week are working days, and which are days off. *Mask* must consist of the characters “SMTWTFS”, in that order, in either upper or lower case. Starting at Sunday, and going through Saturday, each character in *mask* indicates a work day with upper case and a day off with lower case. Thus, a normal Monday through Friday work week would be given as “sMTWTFs.”

### Examples

```
WorkDays (ToDate("May 6, 1997"), ToDate("Jun 3, 1997"), "sMTWTFs") → 21
```

---

## YEAROF (date)

The ‘YearOf’ function returns the year represented in *date*.

### Examples

```
YearOf (ToDate ("Saturday, April 25, 1964")) → 1964
YearOf (ToDate ("09/12/89")) → 1989
YearOf (ToDate ("12/26/01")) → 1901
YearOf (ToDate ("26/01/1654")) → 1654
```

If *startDate* contains the date value Aug 20, 1999, then

```
YearOf (startDate) → 1999
```

**Related Functions**

'DayOf' returns the number of the day in a date. 'MonthOf' returns the number of the month in a date. 'WeekOfYear' returns the number of the week of the year in a date. 'LastDayofMonth' returns a date value which is the last day of the month represented in a date.