12 Using AppleScript

In this chapter:

12 Using AppleScript

Scripting languages allow you to control Windows and Mac OS applications with program-like scripts.You can write scripts to perform tasks as simple as opening and printing a form, or as complex as controlling a sophisticated workflow process.

This chapter provides an overview of Informed's scripting capabilities. 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.

This chapter includes a thorough reference to Informed's implementation of the AppleScript scripting language. The reference material includes a description of each Informed AppleScript object as well as examples and descriptions of the terminology to use when writing scripts for Informed Filler. For information about Informed's JavaScript implementation, please refer to Chapter 12 of the *Informed Designer Forms Automation* manual.

Overview

A single script can automate a task that normally requires several steps. For example, you might write a script that would find and print all 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. For the Informed Filler user, performing such tasks becomes as simple as selecting a script

Informed's scripting features rely on Informed scripting plug-ins. By using plug-ins, Shana Corporation can easily support new scripting languages simply by implementing new scripting plug-ins. At the time this documentation was prepared, Informed Designer included plug-ins for the following scripting languages:

- AppleScript
- JavaScript

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.

Informed Designer can store scripts in form documents. 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:

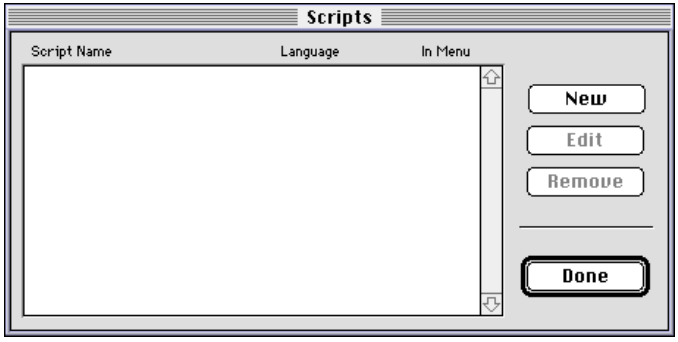
- selecting a menu item
- clicking a button
- typing a value in a lookup cell
- submitting a form

Before configuring an action, however, you must first enter the script using Informed Designer's Scripts command. See the following section for instructions on how to enter and edit scripts.

Entering and Editing Scripts

You use Informed Designer's Scripts command to add, remove, and edit scripts.

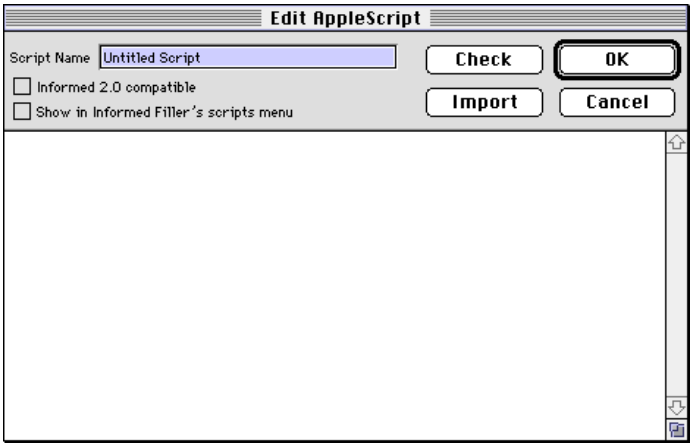
- Choose **Scripts** from the Configure submenu under the Form menu. The Scripts dialog appears.



If any scripts are currently attached to the form, their names will appear in the scrolling list in the order that they were created.

To add a new script to your form:

- Click ‘New.’ The Edit Script dialog box appears.

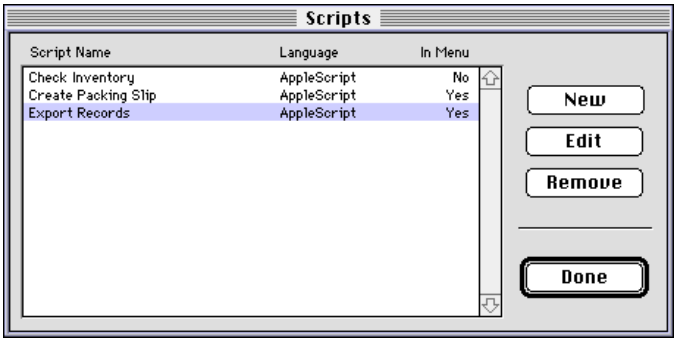


- Type the name of the new script in the ‘Script Name’ text box. This is the name that you’ll see when you configure an action to invoke a script.
- Enter a script by typing in the text box, or click the ‘Import’ button to import a script from another file.

In version 2.5 (or later) of Informed, a script that’s attached to a form does not automatically appear in Informed Filler’s Scripts menu. If you want a script to appear in Informed Filler’s Scripts menu:

- Click the ‘Show in Informed Filler’s Scripts menu’ checkbox.

- Check for errors by clicking the ‘Check’ button. If an error is detected, you’ll see a message describing the error. If there are no errors, the script will display properly formatted.
- Click ‘OK’ on the Edit Script dialog box. Informed Designer will store the script with the form and display it in the scrolling list on the Scripts dialog.



To edit an existing script:

- Select the script name in the Scripts dialog scrolling list, then click ‘Edit.’
- Make the appropriate changes on the Edit Scripts dialog.

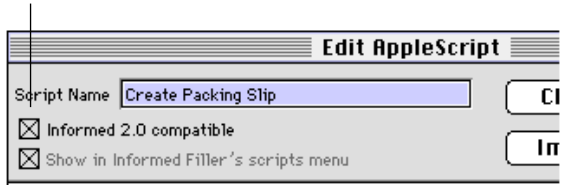
To remove a script:

- Select the script name in the Scripts dialog scrolling list, then click ‘Remove.’

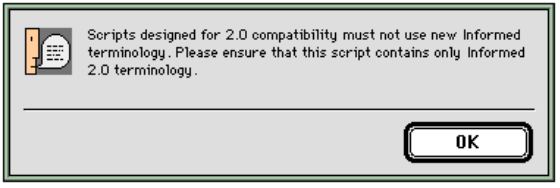
Pre-2.5 Script Compatibility

Version 2.5 of Informed includes changes to the terminology in Informed’s AppleScript implementation. Writing an AppleScript script in version 2.5 requires special consideration to make sure that the new script will function properly with earlier versions of Informed Filler.

To make a version 2.5 AppleScript script compatible with version 2.0 of Informed, click the ‘Informed 2.0 compatible’ checkbox on the Edit AppleScript dialog.



When you click the ‘Informed 2.0 compatible’ checkbox, Informed displays a message, warning you that your script must not contain any of the new 2.5 scripting terminology.



Throughout this document, all new or modified scripting terminology is highlighted by the presence of an arrow icon in the left margin.

Informed AppleScript Implementation

This section provides an overview of Informed's AppleScript implementation. It is assumed that you already understand the basics of AppleScript and are familiar with the Informed Designer and Informed Filler applications.

Reference Objects

Using AppleScript to communicate with Informed Filler can be thought of as a conversation with various objects in the Informed Filler application. To initiate this 'conversation' you need to create reference objects. A reference object is an AppleScript object that refers to one or more elements of the Informed user interface. For example, a record object refers to one or more records in Informed Filler and a cell object refers to one or more cells.

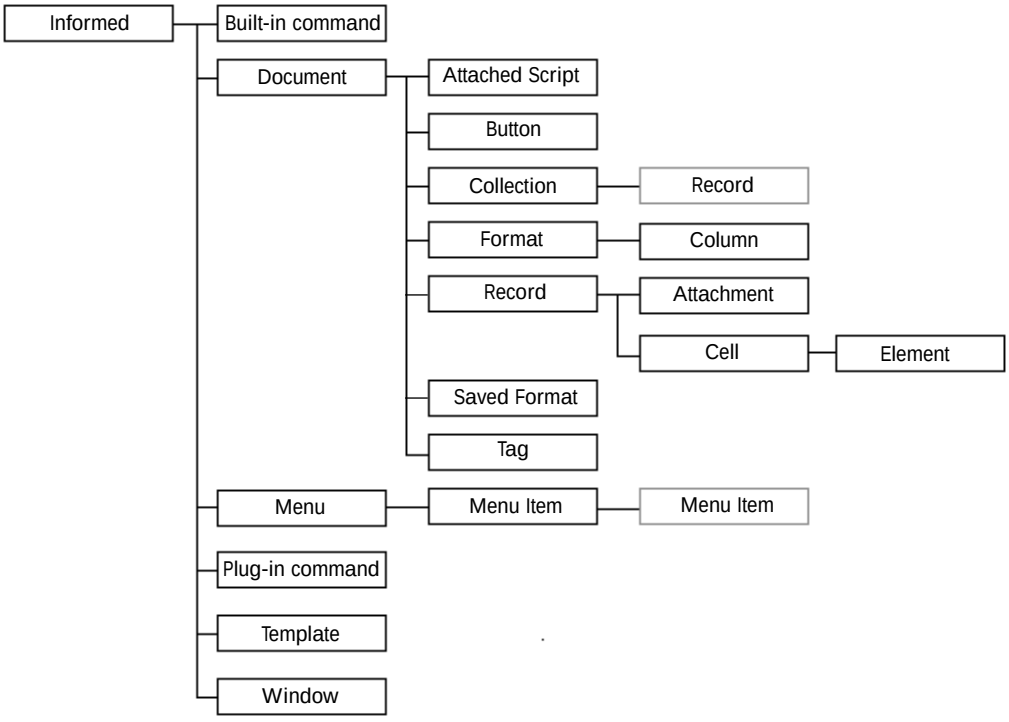
Reference objects can refer to their corresponding Informed elements in a number of different ways. For example, a record object can refer to one or more records in Informed by index, id, absolute position, relative position, range, or test.

```
tell application "Informed Filler"
    -- close the frontmost document
    close document 1
end tell
```

The example above refers to an application by "name" and a document by "index."

Containment

Reference objects are organized hierarchically so that some objects are elements of others. For example, an Attachment object is an element of a Record object, and a Record object is an element of a Document object. This organization is referred to as *containment*. One object is *contained* by another or one object is the *container* of another. The object containment hierarchy in Informed is shown in the following diagram.



To refer to a specific object you specify each object in the containment hierarchy. The following example refers to the cell named "Cost" in the third record of the frontmost document of Informed.

```
cell "Cost" of record 3 of document 1 of application "Informed Filler"
```

Implied Containment

One way to simplify your scripts is to use “implied containment.” This means that certain objects can be left out of the object specification and appropriate defaults will be provided. The following objects are optional in a containment specification:

- Document
- Record

If you use implied containment in your scripts, you must be aware of the following rules:

- If the document is not specified, then the default document is the current document:

```
--refers to the fourth cell of the third record of the current document
cell 4 of record 3
```

- If no record is specified, then all records of the specified container are assumed:

```
--refers to the fourth cell of every record of the second document
cell 4 of document 2
```

- If no container is specified (or if the specified container is Informed) then the current record of the current document is assumed:

```
--refers to the fourth cell of the current record of the current document
cell 4
```

The Tell Statement

The “tell” statement is a useful shorthand that makes your scripts much easier to follow. Consider the following two scripts which perform identical functions:

```
set theName to cell "Name" of application "Informed Filler"
set theCompany to cell "Company" of application "Informed Filler"

tell application "Informed Filler"
    set theName to cell "Name"
    set theCompany to cell "Company"
end tell
```

The “tell” statement in the second script makes the script easier to understand and follow, because it lets you know right away that all the statements inside the tell block are directed at the application “Informed Filler.” You can read the inner statements in that context, which is easier than reading the longer commands in the first example.

When the objects you are referring to are nested much more deeply, the “tell” statements can really start to clarify your scripts. Consider the following example:

```
set theName to cell "Name" of record 1 thru 3 of the current collection of
    document "Personnel" of application "Informed Filler"
set theCompany to cell "Company" of record 1 thru 3 of the current collection
    of document "Personnel" of application "Informed Filler"

tell application "Informed Filler"
    tell record 1 thru 3 of the current collection of document "Personnel"
        set theName to cell "Name"
        set theCompany to cell "Company"
    end tell
end tell
```

Reference Object Descriptions

The following sections describe each of the reference objects supported by Informed. Each object description lists the reference methods, properties, and commands supported by the object.

application

The **application** object represents the Informed application.

Reference

The application object can be referenced by name.

Properties

The following table lists the properties of the Informed application object.

application properties

Property	Writeable?	Description
class	no	The application class
current document	yes	The current document displayed by the application.
frontmost	yes	Is this the frontmost application?
name	no	The name of the application.
platform	no	The operating system the application is running on. This property can be MacOS68K, MacOSPPC, Win16, or Win32.
registered company	no	The name of the company the application is registered to.
registered name	no	The name of the user the application is registered to.
serial number	no	The serial number of the application.
suppress UI	yes	Suppress the display of dialogs and error messages?
version	no	The version number of the application.

Commands

The following commands are defined for the application object:

count

The **count** command returns the number of built-in commands, documents, menus, plug-in commands, templates, or windows within the Informed application.

Arguments for the count command

Argument	Description
each	This argument specifies the class of the elements to be counted. Its value must be builtin command, document, menu, plugin command, template, or window

```
tell application "Informed Filler"
set documentCount to count each document
end tell
```

make

The **make** command creates a new document in Informed.

Arguments for the make command

Argument	Description
new	This argument specifies the class of the element. Its value must be document.
with data	This argument specifies the data from which the new element will be created. Its value must be a template object.

```
tell application "Informed Filler"
    make new document with data template id "Transact B308"
end tell
```

quit

The **quit** command quits Informed.

Arguments for the quit command

Argument	Description
[saving]	Specifies whether to save changes before quitting. If yes, changes will be saved. If no, changes will not be saved. If ask, Informed will display a dialog asking if the user wants to save the changes. The default value is ask.

```
tell application "Informed Filler"
    --Prompt to save changes in each modified document and then quit Informed.
    quit
end tell


tell application "Informed Filler"
    --Quit Informed without saving.
    quit saving no
end tell
```

attached script

A document can contain a number of attached scripts. An attached script can be linked to a button or a menu item and can be executed by a selecting the associated menu item or button. It can also be executed directly by another script. An **attached script** object represents one or more scripts that are attached to a document.

Reference

An attached script object can reference attached scripts by:

- name
- index
- id
- absolute position
- relative position
- range
- test

Properties

The following table lists the properties of an attached script object.

attached script properties

Property	Writeable?	Description
container	no	The container for the attached script. An attached script is always contained by a document object.
id	no	The unique id of the attached script.
index	no	The index of the attached script.
name	no	The name of the attached script.
class	no	The attached script class.

Commands

The following commands are defined for an attached script object:

execute

The **execute** command executes an attached script and returns its result.

```
tell application "Informed Filler"
    --execute a specific attached script
    execute attached script "Process Orders" of document 1
end tell
```

exists

The **exists** command verifies the existence of an attached script.

```
tell application "Informed Filler"
    --execute a particular attached script if it exists
    tell attached script "Process Orders" of document 1
        if exists then
            execute
        end if
    end tell
end tell
```

attachment

With paper forms, associated documents such as receipts or drawings are often attached to a form with a paper clip. Informed Filler provides this same capability by allowing you to attach electronic documents to electronic forms. An **attachment** object represents one or more files that are attached to a record.

Reference

An attachment object can reference attachments by:

- name
- index
- id
- absolute position
- relative position
- range
- test

Properties

The following table lists the properties of an attachment object.

attachment properties

Property	Writeable?	Description
container	no	The container for the attachment. An attachment is always contained by a record.
id	no	The unique id of the attachment.
index	no	The index of the attachment.
name	no	The name of the attachment.
class	no	The attachment class.

Commands

The following commands are defined for an attachment object:

exists

The **exists** command verifies the existence of an attachment.

```
tell application "Informed Filler"
    tell current record of document 1
        if exists attachment "Read Me" then
            save attachment "Read Me" in file "HD:Documents:Read Me"
        end if
    end tell
end tell
```

delete

The **delete** command deletes an attachment from a record.

```
tell application "Informed Filler"
    delete attachment "Read Me" of current record of document 1
end tell
```

save

The **save** command saves an attachment.

Arguments for the save command

Argument	Description
in	This argument must be a file object which specifies the file into which the attachment will be saved.

```
tell application "Informed Filler"
    tell current record of document 1
        --save a specific attachment if it exists
        if exists attachment "Read Me" then
            save attachment "Read Me" in file "HD:Documents:Read Me"
        end if
    end tell
end tell
```

builtin command

Built-in commands correspond to the commands that are built into Informed Filler. The “Send Mail” command is an example of a built-in command. A **builtin command** object represents one or more built-in commands in Informed Filler. For a list of Informed’s built-in commands, please see “Appendix B” in your *Informed Designer Forms Automation* manual.

Reference

A builtin command object can reference built-in commands by:

- name
- index
- id
- absolute position
- relative position
- range
- test

Properties

The following table lists the properties of a builtin command object.

builtin command properties

Property	Writeable?	Description
enabled	yes	Is the built-in command enabled?
id	no	The unique id of the built-in command.
index	no	The index of the built-in command.
name	no	The name of the built-in command.
class	no	The builtin command class.

Commands

The following commands are defined for the built-in command object:

execute

The **execute** command executes a built-in command.

```
tell application "Informed Filler"
    execute builtin command "Enlarged Size"
end tell
```

exists

The **exists** command verifies the existence of a built-in command.

```
tell application "Informed Filler"
    if exists builtin command "Go To Record" then
        execute builtin command "Go To Record"
    end if
end tell
```

button

A button on a form can be configured to invoke commands that are built into Informed Filler, commands that are available through Informed plug-ins, or scripts that are attached to the form. A **button** object represents one or more buttons on a form.

Reference

A button object can reference buttons by:

- name
- index
- absolute position
- relative position
- range
- test

Properties

The following table lists the properties of a button object.

button properties

Property	Writeable?	Description
container	no	The container for the button. A button is always contained by a document.
enabled	yes	Is the button enabled?
index	no	The index of the button.
name	no	The name of the button.
class	no	The button class.
title	no	The title of the button.

Commands

The following commands are defined for a button object:

execute

The **execute** command executes a button's configured action.

```
tell application "Informed Filler"
    execute button "Submit" of document "Purchases"
end tell
```

exists

The **exists** command verifies the existence of a button.

```
tell document 1 of application "Informed Filler"
    if exists button "Go To Record" then
        execute button "Go To Record"
    end if
end tell
```

cell

A **cell** object represents one or more cells 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).

Reference

A cell object can reference cells by:

- name
- index
- id
- absolute position
- relative position
- range
- test

Properties

The following table lists the properties of a cell object.

cell properties

Property	Writeable?	Description
container	no	The container for the cell. A cell is always contained by a record.
current element	yes	The current element.
display only	yes	Is the cell display only?
extra choices	yes	The list of extra choices for the cell.
id	no	The unique id of the cell.
index	no	The index of the cell, relative to other cells.
main choices	no	The list of main choices for the cell.
name	no	The name of the cell.
class	no	The Cell class.
signed	no	Is the cell signed?
table ID	no	The table id for a column cell.
title	no	The title of the cell.
value	yes	The value of the cell.

Commands

The following commands are defined for a cell object:

clear

The **clear** command clears the cell of any data.

```
tell document 1 of application "Informed Filler"
    --clear the cell named "Quantity" of the current record
    clear cell "Quantity" of current record
end tell


tell application "Informed Filler"
    tell current record of document 1
        --clear the second row of the table column cell named "Price"
        clear element 2 of cell "Price"
    end tell
end tell
```

commit

The **commit** command commits the cell data, triggering any formatting, check calculations, or lookups configured for the cell.

Arguments for the commit command

Argument	Description
[lookup]	Specifies whether or not a lookup is performed. If `true`, the lookup is performed unconditionally. If `false`, the lookup is ignored. If not specified, a lookup is performed only if the cell's value has changed. This argument is ignored if the specified cell is not a lookup cell.

```
tell document 1 of application "Informed Filler"
    --set the value of a cell, commit the data without the configured lookup
    setx value of cell "Employee Number" of current record to "1012" without lookup

    --do some other stuff

    --commit the cell and force the lookup
    set current cell to cell "Employee Number"
    commit cell "Employee Number" of current record with lookup
end tell
```

count

The **count** command returns the number of elements in a cell.

Arguments for the count command

Argument	Description
each	This argument specifies the class of the elements to be counted. Its value must be element.

```
tell application "Informed Filler"
    set rowCount to count elements in cell "Result" of current record of document 1
end tell
```

data size

The **data size** command returns the size of the cell data in bytes.

Arguments for the data size command

Argument	Description
as	This argument specifies the data type for which the size is calculated.

```
tell application "Informed Filler"
    tell current record of document 1
        set vSize to data size of cell "Age" as text
    end tell
end tell
```

exists

The **exists** command verifies the existence of a cell.

```
tell document 1 of application "Informed Filler"
    set vExists to exists cell "Quantity" of current record of document 1
end tell
```

get

The **get** command gets the value of a cell.

Arguments for the get command

Argument	Description
as	The data type to be returned.

```
tell application "Informed Filler"
    tell current record of document "Order Form"
        --get the value of the field cell named "Signed Date" as text
        set vSignedDate to get cell "Signed Date" as text
        --get the value of each element of the table cell named "Description"
        set vDescription to get cell "Description"
    end tell
end tell
```

```
tell application "Informed Filler"
    --get the value of the "Date" cell of every record of the current collection
    --of the document. See "Containment" earlier in this chapter for details.
    set vDate to get cell "Date" of current collection of document "Order Form"
end tell
```

```
tell application "Informed Filler"
    --get the value of the "Date" cell of every record of the document.
    --See "Containment" earlier in this chapter for details.
    set vDate to get cell "Date" of document "Order Form"
end tell
```

set

The **set** command sets the value of a cell.

Arguments for the set command

Argument	Description
to	The value to which the cell will be set.

```
tell application "Informed Filler"
    tell current record of current document
        --set the value of the field cell "Quantity"
        set value of cell "Quantity" to "100"
        --set the value of each element of the table cell "Part Number"
        set value of cell "Part Number" to {"P-345", "P-400", "J-100"}
    end tell
end tell
```

setx

The **setx** command sets the value of a cell.

Arguments for the setx command

Argument	Description
to	The value to which the cell will be set.
[committing]	If `true`, the data is committed to the cell immediately and any check calculations or formatting options are triggered. If `false`, the data is not committed immediately. The default is `true`.
[lookup]	Specifies whether or not a lookup is performed. If `true`, the lookup is performed unconditionally. If `false`, the lookup is ignored. If not specified, a lookup is performed only if the cell's value has changed. This argument is ignored if the specified cell is not a lookup cell. **Note**: If `lookup` is specified (`true` or `false`), then `committing` must be `true`.

```
tell document 1 of application "Informed Filler"
    --set the prefix for a phone cell without committing
    setx value of cell "Phone" of current record to "(403) 555-" without committing
end tell


tell document 1 of application "Informed Filler"
    --set the value of a cell, commit the data without the configured lookup
    setx value of cell "Employee Number" of current record to "1012" without lookup

    --do some other stuff

    --commit the cell and force the lookup
    set current cell to cell "Employee Number"
    commit cell "Employee Number" of current record with lookup
end tell
```

sign

The **sign** command signs the indicated signature cells.

Arguments for the sign command

Argument	Description
[using]	Specifies which signing service to use. This argument can be a constant such as `Entrust`, `ISignIMAP`, or `ISignPOP` or a string that specifies the name of a signing plug-in. If not specified Informed uses the signing service selected on the user's Preferences dialog, or displays a dialog asking the user to select a signing service.

```
tell document 1 of application "Informed Filler"
    --sign the current record using the ISignPOP signing service
    sign cell "Signature" of current record using ISignPOP
end tell


tell document 1 of application "Informed Filler"
    --sign all records in the current collection using the Entrust signing service
    --See "Containment" earlier in this chapter for more information.
    sign cell "Signature" of current collection using Entrust
end tell
```

verify

The **verify** command verifies the indicated signature cells and returns `true` if the cell contains a valid signature, and `false` if the signature is not valid (or if the cell is not signed).

```
tell document 1 of application "Informed Filler"
    --verify the signature in the cell named "Approval" of the current record.
     set vValid to verify cell "Approval" of current record
end tell


tell document 1 of application "Informed Filler"
    --verify the signature in the cell named "Approval" of every record
    set current collection to every record
    set vValid to verify cell "Approval" of every record
end tell
```

collection

A **collection** object represents the current collection of records in a document.

Reference

A collection object can reference collections by:

- index
- id
- absolute position
- test

Properties

The following table lists the properties of a collection object.

collection properties

Property	Writeable?	Description
container	no	The container for the collection. A collection is always contained by a document.
id	no	The unique id of the collection
index	no	The index of the collection.
class	no	The collection class.

Commands

The following commands are defined for a collection object:

count

The **count** command returns the number of records in a collection.

Arguments for the count command

Argument	Description
each	This argument specifies the class of the elements to be counted. Its value must be record.

```
tell application "Informed Filler"
    set vCount to count each record in current collection of document 1
end tell
```

exists

The **exists** command verifies the existence of a collection.

```
tell document 1 of application "Informed Filler"
    set vExists to exists current collection
end tell
```

export

The **export** command exports every record in a collection to a file.

Arguments for the export command

Argument	Description
to	This argument must be a file object which specifies the file into which the data will be exported.
[using]	Specifies the cells which will be exported. This argument can be a single cell object, a list of cell objects, a single column, or a list of columns. If not specified, all cells will be exported. No container is allowed for this argument.
[as]	Specifies the file format of the export file. This argument can be one of the following constants: `Interchange`, `tab delimited`, or `comma delimited`, or a string that specifies a format name. The default value for the format argument is `Interchange`.
[append]	If `true`, the exported records are appended to the end of the export file. If `false`, the export file is replaced with the exported records. The default value is `false`.
[rowwise]	If `true`, then tables will be exported in row order. If `false`, tables will be exported in column order. The default value is `false`.
[quotes]	If `true`, then all exported values—except numbers—are surrounded with quotes. If `false`, then only those values which contain delimiter characters are surrounded with quotes. This argument is ignored if the format argument is not `tab delimited` or `comma delimited`.
[merge]	If `true`, Informed will list each cell name on the first line of a new export file. The default value is `true`. This argument is ignored if the format argument is not `tab delimited` or `comma delimited`.
[notes]	If `true`, then any notes attached to the form will be exported. The default value is `true`. This argument is ignored if the format argument is not `Interchange`.

```
tell document 1 of application "Informed Filler"
    --export current collection using the tab delimited format
    export current collection to file "HD:Invoice98" as tab delimited
end tell
```

make

The **make** command creates a new record in a collection.

Arguments for the make command

Argument	Description
new	This argument specifies the class of the new element. Its value must be record.
at	The location at which to insert the new element.

```
tell application "Informed Filler"
    make new record at current collection of document "School Suspensions"
end tell
```

print

The **print** command prints every record in a collection.

Arguments for the print command

Argument	Description
[as]	Specifies whether to print as forms or as a list. This argument can be either `forms` or `record list`. The default value is `forms`.
[copies]	Specifies the number of copies to print. The default value is 1.
[from page]	Specifies the page to start printing from. The default value is the first page.
[to page]	Specifies the page to stop printing at. The default value is the last page.
[from part]	Specifies which part of a multipart form to start printing from. The default value is the first part.
[to part]	Specifies which part of a multipart form to stop printing at. The default value is the last part.
[print template]	If no, then don't print the template. The default value is yes.
[print data]	If no, then don't print the data. The default value is yes.
[collate]	Specifies whether or not to collate pages. The default value is yes.

```
tell application "Informed Filler"
    --print 2 copies of the current collection of records as a list
    print current collection of document "Invoice" as record list copies 2
end tell
```

send

The **send** command sends every record in a collection using an electronic mail service.

Arguments for the send command

Argument	Description
[step]	Specifies the name or the index of the routing step to use. If not specified, no routing step is used. If a routing step is used, all other optional arguments override their corresponding settings in the routing step configuration.
[recipients]	Specifies the recipients. This argument can be a string or a list of strings. The default value is no recipients.
[cc recipients]	Specifies a list of recipients to cc. This argument can be a string or a list of strings. The default is no cc recipients.
[bcc recipients]	Specifies a list of recipients to bcc. This argument can be a string or a list of strings. The default is no bcc recipients.
[append recipients]	Specifies whether or not to append other recipients to those already specified in a routing step. The default value is `false`.
[subject]	Specifies the subject of the message. The default is the name of the document being sent.
[body]	The body of the mail message. The default value is no body.
[data format]	The format to send the form in. This argument can be one of the following constants: `data`, `package`, `Interchange`, `comma delimited`, or `tab delimited`, or a string that specifies the name of a format. The default value is `data`.
[enclose as]	Specifies the name of the form attachment. The default value is the name of the data document.
[message attachments]	Specifies any additional attachments. This argument can be a file object or a list of file objects.
[append attachments]	Specifies whether or not to append other attachments to those already specified in a routing step. The default value is `false`.
[using]	This argument is illegal if a routing step is provided. If no routing step is provided, this argument can be a constant or a string that specifies the name of a mail plug-in. If not specified the default mail system on the user's machine is used. Informed provides the following constants for mail systems on MacOS: `SMTP`, `Eudora`, `MSMail`, `CCMail`, `GroupWise`, `QuarterDeck`, `QuickMail`

```
tell application "Informed Filler"
    --send the current collection to the specified recipient
    send current collection of document 1 recipients "btaylor@worldcorp.com"
end tell


tell application "Informed Filler"
    --send the current collection using a particular routing step
    send current collection of document 1 step "Send to Accounting"
end tell


tell application "Informed Filler"
    --Send the current collection using the provided mail addressing
    --information.
    send current collection of document 1 recipients "gSmith@worldcorp.com" ¬
        subject "Time Sheet" body "Here is my time sheet" data format package
end tell
```

sort

The **sort** command sorts the records in the current collection.

Arguments for the sort command

Argument	Description
on	Specifies the cells or columns on which to sort. Its value must be a cell object, column object, or list of cell or column objects. If a list is specified, the current collection is sorted on each cell in the list, beginning with the first cell in the list and ending with the last. No container is allowed for this argument.
[descending]	Specifies whether or not to sort in descending order. The default is `false`.

```
tell application "Informed Filler"
    --sort the records on the cell "Quantity"
    sort current collection of document "Invoices" on cell "Quantity"
end tell

tell application "Informed Filler"
    --Sort the current collection of the document "Invoices" first by the cell
    --"F Name" and then by the cell "L Name".
    sort current collection of document "Invoices" on cell {"F Name", "L Name"}
end tell
```

column

Informed Filler's Record List is a standard window that displays records in a list format. Information on the Record List is divided into rows and columns. Each row in the list represents one record. Each column corresponds to one cell on the form. A **column** object represents one or more columns on the Record List.

Reference

A column object can reference columns by:

- name
- index
- id
- absolute position
- relative position
- range
- test

Properties

The following table lists the properties of a column object.

column properties

Property	Writeable?	Description
alignment	no	The alignment of the data in the column. This property can be one of the following constants: `left`, `center`, or `right`.
average	no	The average of the data in the column. This property only works with averaged columns.
cell ID	no	The id of the cell associated with the column.
container	no	The container for the column. A column is always contained by a format.
id	no	The unique id of the column.
index	no	The index of the column.
name	no	The name of the column.
class	no	The column class.
selected	no	Is the column selected?
sorted	no	Is the column sorted?
total	no	The total of the data in the column. This property only works with totalled columns.

totals type	no	The totals type of the column. This property can be one of the following constants: none, totalled, or averaged.
width	no	The width of the column in pixels.

Commands

The following commands are defined for a column object:

clear

The **clear** command clears the column of any data.

```
tell application "Informed Filler"
    clear column "Quantity" of current format of document 1
end tell
```

exists

The **exists** command verifies the existence a column.

```
tell application "Informed Filler"
    tell current format of document 1
        if exists column 3 then
            set totals type of column 3 to totalled
            set vTotal to total of column 3
        end if
    end tell
end tell
```

document

A **document** object represents an opened data document in Informed Filler.

Reference

A document object can reference documents by:

- name
- index
- id
- absolute position
- relative position
- range
- test

Properties

The following table lists the properties of a document object.

document properties

Property	Writeable?	Description
attachments window	no	The attachments window
author name	no	The template author's name as entered on the Template Information dialog.
author organization	no	The template author's organization as entered on the Template Information dialog.
check period	yes	The revision check period. This property can be one of the following constants: `every time`, `daily`, `weekly`, `monthly`, or `never`.
current cell	yes	The current cell.
current collection	yes	The current collection of records.
current format	yes	The current Record List format.
current page	yes	The current page of the form.
current record	yes	The current record in the document.
description	no	The description of the template as entered on the Template Information dialog.
disk file	no	The disk file that contains the data document.
distributed	no	Is the template distributed?
form window	no	The form window.
id	no	The unique id of the data document.
index	yes	The index of the data document.
last checked	no	When the last revision check occurred.
modified	no	Has the data document been modified?
name	no	The name of the data document.
class	no	The document class.
page count	no	The number of pages in the document.
record list window	no	The Record List window.
revision	no	The revision number of the template as entered on the Template Information dialog.

status	no	The status of the template. This property will be one of the following constants: `current`, `noncurrent`, or `discontinued`.
status message	no	The status message for the template as entered on the Template Information dialog.
template ID	no	The template document's template ID as entered on the Template Information dialog.
template name	no	The template document's template name as entered on the Template Information dialog.

Commands

The following commands are defined for a document object:

close

The **close** command closes a document.

Arguments for the close command

Argument	Description
[saving]	Specifies whether changes should be saved before closing. If `yes`, changes will be saved before closing. If `no`, changes will not be saved. If `ask`, Informed will display a dialog asking if the user wants to save the changes if necessary. The default value is `ask`.
[saving in]	Specifies the file in which to save the document. The default value is the file into which the document was previously saved. If the document was not previously saved, the standard Save dialog is displayed.

```
tell application "Informed Filler"
    --close the second document. Prompt the user to save.
    close document 2
end tell


tell application "Informed Filler"
    --close a specific document without saving.
    close document "Purchase Order" saving no
end tell
```

count

The **count** command returns the number of attached scripts, buttons, collections, formats, records, saved formats, or tags within a document.

Arguments for the count command

Argument	Description
each	This argument specifies the class of the elements to be counted. Its value must be attached script, button, collection, format, record, saved format, or tag.

```
tell application "Informed Filler"
    set vCount to count each record in document "Salary Increase Request"
end tell
```

exists

The **exists** command verifies the existence a document.

```
tell application "Informed Filler"
    if exists document "Attendance" then
        close document "Attendance" saving yes
    end if
end tell
```

export

The **export** command exports every record in a document to a file.

Arguments for the export command

Argument	Description
to	This argument must be a file object which specifies the file into which the data will be exported.
[using]	Specifies the cells which will be exported. This argument can be a single cell object, a list of cell objects, a single column, or a list of columns. If not specified all cells will be exported. No container is allowed for this argument.
[as]	Specifies the file format of the export file. This argument can be one of the following constants: `Interchange`, `tab delimited`, or `comma delimited`, or a string that specifies a format name. The default value for the format argument is `Interchange`.
[append]	If `true`, the exported records are appended to the end of the export file. If `false`, the export file is replaced with the exported records. The default value is `false`.
[rowwise]	If `true`, then tables will be exported in row order. If `false`, tables will be exported in column order. The default value is `false`.

[quotes]	If `true`, then all exported values—except numbers—are surrounded with quotes. If `false`, then only those values which contain delimiter characters are surrounded with quotes. This argument is ignored if the format argument is not `tab delimited` or `comma delimited`.
[merge]	If `true`, Informed will list each cell name on the first line of a new export file. The default value is `true`. This argument is ignored if the format argument is not `tab delimited` or `comma delimited`.
[notes]	If `true`, then any notes attached to the form will be exported. The default value is `true`. This argument is ignored if the format argument is not `Interchange`.

```
tell application "Informed Filler"
    --export every record of the current document as tab delimited
    export current document to file "HD:Documents:Invoices" as tab delimited
end tell
```

```
tell application "Informed Filler"
    --export every record of the current document excluding any notes
    export current document to file "HD:Documents:Invoices" notes no
end tell
```

import

The **import** command imports one or more files into a document.

Arguments for the import command

Argument	Description
into	Specifies the files to import into the document. Its value must be a file object, an alias object, or a list of file or alias objects.
[append]	If `true`, the imported records are appended to the end of the current collection. If `false`, the imported records become the current collection. The default value is `false`.

```
tell application "Informed Filler"
    --import a file and append the records to the current collection.
    import file "HD:Documents:Invoices98" into document 2 with append
end tell
```

make

The **make** command creates a new record, saved format, or tag in a document.

Arguments for the make command

Argument	Description
new	This argument specifies the class of the new element. Its value must be record.
at	The location at which to insert the new element.
[with properties]	Specifies the initial values for the properties of the new element. This argument is not used when creating a new record. This argument must be an AppleScript record with a name property that specifies the name of the new saved format or tag.

```
tell application "Informed Filler"
    --create a new record in the second document.
    make new record at document 2
end tell

tell application "Informed Filler"
    --set the current collection to all records that match a test, and create
    --a new tag for that collection.
    set current collection of document 1 to every record whose cell "City"¬
        is equal to "New York"
    make new tag at document 1 with properties {name:"NY Clients"}
end tell
```

print

The **print** command prints every record in a document.

Arguments for the print command

Argument	Description
[as]	Specifies whether to print as forms or as a list. This argument can be either `forms` or `record list`. The default value is `forms`.
[copies]	Specifies the number of copies to print. The default value is 1.
[from page]	Specifies the page to start printing from. The default value is the first page.
[to page]	Specifies the page to stop printing at. The default value is the last page.
[from part]	Specifies which part of a multipart form to start printing from. The default value is the first part.
[to part]	Specifies which part of a multipart form to stop printing at. The default value is the last part.
[print template]	If no, then don’t print the template. The default value is yes.

[print data]	If no, then don't print the data. The default value is yes.
[collate]	Specifies whether or not to collate pages. The default value is yes.

```
tell application "Informed Filler"
    --print a copy of every record in the current document
    print document 1
end tell
```

```
tell application "Informed Filler"
    --print 2 copies of the document named "Invoices" as a list
    print document "Invoices" as record list copies 2
end tell
```

save

The **save** command saves a document.

Arguments for the save command

Argument	Description
[in]	This argument must be a file object which specifies the file into which the document will be saved. The default value is the file into which the document was previously saved. If the document was not previously saved, the standard Save dialog is displayed.
[as]	The file type of the document in which to save the data.

```
tell application "Informed Filler"
    save document 1
end tell
```

```
tell document 1 of application "Informed Filler"
    --save in a specific file as a package
    save in file "HD:Documents:Invoices98" as package
end tell
```

search

The **search** command searches for specific records in a document.

Arguments for the search command

Argument	Description
in	Specifies the cell to search in. This argument cannot have a container and must be either a single cell or column.
for	The value to search for. This argument must be a list of two values if the match option is RNG.
[match option]	The match option. This argument will be one of the following constants: EQ (is equal to) NE (is not equal to) LT (less than) LE (less than or equal to) GT (greater than) GE (greater than or equal to) BEG (begins with) END (ends with) CON (contains) RNG (range) The default value is RNG if the in argument is a list; CON if the in argument is a text, character or name cell; or EQ otherwise.
[find option]	Specifies which records to look through and what to do with the records found. This argument will be one of the following constants: all records (look through all records) collected records (look through collected records) add to collection (add to current collection) omit from collection (omit from collection) first match (go to first match in collection) select matches (select matches in Record List). The default value is all records.

```
tell application "Informed Filler"
    --Set the current collection to every record that matches a test.
    search document 1 in cell "Name" for "Smith"
end tell
```

```
tell application "Informed Filler"
    --find all records that match a test and add them to the current collection.
    search document 1 in cell "Name" for "Thomson" find option add to collection
end tell
```

send

The **send** command sends every record in a document using an electronic mail service.

Arguments for the send command

Argument	Description
[step]	Specifies the name or the index of the routing step to use. If not specified, no routing step is used. If a routing step is used, all other optional arguments override their corresponding settings in the routing step configuration.
[recipients]	Specifies the recipients. This argument can be a string or a list of strings. The default value is no recipients.
[cc recipients]	Specifies a list of recipients to cc. This argument can be a string or a list of strings. The default is no cc recipients.
[bcc recipients]	Specifies a list of recipients to bcc. This argument can be a string or a list of strings. The default is no bcc recipients.
[append recipients]	Specifies whether or not to append other recipients to those already specified in a routing step. The default value is `false`.
[subject]	Specifies the subject of the message. The default is the name of the document being sent.
[body]	The body of the mail message. The default value is no body.
[data format]	The format to send the form in. This argument can be one of the following constants: `data`, `package`, `Interchange`, `comma delimited`, or `tab delimited`, or a string that specifies the name of a format. The default value is `data`.
[enclose as]	Specifies the name of the form attachment. The default value is the name of the data document.
[message attachments]	Specifies any additional attachments. This argument can be a file object or a list of file objects.
[append attachments]	Specifies whether or not to append other attachments to those already specified in a routing step. The default value is `false`.

[using]	This argument is illegal if a routing step is provided. If no routing step is provided, this argument can be a constant or a string that specifies the name of a mail plug-in. If not specified the default mail system on the user's machine is used. Informed provides the following constants for mail systems on MacOS: `SMTP`, `Eudora`, `MSMail`, `CCMail`, `GroupWise`, `QuarterDeck`, `QuickMail`

```
tell application "Informed Filler"
    --send every record in the document
    send document 1
end tell


tell application "Informed Filler"
    --send every record using a particular routing step
    send document 1 step "Send to Accounting"
end tell


tell application "Informed Filler"
    --Send every record using the provided mail addressing
    --information.
    send document 1 recipients "gSmith@worldcorp.com" subject "Time Sheet"¬
            body "Here is my time sheet" data format package
end tell
```

element

An **element** object refers to an element of a cell. A field cell contains only one element, whereas a table cell contains one element for each row of the table.

Reference

An element object can reference elements by:

- index
- absolute position
- relative position
- range
- test

Properties

The following table lists the properties of an element object.

element properties

Property	Writeable?	Description
container	no	The container for the element. An element is always contained by a cell.
index	no	The index of the row element.
class	no	The element class.
value	yes	The value of the row element.

Commands

The following commands are defined for an element object:

clear

The **clear** command clears the element of any data.

```
tell document 1 of application "Informed Filler"
    --clear the third element of the table cell named "Price" in the current record.
    clear element 3 of cell "Price" of current record
end tell
```

commit

The **commit** command commits the element data, triggering any formatting, check calculations or lookups configured for the element.

Arguments for the commit command

Argument	Description
[lookup]	Specifies whether or not a lookup will be performed. If `true`, the lookup is performed unconditionally. If `false`, the lookup is ignored. If not specified a lookup is performed only if the element's value has changed. This argument is ignored if the specified element does not belong to a lookup cell.

```
tell document 1 of application "Informed Filler"
    --set the value of an element, commit the data without the configured lookup
    setx value of element 3 of cell "Part" of current record to "P1" without lookup

    --do some other stuff

    --commit the element and force the lookup
    set current cell to element 3 of cell "Part" of current record
    commit element 3 of cell "Part" of current record with lookup
end tell
```

data size

The **data size** command returns the size of the element data in bytes.

Arguments for the data size command

Argument	Description
as	This argument specifies the data type for which the size is calculated.

```
tell application "Informed Filler"
    tell cell "Age" of current record of document 1
        set vSize to data size of element 2 as text
    end tell
end tell
```

exists

The **exists** command verifies the existence an element.

```
tell document 1 of application "Informed Filler"
    if exists element 11 of cell "Quantity" of current record then
        clear element 11 if cell "Quantity" of current record
    end if
end tell
```

get

The **get** command gets the value of an element.

Arguments for the get command

Argument	Description
as	The data type to be returned.

```
tell application "Informed Filler"
    tell current record of document "Order Form"
        --get the value of element 3 of the table cell named "Birthday" as text
        set vSignedDate to get element 3 of cell "Birthday" as text
    end tell
end tell
```

set

The **set** command sets the value of an element.

Arguments for the set command

Argument	Description
to	The value to which the element will be set.

```
tell document 1 of application "Informed Filler"
    --set the value of a single element in the table cell named "Price"
    set value of element 3 of cell "Price" of current record to "100.00"
end tell
```

setx

The **setx** command sets the value of an element.

Arguments for the setx command

Argument	Description
to	The value to which the element will be set.
[committing]	If `true`, the data is committed to the element immediately and any check calculations or formatting options are triggered. If `false`, the data is not committed immediately. The default is `true`.
[lookup]	Specifies whether or not a lookup is performed. If `true`, the lookup is performed unconditionally. If `false`, the lookup is ignored. If not specified, a lookup is performed only if the element's value has changed. This argument is ignored if the specified element does not belong to a lookup cell. **Note**: If lookup is specified (`true` or `false`), then committing must be `true`.

```
tell document 1 of application "Informed Filler"
    --set the value of a single element in the table cell named "Price"
    setx value of element 3 of cell "Price" of current record to "100.00"
end tell

tell document 1 of application "Informed Filler"
    --set the value of a single element of a table cell for multiple records
    --and perform the configured lookup
    set current collection to every record
    setx value of element 1 of cell "Part Number" of every record to "P-200"¬
           with lookup
end tell
```

format

A **format** object represents the current Record List format.

Reference

A format object can reference formats by:

- index
- id
- absolute position
- test

Properties

The following table lists the properties of a format object.

Format Properties

Property	Writeable?	Description
container	no	The container for the format. A format is always contained by a document.
id	no	The unique id for the format.
index	no	The index of the format.
class	no	The format class.
totals visible	yes	Are the totals for the columns visible?

Commands

The following commands are defined for a format object:

count

The **count** command returns the number of columns in a format.

Arguments for the count command

Argument	Description
each	This argument specifies the class of the elements to be counted. Its value must be column.

```
tell application "Informed Filler"
    set vCount to count each column in current format of current document
end tell
```

exists

The **exists** command verifies the existence of a format.

```
tell application "Informed Filler"
    set vExists to exists current format of document "Payroll"
end tell
```

menu

A **menu** object represents one or more menus in Informed Filler's menu bar.

Reference

A menu object can reference menus by:

- name
- index
- absolute position
- relative position
- range
- test

Properties

The following table lists the properties of a menu object.

menu properties

Property	Writeable?	Description
enabled	yes	Is the menu enabled?
index	no	The index of the menu.
name	no	The name of the menu.
class	no	The menu class.

Commands

The following commands are defined for a menu object:

count

The **count** command returns the number of menu items within a menu.

Arguments for the count command

Argument	Description
each	This argument specifies the class of the elements to be counted. Its value must be menu item.

```
tell application "Informed Filler"
    set vCount to count each menu
end tell
```

exists

The **exists** command verifies the existence of a menu.

```
tell application "Informed Filler"
    if exists menu 5 then
        set vName to name of menu 5
    end if
end tell
```

menu item

A **menu item** object represents one or more menu items in a particular menu.

Reference

A menu item object can reference menu items by:

- name
- index
- absolute position
- relative position
- range
- test

Properties

The following table lists the properties of a menu item object.

menu item properties

Property	Writeable?	Description
container	no	The container for the menu item. A menu item is always contained by a menu.
enabled	yes	Is the menu item enabled?
index	no	The index of the menu item.
name	no	The name of the menu item.
class	no	The menu item class.

Commands

The following commands are defined for a menu item object:

count

The **count** command returns the number of menu items within a menu item.

Arguments for the count command

Argument	Description
each	This argument specifies the class of the elements to be counted. Its value must be menu item.

```
tell application "Informed Filler"
    set vTagCount to count each menu item in menu item "Tags" of menu "Database"
end tell
```

execute

The **execute** command executes a menu item's configured action.

```
tell application "Informed Filler"
    --execute the fourth menu item of the Window menu
    execute menu item "Show Record List" of menu "Window"
end tell
```

exists

The **exists** command verifies the existence of a menu item.

```
tell application "Informed Filler"
    if exists menu item "Duplicate" of menu "Database" then
        execute menu item "Duplicate" of menu "Database"
    end if
end tell
```

plugin command

Some of Informed Filler's features are made available by installing Informed plug-ins. Certain plug-ins have commands associated with them. A **plugin command** object represents one or more plug-in commands in Informed Filler.

Reference

A plugin command object can reference plug-in commands by:

- name
- index
- id
- absolute position
- relative position
- range
- test

Properties

The following table lists the properties of a plugin command object.

plugin command properties

Property	Writeable?	Description
enable	yes	Is the plug-in command enabled?
id	no	The unique id of the plug-in command.
index	no	The index of the plug-in command.
name	no	The name of the plug-in command.
class	no	The plugin command class.

Commands

The following commands are defined for a plugin command object:

execute

The **execute** command executes a plug-in command.

Arguments for the execute command

Argument	Description
[with data]	Specifies the data string required by the plug-in.

```
tell application "Informed Filler"
    execute plugin command vPluginName with data vPluginData
end tell
```

exists

The **exists** command verifies the existence of a plug-in command.

```
tell application "Informed Filler"
    if exists plugin command vPluginName then
        execute plugin command vPluginName with data vPluginData
    end if
end tell
```

record

In Informed Filler, completed forms are stored as records in a data document. A **record** object represents one or more records in a data document.

Reference

A record object can referenced records by:

- index
- id
- absolute position
- relative position
- range
- test

Properties

The following table lists the properties of a record object.

record properties

Property	Writeable?	Description
attachment count	no	The number of attachments enclosed in the record.
container	no	The container for the record. A record is always contained by a document.
created	no	When the record was created.
id	no	The unique id of the record.
index	no	The index of the record.
last mailed	no	When the record was last mailed.
last modified	no	When the record was last modified.
last printed	no	When the record was last printed.
modified	no	Has the record been modified?
class	no	The record class.
selected	yes	Is the record selected in the Record List?

Commands

The following commands are defined for a record object:

clear

The **clear** command clears the record of any data.

```
tell application "Informed Filler"
    clear current record of document 1
end tell
```

commit

The **commit** command commits the record data to the document.

```
tell application "Informed Filler"
    commit current record of document "Invoices"
end tell
```

count

The **count** command returns the number of attachments or cells within a record.

Arguments for the count command

Argument	Description
each	This argument specifies the class of the elements to be counted. Its value must be attachment or cell.

```
tell application "Informed Filler"
    set vTagCount to count each attachment in record 5 of document "Order Form"
end tell
```

duplicate

The **duplicate** command duplicates a record.

```
tell application "Informed Filler"
    duplicate current record of document 1
end tell
```

exists

The exists command verifies the existence of a record.

```
tell document 1 of application "Informed Filler"
    set current collection to every record
    if exists record 5 of current collection then
        delete record 5 of current collection
    end if
end tell
```

export

The **export** command exports a record to a file.

Arguments for the export command

Argument	Description
[to]	This argument must be a file object which specifies the file into which the data will be exported.
[using]	Specifies the cells which will be exported. This argument can be a single cell object, a list of cell objects, a single column, or a list of columns. If not specified, all cells will be exported. No container is allowed for this argument.
[as]	Specifies the file format of the export file. This argument can be one of the following constants: `Interchange`, `tab delimited`, or `comma delimited`, or a string that specifies a format name. The default value for the format argument is `Interchange`.

[append]	If `true`, the exported records are appended to the end of the export file. If `false`, the export file is replaced with the exported records. The default value is `false`.
[rowwise]	If `true`, then tables will be exported in row order. If `false`, tables will be exported in column order. The default value is `false`.
[quotes]	If `true`, then all exported values—except numbers—are surrounded with quotes. If `false`, then only those values which contain delimiter characters are surrounded with quotes. This argument is ignored if the format argument is not `tab delimited` or `comma delimited`.
[merge]	If `true`, Informed will list each cell name on the first line of a new export file. The default value is `true`. This argument is ignored if the format argument is not `tab delimited` or `comma delimited`.
[notes]	If `true`, then any notes attached to the form will be exported. The default value is `true`. This argument is ignored if the format argument is not `Interchange`.

```
tell application "Informed Filler"
    export every record of document 2 to file "HD:Accounting:Invoice98"
end tell

tell application "Informed Filler"
    --export current record using the tab delimited format
    export current record of document "Invoice" to file "HD:Accounting:Invoice98"¬
            as tab delimited append true
end tell
```

make

The **make** command creates a new attachment in a record.

Arguments for the make command

Argument	Description
new	This argument specifies the class of the new element. Its value must be record.
at	The location at which to insert the new element.
with data	The initial data for the new element.
with properties	Specifies the initial values for the properties of the new element. This argument is not used when creating a new record. This argument must be an object with a name property that specifies the name of the new saved format or tag.

```
tell document 1 of application "Informed Filler"
    make new attachment at current record with data file "HD:Documents:Read Me"
end tell
```

omit

The **omit** command omits a record from the current collection.

```
tell document 1 of application "Informed Filler"
    omit record 1 of current collection
end tell


tell document 1 of application "Informed Filler"
    set current collection to every record
    omit every record of current collection whose cell "Quantity" is ¬
            equal to "100"
end tell
```

print

The **print** command prints a specified record.

Arguments for the print command

Argument	Description
[as]	Specifies whether to print as forms or as a list. This argument can be either `forms` or `record list`. The default value is `forms`.
[copies]	Specifies the number of copies to print. The default value is 1.
[from page]	Specifies the page to start printing from. The default value is the first page.
[to page]	Specifies the page to stop printing at. The default value is the last page.
[from part]	Specifies which part of a multipart form to start printing from. The default value is the first part.
[to part]	Specifies which part of a multipart form to stop printing at. The default value is the last part.
[print template]	If no, then don't print the template. The default value is yes.
[print data]	If no, then don't print the data. The default value is yes.
[collate]	Specifies whether or not to collate pages. The default value is yes.

```
tell document 1 of application "Informed Filler"
    --print a copy of the current record
    print current record
end tell


tell document 1 of application "Informed Filler"
    --print 2 copies of the records 1 through 5 of document 1
    print records 1 thru 5 as record list copies 2
end tell
```

delete

The **delete** command deletes a record from a document.

```
tell document 1 of application "Informed Filler"
    --Delete the current record of the current document
    delete current record
end tell

tell document 1 of application "Informed Filler"
    --delete all records that match a test
    delete every record whose cell "Quantity" = 5
end tell
```

revert

The **revert** command restores a record to its last saved state.

```
tell application "Informed Filler"
    revert current record of document 1
end tell
```

send

The **send** command sends a record using an electronic mail service.

Arguments for the send command

Argument	Description
[step]	Specifies the name or the index of the routing step to use. If not specified, no routing step is used. If a routing step is used, all other optional arguments override their corresponding settings in the routing step configuration.
[recipients]	Specifies the recipients. This argument can be a string or a list of strings. The default value is no recipients.
[cc recipients]	Specifies a list of recipients to cc. This argument can be a string or a list of strings. The default is no cc recipients.
[bcc recipients]	Specifies a list of recipients to bcc. This argument can be a string or a list of strings. The default is no bcc recipients.
[append recipients]	Specifies whether or not to append other recipients to those already specified in a routing step. The default value is `false`.
[subject]	Specifies the subject of the message. The default is the name of the document being sent.
[body]	The body of the mail message. The default value is no body.

[data format]	The format to send the form in. This argument can be one of the following constants: `data`, `package`, `Interchange`, `comma delimited`, or `tab delimited`, or a string that specifies the name of a format. The default value is `data`.
[enclose as]	Specifies the name of the form attachment. The default value is the name of the data document.
[message attachments]	Specifies any additional attachments. This argument can be a file object or a list of file objects.
[append attachments]	Specifies whether or not to append other attachments to those already specified in a routing step. The default value is `false`.
[using]	This argument is illegal if a routing step is provided. If no routing step is provided, this argument can be a constant or a string that specifies the name of a mail plug-in. If not specified the default mail system on the user's machine is used. Informed provides the following constants for mail systems on MacOS: `SMTP`, `Eudora`, `MSMail`, `CCMail`, `GroupWise`, `QuarterDeck`, `QuickMail`

```
tell application "Informed Filler"
    --send the current record
    send current record of document 1
end tell


tell application "Informed Filler"
    --send the current record using a particular routing step
    send current record of document 1 step "Send to Accounting"
end tell


tell application "Informed Filler"
    --Send the current record using the provided mail addressing
    --information.
    send current record of document 1 recipients "gSmith@worldcorp.com" ¬
            subject "Time Sheet" body "Here is my time sheet." data format package
end tell
```

saved format

Informed Filler's Record List window can be customized to display data in a variety of different formats. A **saved format** object represents one or more saved Record List formats.

Reference

A saved format object can reference saved formats by:

- name
- index
- id
- absolute position
- relative position
- range
- test

Properties

The following table lists the properties of a saved format object.

saved format properties

Property	Writeable?	Description
container	no	The container for the saved format. A saved format is always contained by a document.
id	no	The unique id for the saved format.
index	no	The index of the saved format.
name	no	The name of the saved format.
class	no	The saved format class.

Commands

The following commands are defined for a saved format object:

exists

The **exists** command verifies the existence of a saved format.

```
tell document 1 of application "Informed Filler"
    set vExists to exists saved format "Cash Report"
end tell
```

delete

The **delete** command deletes a saved format from a document.

```
tell document "Tax Form" of application "Informed Filler"
    delete saved format "Internal Report"
end tell
```

tag

Informed Filler's tag feature provides an easy way to identify unique collections of records so that they can be quickly recalled and viewed. A **tag** object represents one or more tags in a document.

Reference

A tag object can reference tags by:

- name
- index
- id
- absolute position
- relative position
- range
- test

Properties

The following table lists the properties of a tag object.

tag properties

Property	Writeable?	Description
container	no	The container for the tag. A tag is always contained by a document.
id	no	The unique id for the tag.
index	no	The index of the tag.
name	no	The name of the tag.
class	no	The tag class.

Commands

The following commands are defined for a tag object:

exists

The exists command verifies the existence of a tag.

```
tell document 1 of application "Informed Filler"
    if exists tag "Invoices" then
        delete tag "Invoices"
    end if
end tell
```

delete

The **delete** command deletes a tag from a document.

```
tell document 1 of application "Informed Filler"
    --delete the tag named invoices
    delete tag "Invoices"
end tell


tell document 1 of application "Informed Filler"
    --Delete every tag in the current document and set the current collection to
    --all records in the document
    delete every tag
    set current collection to every record
end tell
```

template

A **template** object represents the form template used by a data document.

Reference

A template object can reference templates by:

- name
- index
- id
- absolute position
- relative position
- range
- test

Properties

The following table lists the properties of a template object.

template properties

Property	Writeable?	Description
disk file	no	The disk file that contains the template
id	no	The template id for the template as entered on the Template Information dialog.
index	no	The index of the template.
name	no	The template name of the template as entered on the Template Information dialog.
class	no	The template class.
revision	no	The revision number of the template as entered on the Template Information dialog.

Commands

The following commands are defined for a template object:

exists

The **exists** command verifies the existence of a template.

```
tell application "Informed Filler"
    set vExists to exists template id "InvoiceWC95"
end tell
```

window

A **window** object represents a window in Informed Filler.

Reference

A window object can reference windows by:

- name
- index
- id
- absolute position
- relative position
- range
- test

Properties

The following table lists the properties of a window object.

window properties

Property	Writeable?	Description
bounds	yes	The boundary rectangle for the window.
closeable	no	Does the window have a close box?
floating	no	Is the Window a floating window?
id	no	The unique ID of the window.
index	yes	The number of the window.
kind	no	The window kind. This property can be one the following constants: `attachments kind`, `clipboard kind`, `form kind`, or `record list kind`.
modal	no	Is the window modal?
name	no	The title of the window.
class	no	The window class.
parent document	no	The document to which the window belongs.
resizable	no	Is the window resizable?
titled	no	Does the window have a title bar?
visible	yes	Is the window visible?
zoomable	no	Is the window zoomable?
zoomed	yes	Is the window zoomed?

Commands

The following commands are defined for a window object:

close

The **close** command closes a window.

Arguments for the close command

Argument	Description
[saving]	Specifies whether changes should be saved before closing. If `yes`, changes will be saved. If `no`, changes will not be saved. If `ask`, Informed will display a dialog asking if the user wants to save the changes. The default value is `ask`. This argument is ignored if the window is not a form window.

[saving in]	Specifies the file in which to save the window. The default value is the file into which the window's parent document was previously saved. It the document was not previously saved, the standard Save dialog is displayed.

```
tell application "Informed Filler"
    --Close the frontmost window and prompt to save (if it is a form window).
    close window 1
end tell


tell application "Informed Filler"
    --Close the record list window of the current document.
    close record list window of document 1
end tell


tell application "Informed Filler"
    --close the frontmost window without saving
    close window 1 saving no
end tell
```

exists

The **exists** command verifies the existence of a window.

```
tell application "Informed Filler"
    set vExists to exists window "Invoice"
end tell
```

open

The **open** command opens a window and returns the resulting window object.

```
tell application "Informed Filler"
    open attachments window of document 2
end tell
```