SP Form Assistant, v. 3.0

SP Form Assistant, a FrameMaker-to-Acrobat TimeSavers add-on, enables the insertion of Acrobat form fields and control over their properties through hypertext markers included in FrameMaker documents.

Familiarity with the uses of PDF form fields is assumed.

Documentation:

- *SP-FormGuide.pdf (this file)*
- *SP-FormQref.pdf*

Software

- *SP-FormAsst1.ini*
- *SP-FormAsst2.ini*
- *SP-FormDefs.txt*
- *MarkerBuilders-FormAsst30.zip (only with FM v10 or later)*

Demos:

- *SP-FormDemo.fm*
- *SP-FormDemo-result.pdf*
- *SP-RolloverPop-demo.fm*
- *SP-RolloverPop-demo-result.pdf*

Installation

1 Unzip the `SP-FormAsst` zip distribution file into a temporary folder.

2 Place the `SP-FormAsst1.ini`, `SP-FormAsst2.ini` and `SP-FormDefs.txt` files in the `C:\TimeSavers` folder.

3 Turn on Form Assistant through the Assistants tab in the TimeSavers settings dialog box (or the `SPTS.set` settings file).

FrameMaker 10 or later: Installing the Marker Builder ExtendScript

Unzip the `MarkerBuilders-FormAsst30.zip` to the target folder on your hard drive:
`C:\Users\<UserName>\AppData\Roaming\Adobe\FrameMaker\<VersionNumber>\startup`
where `<UserName>` is your user name, and `<VersionNumber>` is the number of the FrameMaker version being used.

Restart FrameMaker and you should now see a FM2PDF menu.

Note: You can also locate the target folder as follows:
Start > Run (or Windows key + R), and then type `%appdata%`
Click Adobe, then click FrameMaker, then click the version number, then click Startup.

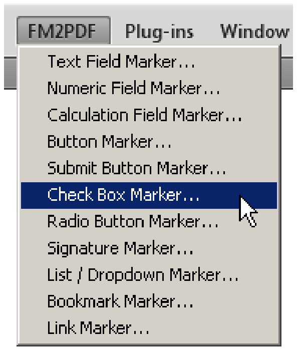
FrameMaker 10 or later: Inserting form fields markers through the FM2PDF menu

The custom hypertext markers are inserted at the beginning of the current paragraph. For the best control over the active area in the PDF, insert the marker in an otherwise empty table cell or text frame.

When placing custom markers in table cells through the menu, the insertion point is moved to the table cell below after the insertion.

Whenever possible, default values for various items in the dialog boxes are used when the fields are empty; for example, leaving the field name empty will result in * being used (=autoname), or default field appearance.

Use Special > Hypertext to edit the markers inserted through the FM2PDF menu.



Converting Files to PDF

After installation, when Saving as PDF (or distilling PostScript files) with FrameMaker-to-Acrobat TimeSavers active, you will see the following messages in the Distiller message window (or in the log file):

```
*** SP Form Assistant, v. 3.0
*** Reading SP-FormDefs.txt  . . . done
```

When distilling is completed, post-processing steps related to tab order and field calculation order are performed, and the following messages are displayed:

```
*** SP Form Assistant 3.0: Processing Fields, Tab Order and Field Calculation Order
*** Total number of form fields created: nnn
```

Note that only if specific markers enabled by the Assistant are actually present in the source FrameMaker files will there be corresponding form fields in the resulting PDFs.

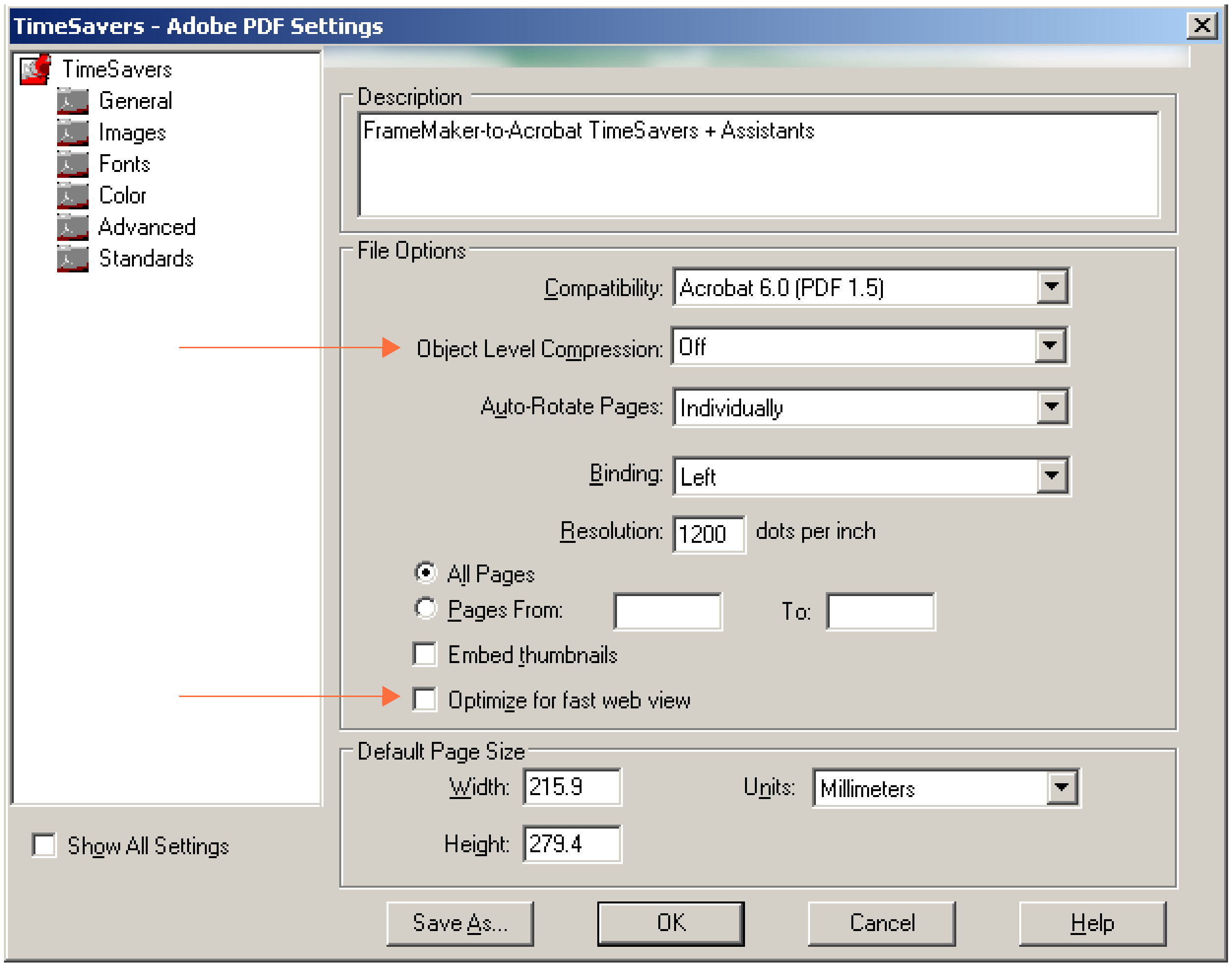
Demo Files

- The `SP-FormDemo.fm` file includes many markers defining the different field types. Convert this to a PDF with the TimeSavers/Form Assistant active, and inspect the results. You can compare the results to the PDF supplied showing the expected outcome (`SP-FormDemo-result.pdf`).
- The `SP-RolloverPop-demo.fm` file demonstrates rollovers/popups. When distilling it, make sure that the ConvertFMnewlink function in the TimeSavers is enabled, so that button destinations are properly defined through the `newlink TMS` markers included in the demo file. If the variable popups produced with `~xPop` are produced incorrectly in terms of size, update the driver resolution setting in the TimeSavers' settings dialog box (or `/PrinterDriverResolution` in the SPTS.set file), so that it matches the resolution of the printer driver used to produce the PS file being distilled.

Recommended Distiller settings

In your Distiller job options used in conjunction with TimeSavers/Form Assistant, General section, the following settings are recommended:

- Set Object Level Compression to Off
- Turn off Optimize for fast web view



Description: FrameMaker-to-Acrobat TimeSavers + Assistants

File Options
Compatibility: Acrobat 6.0 (PDF 1.5)
Object Level Compression: Off
Auto-Rotate Pages: Individually
Binding: Left
Resolution: 1200 dots per inch
[✓ All Pages]
Pages From: To:
☐ Embed thumbnails
☐ Optimize for fast web view

Default Page Size
Width: 215.9 Units: Millimeters
Height: 279.4

☐ Show All Settings

If you do not turn off Object Level Compression, the following distilling error may be encountered:

```
%%%%[ Error generating pdf file. Please retry with these
features turned off: optimize CompressObjects ]%%%%
```

Page Height Definition

Starting with Form Assistant version 3, ***it is necessary to define the page height***. You can specify a default page height in the `SP-FormDefs.txt` file and override it as necessary in specific files through hypertext markers. If the specified page height does not match the PDF page height, form fields will be misplaced (and sometimes even be placed outside the page).

Specifying Default Page Height in SP-FormDefs.txt file

At the beginning of the settings file, you will find the following lines:

```
%%% DEFAULT PAGE HEIGHT (in inches):
globaldict begin
  11  % Only change this line
/TS-FormHeight TSset end
```

Defining Page Height through Hypertext Markers

To override the page height definition for specific FrameMaker files, insert this hypertext marker:

```
alert ~FormHeight 6
```

“Save?” message when closing the PDF

PDFs distilled with form fields trigger a “Save” behavior, even when nothing has been changed in the PDF file. To prevent this, use one of the following techniques:

- Open the PDF, and perform a “Save As” as the last step, overwriting the same file (highly recommended in any case with all PDFs).
- Add a “page action” which clears the “save needed” property when the PDF is opened. If you really change the document afterwards, you will still be prompted for save.
  This can be added to individual documents through the following hypertext marker:
  ```
  alert ~JSPage (this.dirty = false)
  ```
  Or you can add the following line to the `TS-pre.ini` file (to make it a default setting in all documents being distilled):
  ```
  [ TS~jspage~1 (this.dirty = false) TS~jspage~2 pdf
  ```

If you are distilling batches of files which do not have form fields of any type, consider turning off the Form Assistant. Quite apart from triggering a Save, Acrobat’s “Form Dictionary” interferes with “byteserving” which is generally useful if the PDFs are placed on a web site.

Field Marker Syntax

Fields are specified through hypertext `alert` markers, using the following general syntax:

```
alert ~XX (Field Name) /FieldAppearance (Tooltip Text) ... Field-specific params ...
```

For example, the `alert ~BUN` marker defines a button with a named action:

```
alert ~BUN (bu2) /rg2s-bC5-va (Print document) (Print) /Print /i
```

When the resulting button is clicked, Acrobat's Print dialog box appears.

(All of Acrobat's named actions can be specified through the `alert ~BUN` marker.)

The field is created using FrameMaker's active area size. To control this size in the context of forms, it is recommended to place the markers in an otherwise empty table cells or text frames.

Field area can be adjusted globally through the SP-FormDefs.txt file – increased or decreased by the specified number of points (negative values shrink the area, positive values extend the areas):

```
1.5     /TSFA_FieldChangeArea    TSset
```

Error Handling

When distilling with the TimeSavers/Form Assistant, all form-related markers are checked for validity:

If a marker is …

- missing a parameter
- has an extra parameter
- using a parameter of a wrong type (e.g. number instead of text string)

… an error message is displayed in the Distiller log.

The PDF is created, and Acrobat notes indicate any incorrect markers and additional information about the type of errors encountered. Incorrect form shortcuts do not fail the distilling of the PDF.

However, undefined names will cause a distilling error. For example, parameters such as `(XYZ)` or `/XYZ` will not cause a distilling error, but `XYZ` will cause this error:

```
%%[ Error: undefined; OffendingCommand: XYZ ]%%
```

When parameters reference a value not defined in `SP-FormDefs.txt` (for color, font, etc.), or have an incorrect value in the case of predefined Acrobat settings, default settings are applied without causing an error.

Default values are used as follows:

- Color: black
- Border width: 4-9 results in no line visible; non-numeric values are interpreted as thin
- Border style: solid
- Size: 11
- Font: Helvetica
- Currency definition: `2 2 2 ("$") /Pre`

Note: If your form has identical markers on the same page and some fields are missing altogether from your PDF, turn off the `/DeleteDuplicateStrict` option in the TimeSavers' settings file.

Tab Order

Fields are created in the order of hypertext marker insertion, per page. The field creation order affects the tabbing order in the PDF file, and may appear to be “random”.

There are two ways to control the tabbing order, depending on your form layout and needs.

Row / Column Order

You can add custom markers to set the tabbing order by rows or columns:

- `alert ~TabOrderRow`
- `alert ~TabOrderCol`

These correspond to the action you can set manually in Acrobat (when you right-click the page thumbnail and select Page Properties > Tab Order); (repeat manual operations if the form is recreated).

Typically, `alert ~TabOrderRow` is used for most layouts.

The Acrobat action set by these markers is limited to the current page. In multi-page forms, the marker ***should be present in every page***. The easiest way to do so is to place the marker on all master pages (e.g. paste it in the header or footer frame, even if otherwise empty).

User-Specified Order

If your form layout has multiple columns, or is otherwise complex, the simple Row/Column tabbing order will likely not work for your form.

You can control the tabbing order precisely by specifying the sequence through a number placed at the beginning of the tooltip, followed by the ^ (circumflex) character (this number serves only to control the tabbing order and will not be displayed as part of the tooltip).

For example, the following markers:

```
alert ~TFS  (Name.First) TSFA (00010^First Name) ( ) /L
alert ~TFS  (Name.Last) TSFA (00020^Last Name) ( ) /L
```

create two fields; regardless of the insertion order in FrameMaker, pressing the tab key will take the user to the Name.First field first, followed by Name.Last.

If this marker is added later:

```
alert ~TFS (Name.Middle) TSFA (00015^Middle Name) ( ) /L
```

the tabbing sequence will be: Name.First > Name.Middle > Name.Last.

Additional fields (if present), for which a tabbing order number is not specified, will follow.

Use the same number of digits (e.g. 5 or 6), with leading zeroes as needed.

When creating the initial set of markers, it is recommended to have gaps in the sequence number (e.g. `00010, 00020, 00030`), so that new fields can be added without having to rearrange the entire sequence (the new fields, for example, may use sequence numbers of `00012, 00014`).

Automatic Field Highlighting for the Current Field

Especially in forms with many fields, it may be easier for end users to get a visual indication of the current field (the one that has the focus, either through clicking or tabbing).

You can specify a temporary change in the border color, background color, border width or text color. These properties will change when the user clicks on them or tabs into them, and upon exiting the fields, the original field properties are restored.

This feature is controlled through the `SP-FormDefs.txt` file, and applies to all fields. For example:

```
/gf1h /TS_FieldHighlight TSset
```

The Field Highlight string is composed of the following items:

/	c	b	w	t
	border color *(single letter)*	background color *(single letter)*	width *(single digit)*	text color *(single letter)*

Use a `*` to indicate that the specific property is unchanged.

Having a highlight string of `/****` means that automatic field highlighting is effectively turned off (default).

Parameters Used in Markers

(FieldName)

Every field must have a unique name. Names are case-sensitive.

When you have two or more fields of the same type with the same name, they become identical – changing one will affect the other(s).

When there is an existing field and you try to create new fields of a different type using the same name – these new fields will be suppressed by Acrobat and will not be created.

When a name is specified as `(*)`, the field is named automatically: it is given a specific prefix based on the field type, followed by the sequential Acrobat page number (based on the PDF page number, and not on the FrameMaker page number), followed by a serial number.

Prefixes for autonamed fields include:

- *BU*
- *TX*
- *NUM*
- *NumVal*
- *DATE*
- *TIME*
- *PCT*
- *ZIP*
- *ZIP4*
- *PHONE*
- *SSN*
- *LIST*

The first autoname for a button on page 1 will be `BU1-1`. Notice that the sequential number of the button on a specific page (e.g. `BU1-2`, `BU1-3`) reflects the order of marker creation in FrameMaker, which may not match the position on a page.

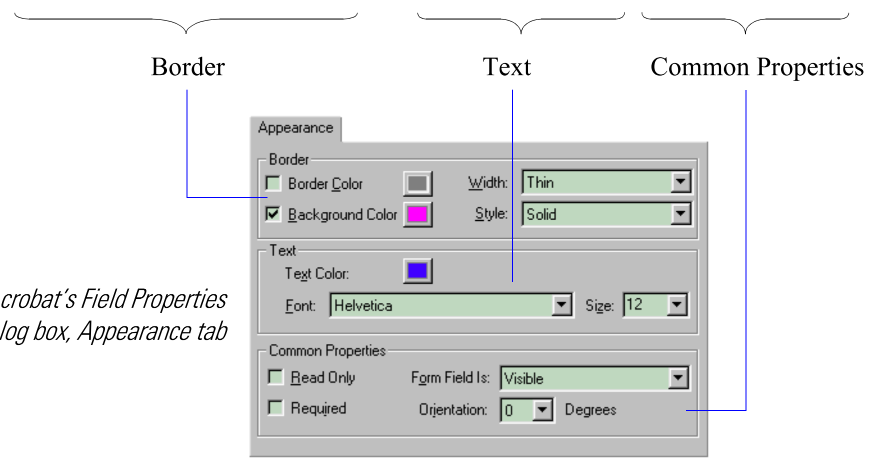
Automatic names are useful when defining general-purpose items, such as buttons or text fields for forms where the field data will not be processed electronically. Do not use an autoname when you need to relate to a value of a specific field (for example, if calculating the sum of two fields).

Note: *Don’t use autonames when button definitions are placed on master pages, as each instance of the button on body pages will have a different name.*

/FieldAppearance

The Field Appearance specification, a forward slash followed by 11 characters, has the following format:

/	c	b	w	s	-	c	f	s	-	s	r
	border color	background color	width	style		color	font	size		state+ read-only	rotation+ required



☐ Border Color ▨ Width: Thin
☑ Background Color 🟪 Style: Solid

Text Color: 🟦
Font: Helvetica Size: 12

☐ Read Only Form Field Is: Visible
☐ Required Orientation: 0 Degrees

Acrobat's Field Properties dialog box, Appearance tab

Different letters/digits indicate properties for the corresponding attribute, as set in the `SP-FormDefs.txt` file (or predefined in Acrobat). Consult the SP Form Assistant: Field Appearance Quick Reference for the default symbols for the various properties.

Fonts

When using fields with fonts not included in Acrobat's Base14 fonts (Times, Helvetica, Courier, Zapf Dingbats, Symbol), **use Acrobat's Save or Save As functions (overwriting the same file) so that the appearance of these fields will be stored in the PDF** and will be correct when the file is displayed in Acrobat Reader.

Acrobat uses the Zapf Dingbats font for check boxes and radio buttons, regardless of the font specified in the Field Appearance specification.

Adding or Modifying Symbols in SP-FormDefs.txt

You can modify the existing positions for Acrobat's base fonts or add new positions for fonts not included in the base fonts as needed, by editing the settings in the `SP-FormDefs.txt` file; you need to know the OpenType/PostScript/TrueType name for the specific font.

You can find the PostScript name by opening the `.pfb` file, searching for the `/FontName` entry. The font name will be listed as follows:

```
/FontName /BulmerMT def
```

`/BulmerMT` is the font name that you need to use in the `SP-FormDefs.txt` file.

Note: *Make sure you distill such files with such fonts embedded, so that the form fields are displayed correctly on all end-user systems. Also, specify that Acrobat/Reader 5 is to be used; field appearance with earlier versions of Acrobat field will show the characters used in form fields using substitution fonts. When using symbol fonts for icons, substitution will show the character as present with standard fonts (e.g. the 👁 symbol in the Webdings font is displayed as the letter* `N`*).*

Sample additional font names in `SP-FormDefs.txt`:

```
(m) /Myriad-Roman               TS_StoreFontName  %
(M) /Myriad-Bold                TS_StoreFontName  %
(v) /Verdana                    TS_StoreFontName  %
(V) /Verdana,Bold               TS_StoreFontName  %
(X) /Webdings                   TS_StoreFontName  %
```

Rotation

The field rotation is applied automatically in Acrobat/Distiller during the conversion to PDF, based on the “perceived” page orientation

To change the rotation setting for a specific field, edit its Appearance string, or label.

If * is specified for the field rotation, the default rotation value, as set in the SP-FormDefs.txt file, is used.

Default rotation can also be defined from within a FrameMaker file through the following hypertext marker:

```
alert /  (a) /TS_DefaultRotation TSset
```

State

Fields (including buttons) with a read-only property are not active.

Using Field Appearance Labels

When the same Field Appearance specifications are used frequently, it may be easier to use labels instead of repeating the specification for each marker. In the beginning of the `SP-FormDefs.txt` file, under Field Appearance Labels, there are several predefined labels (`e.g. TSF0` to `TSF9`), which can be adjusted as needed. This also enables the rapid updating of settings across a project.

When using a label, specify it instead of the Field Appearance specification, without parentheses. For example:

```
alert ~BUN (*) TSF3 (Search TechDoc) (Search) /AcroSrch:Query /p
```

(Tooltip)

Tooltip is the “Short Description” text used when creating form fields in Acrobat. This text is displayed when placing the cursor in the field area and waiting for one or two seconds.

Use tooltips to identify the field's purpose or provide hints as to expected value or data format; for example: `Enter date (mm/dd/yy)`.

(ButtonLabel)

ButtonLabel is the actual label to be placed in the button's area (in the "Up" state), centered horizontally and vertically.

(ButtonLabel-Up)

"Up"-state text label (normal state, when button is displayed)

(ButtonLabel-Rollover)

"Rollover"-state text label (displayed when the cursor enters the button's area)

(ButtonLabel-Down)

"Down"-state text label (displayed when the button is clicked).

/NamedAction

Named actions enable access to many Acrobat/Readers commands. Named actions vary between Acrobat/Reader versions, and include commands such as: /Close /Save /SaveAs /Print /PageSetup /Quit /ImportImage /ImportNotes /AcroForm:ImportFDF /ExportNotes /AcroForm:ExportFDF /GeneralInfo /OpenInfo /FontsInfo /SecurityInfo /GeneralPrefs /NotePrefs /FullScreenPrefs /Weblink:Prefs /AcroSearch:Prefs /AcroSrch:Query /AcroSrch:Indexes /AcroSrch:Results /AcroSrch:Assist /AcroSrch:PrevDoc /AcroSrch:PrevHit /AcroSrch:NextHit /AcroSrch:NextDoc

To find out what named actions are available for your version of Acrobat, create a link, bookmark or form field in a PDF and choose an action of Execute a menu item, then click Add and scroll the list of menu items. Note: some actions available in Acrobat are not available in Reader, while others may require the PDF to have extended rights.

/h

`/h` (single character) defines the ***highlight mode*** of the field when click.

Options are: `/i` (invert), `/n` (none), `/o` (outline) and `/p` (push).

Note: *The "Push" highlight mode is especially visible when the border style is set to Inset or Beveled.*

(/Action...)

Action, enclosed in parentheses, defines the button’s action; this has to be specified with the short form of keys (also known as “filtered”).
For example:

```
(/S /Launch /F (readme.txt) )
(/S /GoToR /F (abc.pdf) )
(/S /URI /URI (mailto:shlomo@microtype.com?Subject=Feedback) )
(/S /URI /URI (http://www.microtype.com) )
(/S /JavaScript /JS (app.alert("hello");)  )
(/S /Named /N /Print /Next << /S /GoToR /F (abc.pdf) >> )
```

/JavaScript

/JavaScript is the name of an Acrobat JavaScript code fragment as defined in the `SP-FormDefs.txt` file.

If the code fragment is not defined, a message will appear in Distiller window (and/or log file), such as:

```
'/TS-SetCCAddressFrom' JavaScript code fragment not defined,
'Copy' button; skipped!
```

This is an example of a JavaScript code fragment (in the `SP-FormDefs.txt` file):

```
% JavaScript functions - example
/TS-ReadOutLoud {
(tts.qText("With the Form Assistant, you can define form fields
in your source FrameMaker files with custom hypertext markers.");
tts.talk();)
} def
```

(JavaScriptCode)

JavaScript code, enclosed in parentheses.

/JavaScript-Up

JavaScript code for the Up “event”.

/JavaScript-Enter

JavaScript code for the Enter “event”.

/JavaScript-Exit

JavaScript code for the Exit “event”.

ImageWidth

Image width (number of pixels).

ImageHeight

Image height (number of pixels).

(JPEGfile)

Root filename of the JPEG images to be used for the button's up/down/over states. Suffixes for each of the states are defined in the `SP-FormDefs.txt` file (by default: _up, _down, _over).

The images should be present in the path defined `in SP-FormDefs.txt` (TSimagePath).

For example, if `(square)` is specified as the JPEGfile, expected files for the different button states are: square_up.jpg, square_down.jpg, square_over.jpg

(URL)

The URL to link to for form processing.

(FieldValue)

Text to be displayed by default in the text field.

Specify an empty string `()` if you wish the field to be initially empty.

/Q

`/Q` (single character, case-insensitive) indicating the ***text alignment***:

- `/L` is left-aligned
- `/C` is centered
- `/R` is right-aligned.

nn

Text limit – the maximum number of characters allowed for this field.

/CustomKeystrokeScript

A Keystroke script (Acrobat Javascript) that is triggered each time the user enters text into the text field.

/CustomValidationScript

A Validation script (Acrobat Javascript) that validates the value each time the user enters text into the text field.

/CustomFormatScript

A Format script (Acrobat Javascript) that is triggered after the user enters text into the text field, changing the way the data value appears.

(NumFieldSettings)

Single letter in parentheses, pointing to one of the pre-defined numeric field settings in the `SP-FormDefs.txt` file (see table below); adjust the pre-defined settings, or add new styles (using new symbols).

Symbol	Currency	Currency symbol	Currency symbol placement[a]	# places after decimal point	Separator style[b]	Negative style[c]
(=)	none	("")	/Pre	3	0	2
($)	Dollar	("$")	/Pre	3	2	0
(E)	Euro	(" ")	/Post	2	0	2
(K)	Krona	(" kr")	/Post	2	3	2
(N)	NIS	("NIS")	/Post	0	0	3
(P)	Pound	("£")	/Pre	2	2	2
(Y)	Yen	("¥")	/Post	2	2	2

a. Currency symbol placement:
/Pre = display the currency symbol before number
/Post = display the currency symbol after number

b. Separator style:
`0` = 1,234.56 `1` = 1234.56 `2` = 1.234,56 `3` = 1234,56

c. Negative Style:
`0` = Minus Black –1,234.01 `1` = Red Number 1,234.01
`2` = Black Parentheses (1,234.01) `3` = Red Parentheses (1,234.01)

/min r /max r

Followed by numbers with a fractional point ("real"), these set the lower and upper limit (add `.0` when specifying numbers with no fractional values, such as `5.0` instead of `5`).

To skip either of the two limits, use a `/skip` keyword for the respective limit, followed by any number (which will be ignored).

CalcOrderGroup

Calculation order groups – can be `1`, `2` or `3`. This is applicable when you have dependencies such as one field calculating the subtotal and then another field calculating the tax on the subtotal.

Fields in group 1 are calculated first, followed by fields in group 2, and then by fields in group 3. (Internal order within each group is according to the page order, and within each page according to the marker creation order.)

Specify 1 for all calculation fields if there are no dependencies.

Note: *To inspect or modify the calculation order in Acrobat, use* Tools > Forms > Set Calculation Order.

/Function

The calculation function to be performed:

- `/Add` – value is the sum of the specified fields
- `/Mul` – value is the product of the specified fields
- `/Min` – value is the smallest of the specified fields
- `/Max` – value is the largest of the specified fields
- `/Avg` – value is the average of the specified fields
- `/JS` – execute the specified JavaScript custom calculations

The function name is not case-sensitive. If the name typed is does not match the unique beginning of the available functions (`/AD`, `/MU`, `/MI`, `/MA`, `/AV` or `/J`), a note indicating the problem will be placed in the corresponding location in the PDF file. A message, indicating the field name and the function specified in the marker, is displayed in the Distiller message window (and also included in the log file).

For more information on JavaScript functions, consult the Acrobat JavaScript manuals, available as PDFs from Adobe's Developer Connection).

If you use certain calculations frequently, or if you are close to the FrameMaker limit of characters in the marker text dialog box, you can store the function in a text file, and reference this file using the `distillP` command. For example:

- In the default TimeSavers folder, the text file named `xyz.js` has the following content:

  ```
  (var f=getField("ValueWarn");
  f.textColor=event.value < 60 ? color.red : color.blue)
  ```

- In the marker, the `xyz.js` file is referred to with `distill` command:

  ```
  alert ~NC (ValueWarn) TSF4 (Numeric, calculated) /JS
  (xyz.js) distillP (=) /c
  ```

- When distilling the file, the Distiller window (and log file) will include the following message:

  ```
  --- importing xyz.js ...
  ... completed
  ```

  Use the `distillQ` command instead of `distill` if you don't want to have this message listed.

Alternatively, the JavaScript functions can be stored in the `SP-FormDefs.txt` file as variables, and then called by their variable name. For example:

- In the `SP-FormDefs.txt` file, the following section is included:

  ```
  % JavaScript functions - example
  /TSJ-VAT {
  (var f=getField("VAT");
  var g=getField("SubTotal");
  f.value=g.value * 0.725) } def
  ```

- In the marker, the `TSJ-VAT` function is referenced:

  ```
  alert ~NC (ValueWarn) TSF4 (Numeric, calculated) /JS TSJ-VAT (=) /c
  ```

(Fields or JS)

With all functions, except `/JS` – this parameter is a list of field names to be used in the calculation, enclosed in parentheses. Field names are enclosed in quotes and separated with commas. For example: `("Qty","Price")`.

With the `/JS` function you can use custom JavaScript calculations. For example: `(var f=getField("Item.Tax");`

```
var g=getField("Item.Add");
f.value=g.value * 0.725)
```

PercentValue

Percentage value. 1 is 100%, so `0.9`, for example, will be interpreted as 90%, and `3.4` as 340%. Number can be preceded with a minus to denote a negative value.

nDec

Number of digits after the decimal point.

SeparatorStyle

`0` = 1,234.56 `1` = 1234.56 `2` = 1.234,56 `3` = 1234,56

DeafultPos

This number indicates the item to be listed as the default. The position is sequential, with the first item being 1 (regardless of the export value).

[Options array]

This item lists all options to be presented in the list, in the following format:

```
[ [(1)(First)] [(2)(Second)] [(3)(Third)] [(4)(Fourth)] [(5)(Fifth)] ]
```

The first value in each item is the Export value; the second value is the item as appears in the list. Export values are used when submitting or exporting the form data (e.g. to be processed by applications).

Several common option arrays are included in the `TS-FormDefs.txt` file:

Predefined options	Purpose
TSF-Date	1 through 31
TSF-Day	Sunday through Saturday
TSF-Month	January through December
TSF-Dept	Department names
TSF-Gender	Male-Female
TSF-Job	Job categories

Predefined options	Purpose
TSF-Marital	Marital Status
TSF-PayType	Payment Type (credit cards)
TSF-ExpMonth	Expiration Month
TSF-Ship	Shipping Methods
TSF-State	States in the USA
TSF-Country	All countries

DateFormat

Specified as a number, one of the options defined in `SP-FormDefs.txt`.

#	Date Format
1	m/d
2	m/d/yy
3	mm/dd/yy
4	mm/yy
5	d-mmm
6	d-mmm-yy
7	dd-mmm-yy

#	Date Format
8	yy-mm-dd
9	mmm-yy
10	mmmm-yy
11	mmm d, yyyy
12	mmmm d, yyyy
13	m/d/yy h:MM tt
14	m/d/yyyy HH:MM

Modifying/Adding Date Formats

You can edit the default date formats defined in the `SP-FormDefs.txt` file, or add your own. The following building blocks can be used:

Code	Meaning	Example
yyyy	Long year	2001
yy	Abbreviate Year	01
mmmm	Long month	August
mmm	Abbreviated month	Aug
mm	Numeric month with leading zero	08
m	Numeric month without leading zero	8
dddd	Long day	Wednesday
ddd	Abbreviated day	Wed
dd	Numeric date with leading zero	03
d	Numeric date without leading zero	3

Code	Meaning	Example
HH	24 hour time with leading zero	07
H	24 hour time without leading zero	7
hh	12 hour time with leading zero	07
h	12 hour time without leading zero	7
MM	minutes with leading zero	03
M	minutes without leading zero	3
ss	seconds with leading zero	04
s	seconds without leading zero	4
tt	am/pm indication	am
t	single digit am/pm indication	a

TimeFormat

The following time formats are predefined.

No.	Time Format	Example
0	24HR:MM	14:30
1	12HR:MM	2:30 PM
2	24HR:MM:SS	14:30:15
3	12HR:MM:SS	2:30:15 PM

(c)

A single character to be used as the check mark. This character is formatted with the Zapf Dingbats font (regardless of the font specified in the Field Appearance string). Following is a listing of dingbats characters:

✁	✂	✃	✄	☎	✆	✇	✈	✉	☛	☞	✌	✍	✎	✏	✐
!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	0
✑	✒	✓	✔	✕	✖	✗	✘	✙	✚	✛	✜	✝	✞	✟	✠
1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	@
✡	✢	✣	✤	✥	✦	✧	★	✩	✪	✫	✬	✭	✮	✯	✰
A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
✱	✲	✳	✴	✵	✶	✷	✸	✹	✺	✻	✼	✽	✾	✿	✟
Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	‘
❁	❂	❃	❄	❅	❆	❇	❈	❉	❊	❋	●	❍	■	❏	❐
a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p
❑	❒	▲	▼	◆	❖	◗	❘	❙	❚	❛	❜	❝	❞		
q	r	s	t	u	v	w	x	y	z	{	|	}	~		

The six Acrobat built-in check mark types are: Check (4), Circle (l), Cross (8), Diamond (u), Square (n) and Star (H); all other dingbats are supported.

/Val

The ***export value*** to be exported for this check box when form data is submitted.

/Y

Check box ***Checked/unchecked*** by default: If /Y or /y, it will be checked by default; /N, /n or other values = unchecked.

(Initial Message)

Text to be displayed by default in the text field; specify an empty string () if you wish the field to be initially empty. For default text to be displayed, the field has to be visible (i.e. not hidden) – as set by the Field Appearance parameter. The initial message (for example: “Part description appears in this area”) will not be displayed again after other text items are displayed in this area (when different buttons are entered).

NumLines

If a value of 1 is specified, the text field created will be a single-line field. Any other value will result in a multi-line field.

Defines a multiline text field. The field may contain multiple lines, which may not display entirely if the field is not high/wide enough. No scrolling bars are added, but the insertion point can be placed inside the field and moved with the arrows to scroll the text.

Note: *The rollover text display field – single-line or multi-line alike – has fixed dimensions and will not display the entire text if not large enough.*

(Message)

A text string enclosed in parentheses. This message is displayed in the display box when the field is entered, using the properties defined by the ~DispBox shortcut.

If you need to display the same set of messages frequently, you can store these in the `SP-FormDefs.txt` file (under `% Rollover Messages`), and then refer to the message by its number, with the `&` (ampersand) prefix, e.g. alert ~nBox (&3),alert ~nBox (&25). The default maximum for stored messages is 512; this value can be changed in the `SP-FormAsst.ini` file.

Note: *Another way to refer to messages is by storing the message in a text file in the Distillr folder, and refer to it using the `distill` command. For example:* `~nBox (xyz.txt) distill`

/ChangeProperties

Starting with a slash, followed with two groups of letters, separated by a hyphen. First group of four letters defines changes in the activating field; second group defines changes in the display box. Changes take place when entering the activation field, and the previous state is restored when exiting.

	Activating Field					*Display Box*			
/	**c**	**b**	**w**	**t**	**–**	**c**	**b**	**t**	**s**
	border color	background color	width	text color		border color	background color	text color	text size

Consult the SP Form Assistant: Field Appearance Quick Reference for the default symbols for the various properties (as defined in `SP-FormDefs.txt`).

Use the `*` (asterisk) to specify the default setting for the corresponding property, as defined in the activating field or display box.

For example, /d*1*-**** defines a change in border color and width of the activating field when it is entered, but no change in the display box properties. /d*1*-b2m* will change the display box, in addition, so that it has different colors for border, background and text, when the specific activation field is entered. The border display properties are specific for each activating field, and must be repeated in each shortcut, or changed as required when you wish different fields to cause different display behavior.

(PopUpAction)

Action specification, enclosed in parentheses. For example:
`(/S/Named/N/Print)`

If the Action specification does not start with a slash – such as `(xyz)` – the value is interpreted as a named destination to jump to. It is recommended to define such destinations using the `newlink TMS xyz` marker (with TimeSavers/ConvertFMnewlink enabled), so that `xyz` does not have the variable prefix it has when using `newlink xyz`.

A value of `(none)` or `(None)` will result in no action.

/cw

`/cw` (two characters) define a ***highlighting effect applied to the activating field*** when entered. First letter indicates the border's color; second letter indicates the line width; `*` (asterisk) specifies no change for that property.

Field Marker Reference

Buttons

~BUN

Defines a button with an Acrobat named action.

Syntax

```
alert ~BUN (FieldName) /FieldAppearance (Tooltip) (ButtonLabel) /NamedAction /h
```

Example

```
alert ~BUN (Print) /rg1s-DD5-v* (Print document) (PRINT) /Print /p
```

~BU

Defines a button, user-specified action(s).

Syntax

```
alert ~BU (FieldName) /FieldAppearance (Tooltip) (ButtonLabel) (/Action…) /h
```

Example

```
alert ~BU (Web) /g^2b-KI6-nr (Link to Website) (Website)
(/S /URI /URI (http://www.microtype.com) )  /p
```

~BUL

Defines a button, user-specified action(s), with three labels for the different button states:
• ***Up*** (normal state, when button is displayed)
• ***Rollover*** (when the cursor enters the button's area)
• ***Down*** (when the button is clicked)

Syntax

```
alert ~BUL (FieldName) /FieldAppearance (Tooltip) (ButtonLabel-Up)
(ButtonLabel-Rollover) (ButtonLabel-Down) (/Action…) /h
```

Example

```
alert ~BUL (*) /cAms-4+c-v* (Open readme.txt) (HELP) (ABC) (XYZ) (/S /Launch /F
(readme.txt)) /I
```

~BUIMG

Defines an image button, with Acrobat named action.

Syntax

```
alert ~BUIMG (FieldName) (Tooltip) /NamedAction ImageWidth ImageHeight (JPEGfile)
```

Example

```
alert ~BUIMG (SRCH) (Search) /AcroSrch:Query 50 44 (square)
```

~BUJS

Defines a button with a JavaScript action (mouse up).

Syntax

```
alert ~BUJS (FieldName) /FieldAppearance (Tooltip) (ButtonLabel) /JavaScript /h
```

Example

```
alert ~BUJS (JSexample) /KL1b-KH2-na (Text-to-Speech example) (Say It!) /TS-ReadOutLoud
/p
```

~BUJS-UEX

Defines a button with mouse ***Up***, ***Enter*** and ***Exit*** JavaScript actions.

Syntax

```
alert ~BUJS-UEX (FieldName) /FieldAppearance (Tooltip) (ButtonLabel) /JavaScript-Up
/JavaScript-Enter /JavaScript-Exit /h
```

Example

```
alert ~BUJS-UEX (JSexample3) /KL1b-KH2-na (Up-Enter-Exit examples) (Try It!)
/TS-SampleAlert1 /TS-SampleAlert2 /p
```

~BUJS-EX

Defines a button with mouse ***Enter*** and ***Exit*** JavaScript actions.

Syntax

```
alert ~BUJS-EX (FieldName) /FieldAppearance (Tooltip) (ButtonLabel) /JavaScript-Enter
/JavaScript-Exit /h
```

Example

```
alert ~BUJS-EX (JSexample3) /KL1b-KH2-na (Enter-Exit examples) (Try It!)
/TS-SampleAlert1 /TS-SampleAlert2 /p
```

~BUShowHide

Defines a button with a Show/Hide field action (toggle), to hide or show one or more specified buttons. The label of the button is automatically set to – (when clicking it hides fields) or + (when clicking it shows fields). This field can be used in forms or presentations where many fields are present (e.g. navigational buttons) and we want to be able to temporarily hide some of these, to be shown on demand, as needed.

Syntax

```
alert ~BUShowHide (FieldName) /FieldAppearance (Tooltip) (ButtonLabel)
("Field1","Field2",...) /h
```

Example

```
alert ~BU-ShowHide (ShowHide) /d^1s-pS8-n* (show/hide navigation buttons) (-)
("top","next","prev","back","srch","full","close") /o
```

~BUReset

Defines a button with a Reset action.

Syntax

```
alert ~BUReset (FieldName) /FieldAppearance (Tooltip) (ButtonLabel) /h
```

Example

```
alert ~BUReset (Reset) /cAms-4+c-vr (Reset form data) (Reset Form) /i
```

~BUSubmit

Defines a button with a Submit action (export format is FDF, with data consisting of field data only).

Syntax

```
alert ~BUSubmit (FieldName) /FieldAppearance (Tooltip) (ButtonLabel) (URL) /h
```

Example

```
alert ~BUSubmit (Submit) /cAms-4Ic-vr (Submit form data) (Submit)
(http://FormServer.com/cgi-bin/myscript#FDF) /o
```

~BUSubmitHTML

Defines a button with a Submit action (HTML format).

Syntax

```
alert ~BUSubmitHTML (FieldName) /FieldAppearance (Tooltip) (ButtonLabel) (URL) /h
```

Example

```
alert ~BUSubmitHTML (Submit) /cAms-4Ic-v* (Submit form data) (Submit)
(http://www.forms.com) /o
```

~BUSubmitXML

Defines a button with a Submit action (XFDF format , with data consisting of field data only).

Syntax

```
alert ~BUSubmitXML (FieldName) /FieldAppearance (Tooltip) (ButtonLabel) (URL) /h
```

Example

```
alert ~BUSubmitXML (Submit) /cAms-4Ic-v* (Submit form data) (Submit)
(http://www.forms.com) /o
```

~BUSubmitPDF

Defines a button with a Submit action (complete PDF document).

Syntax

```
alert ~BUSubmitPDF (FieldName) /FieldAppearance (Tooltip) (ButtonLabel) (URL) /h
```

Example

```
alert ~BUSubmitPDF (Submit) /KL1b-KH2-na (Submit form by e-mail; please SAVE before
submitting) (By e-mail) (mailto:timesavers@microtype.com) /p
```

~BUPHOTO

Defines a button that lets end users ***with Adobe Reader XI or later*** to insert a PDF photo into a form.

Syntax

```
alert ~BUPHOTO (FieldName) /FieldAppearance (Tooltip)
```

Example

```
alert ~BUPHOTO (*) /Ng1s-^^^-va (Click and select a PDF file with your image, Adobe Reader
XI or later)
```

Text / Numeric Fields

~TFM

Defines a multiline text field. The field may contain multiple lines, which may not display entirely if the field is not high/wide enough. No scrolling bars are added, but the insertion point can be placed inside the field and moved with the arrows to scroll the text.

Syntax

```
alert ~TFM (FieldName) /FieldAppearance (Tooltip) (FieldValue) /Q
```

Example

```
alert ~TFM (TextFieldMulti) /cAms-4+c-vr (Multiline text field) (Initial value) /c
```

~TFS

Defines a single-line text field. The text may not display entirely if the field is not wide enough. No scrolling bars are added, but the insertion point can be placed inside the field and moved with the arrows to scroll the text horizontally.

Syntax

```
alert ~TFS (FieldName) /FieldAppearance (Tooltip) (FieldValue) /Q
```

Example

```
alert ~TFS (TextFieldSingle) /cAms-4+c-vr (General-purpose text field) (value) /L
```

~TFSX

Defines a single-line text field, with a limit on the number of characters that can be inserted. The text may not display entirely if the field is not wide enough. No scrolling bars are added, but the insertion point can be placed inside the field and moved with the arrows to scroll the text horizontally.

Syntax

```
alert ~TFSX (FieldName) /FieldAppearance (Tooltip) (FieldValue) /Q nn
```

Example

```
alert ~TFSX (StateCode) /cAms-4+c-vr (Text field, with a maximum field length) (value)
/L 2
```

~TFComb

Defines a single-line text field, with preset spacing according to the value provided (number of characters).

Syntax

```
alert ~TFComb (FieldName) /FieldAppearance (Tooltip) (FieldValue) /Q nn
```

Example

```
alert ~TFComb (ID) /cAms-4+c-vr ("Comb" Text field) () /L 2
```

~TFSQ

Defines a text field with a ***keystroke script***, a ***validation script*** and a ***format script***.

If you would like to define a field with just two or one of the scripts, indicate `/skip` instead of the corresponding items (see examples).

Syntax

```
alert ~TFSQ (FieldName) /FieldAppearance (Tooltip) (FieldValue) /Q nn
/CustomKeystrokeScript /CustomValidationScript /CustomFormatScript
```

Examples

```
alert ~TFSQ (email) TSFa (E-mail address) () /L 0 /RegExp-kEmail /skip /skip
alert ~TFSQ (from.email) TSFA (E-mail Address) () /L /RegExp-kEmail /RegExp-vEmail /skip
alert ~TFSQ (PhoneNum) TSFa (Phone Number) () /L 10 /skip /skip /TS-Phone-Format
```

~N

Defines a numeric field, formatted with currency and separators.

Syntax

```
alert ~N (FieldName) /FieldAppearance (Tooltip) (FieldValue) (NumFieldSettings) /Q
```

Example

```
alert ~N (NumField) TSF4 (Numeric field) (12) (E) /L
```

~NV

Defines a numeric field, formatted with currency and separators; lower and upper limits are set for the value entered.

Syntax

```
alert ~NV (FieldName) /FieldAppearance (Tooltip) (FieldValue) (NumFieldSettings) /min  r
/max  r /Q
```

Examples

```
alert ~NV (*) TSF4 (Numeric field, validated) (12) ($) /min 7.0 /max 23.0 /R
alert ~NV (*) TSF4 (Numeric field, validated) (12) ($) /skip 0 /max 23.0 /L
```

~NC

Defines a numeric field (formatted with currency and separators) whose value is the result of a calculation.

Important: Calculation fields must be uniquely named (i.e. do not use the `*` as an autoname for such fields).

Syntax

```
alert ~NC (FieldName) /FieldAppearance (Tooltip) CalcOrderGroup /Function (Fields or JS)
(NumFieldSettings) /Q
```

Example

```
alert ~NC (ValueWarn) TSF4 (Numeric, change color if below 60) /JS
(var f=getField("ValueWarn"); f.textColor=event.value < 60 ? color.red : color.blue) (=)
/c
```

~P

Defines a Percent-formatted field.

Syntax

```
alert ~P (FieldName) /FieldAppearance (Tooltip) PercentValue nDec SeparatorStyle
```

Example

```
alert ~P (Percent) TSF5  (Percent field) 3.3  2  4
```

~PV

Defines a Percent-formatted field; lower and upper limits are set for the value entered.

Syntax

```
alert ~PV (FieldName) /FieldAppearance (Tooltip) PercentValue nDec SeparatorStyle
/min  r /max  r
```

Example

```
alert ~PV (PercentV) TSF3 (Percent, validated) 3.7  3  4 /min 2.2 /max 8.7
```

~DATE

Defines a Date field.

Syntax

```
alert ~DATE (FieldName) /FieldAppearance (Tooltip) (FieldValue) DateFormat /Q
```

Example

```
alert ~DATE (Date) TSF3 (Enter date: mm/dd/yy) () 3 /L
```

~TIME

Defines a Time field.

Syntax

```
alert ~TIME (FieldName) /FieldAppearance (Tooltip) (FieldValue) TimeFormat
```

Example

```
alert ~TIME (Time) TSF5 (Enter Time: hh:mm) () 1
```

~ZIP

Defines a Zip field, which can have 5 digits (no other characters allowed).

Syntax

```
alert ~ZIP (FieldName) /FieldAppearance (Tooltip) (FieldValue)
```

Example

```
alert ~ZIP (Zip) TSF9 (Zip field) ()
```

~ZIP4

Defines a Zip+4 field, which can have 5 digits, a hyphen, and then 4 more digits (no other characters allowed). When entering this field, the hyphen will be inserted automatically if entering 9 digits (or it can be added manually after the first 5 digits).

Syntax

```
alert ~ZIP4 (FieldName) /FieldAppearance (Tooltip) (FieldValue)
```

Example

```
alert ~ZIP4 (Zip4) TSF9 (Zip+4 field) ()
```

~PHONE

Defines a Phone field, formatted as `(nnn) nnn-nnnn`. Parentheses and hyphen will be added automatically if not included when entering the field's data; only numerals allowed.

Syntax

```
alert ~PHONE (FieldName) /FieldAppearance (Tooltip) (FieldValue)
```

Example

```
alert ~PHONE (Phone) TSF7 (Phone) ()
```

~SSN

Defines a field formatted as `nnn-nn-nnnn`. Hyphens will be added automatically if not included when entering the field's data; no other characters allowed.

Syntax

```
alert ~SSN (FieldName) /FieldAppearance (Tooltip) (FieldValue)
```

Lists

List boxes enable the selection of a single item from a displayed list of items. List boxes are useful for long lists; if the number of items in the list extends beyond the vertical boundary of the field, a scrolling bar is displayed.

Combo boxes (also called drop-down buttons) allow the selection of a single item from a drop-down list, requiring less area in the page than a list box.

~LIST

Defines a list box. If the list box is not large enough to display all items, a vertical scrolling bar appears when the field is clicked.

Syntax

```
alert ~LIST (FieldName) /FieldAppearance (Tooltip) DeafultPos [Options array]
```

Examples

```
alert ~LIST  (List) TSF9 (Sample List) 3 [ [(1)(First)] [(2)(Second)] [(3)(Third)]
[(4)(Fourth)] [(5)(Fifth)]]
```

~LISTQ

Defines a list box with additional scripts (e.g. Selection Change; use `/skip` instead of the one you do not need). If the list box is not large enough to display all items, a vertical scrolling bar appears when the field is clicked.

Syntax

```
alert ~LISTQ (FieldName) /FieldAppearance (Tooltip) DeafultPos [Options array]
/CustomKeystrokeScript /CustomValidationScript /CustomFormatScript
```

Example

```
alert ~LISTQ (DeptWithScript) TSF8 (List of Departments) 7 TSF-Dept /TS-SampleAlert1
/skip /skip
```

~DD

Defines a combo box (drop down list).

Syntax

```
alert ~DD (FieldName) /FieldAppearance (Tooltip) DeafultPos [Options array]
```

Examples

```
alert ~DD (DD) TSF8 (sample drop-down) 4
[ [(1)(First)] [(2)(Second)] [(3)(Third)] [(4)(Fourth)] [(5)(Fifth)]]

alert ~DD (DD-ExpMon) TSF8 (sample drop-down) 4 TSF-ExpMonth
```

~DDE

Defines an editable combo box (drop down list).

Syntax

```
alert ~DDE (FieldName) /FieldAppearance (Tooltip) DeafultPos [Options array]
```

Example

```
alert ~DDE (DDE) TSF8 (sample drop-down) 4
[ [(1)(First)] [(2)(Second)] [(3)(Third)] [(4)(Fourth)] [(5)(Fifth)]]
```

~DDQ

Defines an editable combo box (drop down list), with a custom validation script.

Syntax

```
alert ~DDQ (FieldName) /FieldAppearance (Tooltip) DeafultPos [Options array]
/CustomKeystrokeScript /CustomValidationScript /CustomFormatScript
```

Example

```
alert ~DDQ (DDQ) TSF8 (sample drop-down) 4
[ [(1)(First)] [(2)(Second)] [(3)(Third)] [(4)(Fourth)] [(5)(Fifth)]] /UpdateValue /skip
/skip
```

Check Boxes & Radio Buttons

Check boxes are generally used to indicate a Yes or No response in forms, or to indicate the selection of one or more items from a given list. As check boxes have no text associated with them, you must add text near the box to indicate the context.

Radio buttons are useful when the user must choose one out of a few options. Radio buttons can be placed near icons and other visual objects.

~CB

Defines a check box.

Syntax

```
alert ~CB (FieldName) /FieldAppearance (Tooltip) (c) /Val /Y
```

Example

```
alert ~CB (CheckBox1) TSF6 (Check Box) (4) /OptA /N
```

~CBA

Defines a check box with a JavaScript action.

Syntax

```
alert ~CBA (FieldName) /FieldAppearance (Tooltip) (c) /Val /Y (JavaScriptCode)
```

Example

```
alert ~CBA (CheckBox1) TSF6 (Check Box) (4) /OptA /N (app.alert("Option selected is
processed"))
```

~CBQ

Defines a check box with keystroke and validation scripts (use `/skip` instead of the one you do not need).

Syntax

```
alert ~CBQ (FieldName) /FieldAppearance (Tooltip) (c) /Val /Y /CustomKeystrokeScript
/CustomValidationScript
```

Example

```
alert ~CBQ (CheckBoxQ) TSF6 (Check Box) (u) /On /Y /TS-SampleAlert1 /TS-SampleAlert2
```

~RB

Defines a radio button.

Example

```
alert ~RB (RadioButton2) TSF5 (Option B) (l) /OptC
```

Syntax

```
alert ~RB (FieldName) /FieldAppearance (Tooltip) (c) /Val
```

~RBA

Defines a radio button with a JavaScript action.

Syntax

```
alert ~RBA (FieldName) /FieldAppearance (Tooltip) (c) /Val (JavaScriptCode)
```

Example

```
alert ~RBA (RadioButton1) TSF5 (Option C) (l) /OptC (app.alert("Thanks for choosing
option C"))
```

~RBQ

Defines a radio button with keystroke and validation scripts (use `/skip` instead of the one you do not need).

Syntax

```
alert ~RBQ (FieldName) /FieldAppearance (Tooltip) (c) /Val /CustomKeystrokeScript
/CustomValidationScript
```

Example

```
alert ~RBQ (RadioButton1) TSF5 (Option C) (l) /OptC /TS-SampleAlert1 /TS-SampleAlert2
```

~SIG

Defines a digital signature field.

Syntax

```
alert ~Sig (FieldName) /FieldAppearance (Tooltip)
```

Example

```
alert ~Sig (Signature) TSFA (Digital signature)
```

~SIGQ

Defines a digital signature field with JavaScript code executed when "signed".

Syntax

```
alert ~SigQ (FieldName) /FieldAppearance (Tooltip) /JavaScript
```

Syntax

```
alert ~SigQ (Signature2) TSFA (Digital signature) /TS-SampleAlert1
```

Rollovers and Popups

When your PDF includes buttons, you may want to have an explanation of the specific function of each button (like a tool tip) – in addition to what is conveyed directly by the icon and/or label. This text can be displayed as a pop-up only when the button area is entered, so as not to clutter the screen. This can be achieved using form fields, where buttons have "mouse enter" and "mouse exit" actions to display or hide the message/field, and optionally a "mouse down" action, which takes place when the button is clicked.

Fields with these actions can be created easily using the `nBox`, `xBox` or `xBoxL` (rollovers) and `xPop` (variable popup) shortcuts, included with the Form Assistant.

Rollover text display

When using rollovers, additional information is displayed in a fixed location – shown when entering the activating field area and hidden when exiting or clicking. `~DispBox` defines the location/properties of the rollover text display; `~nBox`, `~xBox` and `~xBoxL` define activating fields with different properties.

~DispBox

Draw a text frame with the required dimensions and insert a hypertext marker with the `~DispBox` shortcut. Make sure that the marker is the only item in the text frame, with no other characters (not even spaces).

If all activating fields are present in a specific page, draw the text frame on that body page (the text frame can be included in an anchored frame). You can repeat this on more than one page in the file, as long as both the activating fields and display boxes are on the same page. When activating fields are to be placed throughout the document, the text frame defining the rollover text display should be placed on master page(s).

Note: *An activating field on a page where no text display frame is present (on either a master page or a body page) will result in a JavaScript error when the field is entered.*

Syntax

```
alert ~DispBox /FieldAppearance (Initial Message) NumLines /Q
```

Example

```
alert ~DispBox /^^^^-Dv7-v* ((move cursor over symbols)) 2 /L
```

~nBox

The `~nBox` shortcut is used to define an activating field – displaying a message when the cursor enters the field's area and cleared when cursor is moved out of the activation area. The activating field has no visual properties (e.g. border/fill colors).

Syntax

```
alert ~nBox (Message)
```

Example

```
alert ~nBox (Decision: operation determining which of a number of alternative routes are
to be taken)
```

~xBox

The `~xBox` shortcut defines an activating field with additional properties. In addition to displaying a message, you can define an action that will take place when the field is clicked, change in visual border properties of the activating field, or change visual properties of the display box (border and background color, text size and color).

Syntax

```
alert ~xBox (Message) /ChangeProperties (PopUpAction) /h
```

Example

```
alert ~xBox (Brace Collar \205 RS-88733-40) /d*1*-b2m1 (collar) /o
```

~xBoxL

In addition to all possibilities you can define with `~xBox, ~xBoxL` enables you to constantly display text in the activating field (through its label) and change the visual properties of the activating field or the display box.

Syntax

```
alert ~xBoxL (FieldName) /FieldAppearance (ButtonLabel) (Message) /ChangeProperties
(PopUpAction) /h
```

Example

```
alert ~xBoxL (SRCH) /dj2b-pXy-n* (N) (Acrobat Cross-Document Search) /*h*D-**** (/S
/Named /N /AcroSrch:Query) /p
```

Variable Popups

~xPop

When the mouse enters the activating field, a pop-up is displayed to the left or right of the activating field. Popup size and dimensions are calculated automatically for each popup, based on content, font and size.

Syntax

```
alert ~xPop /FieldAppearance (Message) /cw (PopUpAction)
```

Examples

```
alert ~xPop TSPOP (Proinde cum venabere, licebit, auctore me) (none)
alert ~xPop TSPOP (&34) (Figure3)
```

Controlling Variable Popup Properties

You can control various settings for variable popups through the `SP-FormDefs.txt` file, under `% xPop Setup`.

- `/TS_AltJustify` controls whether popups displayed left of activating area are right-aligned.
- `/MaxWidth` (initially set to 3 inches) is the maximum allowed width for the form field. The form field width will, in most cases, be smaller than this maximum, depending on the specific contents and how it is split between lines.
- Increase the `/IncTextFactor` (initially set to 1.0) if you have a situation where the automatically computed width for the form field is smaller than the required width (and therefore some text is truncated). This can happen because the computed width reflects the total string width without taking into account how the text is split into lines. If your text includes longer words that fall at the end of a line and are moved to the next line because they cannot fit entirely on the current line, this results in a “wasted” white space which was not accounted for. The `/IncTextFactor` provides a mechanism to compensate for this.
- `/XMID` defines the default horizontal center point in the page, which determines whether the variable popup will be displayed to the left or right of the activating field. This point can be redefined in each FrameMaker file as necessary (either as a numeric value, or automatically – by finding the middle point in the paragraph active area).
- `/HGAP` defines the horizontal gap between the button and the text field.
- `/VOFFSET` defines the vertical offset of text field relative (“up”) to the activating button; use a negative value for “down” offset.
- `/BTEXPAND` defines the button area expansion on all sides.

To define XMID automatically, insert a marker in a paragraph on the first page which doesn’t have other hypertext markers, and/or character attributes. The marker text should be as follows:

```
alert / pop TMS_Dict
begin /XMID XUR XLL
sub 2 idiv XLL add
def end
```