Adobe

Note: The information found in this document is taken from the Adobe Designer Help.

Calculations and Scripts

Adobe Designer

Version 6.0

Adobe® Designer Documentation for Microsoft® Windows®
May 2004

As of April 12, 2002, Accelio Corporation (formerly JetForm Corporation) was purchased by Adobe Systems Incorporated. As of that date, any reference to JetForm or Accelio shall be deemed to refer to Adobe Systems Incorporated.

Adobe Systems Incorporated, 345 Park Avenue, San Jose, California 95110, USA.

Contents

34 Properties (Continued)

1 About calculations and scripts

As part of the form design process, a form developer can make use of calculations and scripts to provide a richer user experience. You can add calculations and scripts to most form fields and objects. For example, you can create simple calculations to dynamically update values on an interactive form in response to user input. At a more advanced level, you could create your own functions tailored towards your own custom form processing needs.

Designer supports two scripting languages, each geared towards the needs of a particular type of form developer. FormCalc is a straightforward, easy-to-use calculation language that is modelled on common spreadsheet functionality. It includes a variety of built-in functions designed to reduce the amount of time you need to spend developing your form design. JavaScript, the powerful scripting language, provides you with a great deal of flexibility when creating your scripts, and allows you to leverage any existing knowledge of the language.

Remember that scripting on a form is entirely optional. You can choose to take advantage of scripting to provide a richer user experience, but many of the most powerful features available during form creation are available in Designer without the use of scripts. However, through scripting you can manipulate and control almost all aspects of your form design.

See also

About FormCalc

About JavaScript

About the Object Reference

2 Objects that support calculations and scripts

You can create calculations and scripts that impact all form design objects. However, you can only add calculations and scripts to the form events of a particular object. Not all form design objects support events, and so to modify them you must add the calculation or script to a supported event of another form design object. The following table provides a quick reference of scripting support for objects available in the Standard tab of the Library palette.

Objects that support events	Objects that do not support events
Barcodes	Circle
Button	Content Area
Check Box	Line
Date/Time Field	Rectangle
Drop-Down List	Static Image
Image Field	Static Text
List Box	
Numeric Field	
Password Field	
Radio Button	
Signature Field	
Subform	
Text Field	

See also

About the Object Reference

3 Associating a script with an event

Events are key to the role that both calculations and scripts play in form design. An event is a change of state in a form during either the Form Server rendering process or at run time. When the change of state occurs, the form automatically runs any calculations or scripts associated with the event. Through this process, a form can contain sophisticated logic that transforms the data, the presentation of the data, or even the form itself, in response to circumstances. For example, you could create various calculations to update the total cost on a purchase order form in response to the quantity and type of items purchased.

It is important to carefully consider the event to which you will add a calculation or script. Depending on the type of form you are creating, certain events may never occur, and so any calculations or scripts associated with those events will not run. For example, interactive forms do not require server-based processing (such as when you use Form Server), so forms of this type do not use the layout:ready event, since there is no layout process to react to. Similarly, non-interactive forms do not make use of interactive features, such as buttons, and so do not use events such as click, mouseDown, and mouseUp.

Understanding when events occur, and which events apply to the type of form design you are working on, will greatly reduce the amount of time needed to create your form calculations and scripts.

See also

About the Script Editor

About events

Associating a script with a client or server application

4 Associating a script with a client or server application

For each calculation and script created in Designer, you must specify the location where you want the calculation or script to run. Unless you are using server-based processing such as Form Server, you should ensure that all of your calculations and scripts are set to run on the client application (for example, on Acrobat or a web browser). If you are using server-based processing, you can choose between running calculations on the client application, or running them on the server. By choosing to have calculations and scripts run on the server, you are choosing to run the scripts at a specific point during the form-rendering process.

If you choose Client And Server from the Run At list, then your calculation or script is available to both client and server-based applications. This option is useful, for example, if you don't know whether your end-users will have client or server applications when they attempt to use your form. It is also useful if you want certain form objects to behave one way to a client application and another to a server-based application.

See also

About the Script Editor

About events

Associating a script with an event

5 Comparing FormCalc and JavaScript

FormCalc and JavaScript each present their own set of advantages that you should be aware of prior to writing any scripts on your form. FormCalc is a calculation language that includes a wide range of built-in functions to simplify the most common form functionality. JavaScript is a more powerful and diverse scripting language intended to give you more flexibility and also leverage your existing scripting knowledge.

The following table highlights some of the key differences between FormCalc and JavaScript.

FormCalc	JavaScript
Native Adobe calculation language valid in Designer and Form ServerAdobe document.	Standard scripting language used in many popular software applications
Shorter scripts (typically one line only)	Potential for longer scripts if necessary with the ability to use looping
Contains a variety of useful built-in functions to reduce the amount of scripting required to accomplish common form design tasks	Provides access to the Acrobat Object Model and the JavaScript capabilities of Acrobat
Support for international dates, times, currencies, and number formats	Debugging possible using Acrobat JavaScript debugger
Built-in URL functions for Post, Put, and Get allow web-based interactions	Create custom functions for your own specific needs.
Compatible on all Designer and Form ServerAdobe document supported platforms.	Compatible on all Designer and Form ServerAdobe document supported platforms.

See also

About FormCalc

About JavaScript

Differences between FormCalc and JavaScript functions

6 Using the Script Editor

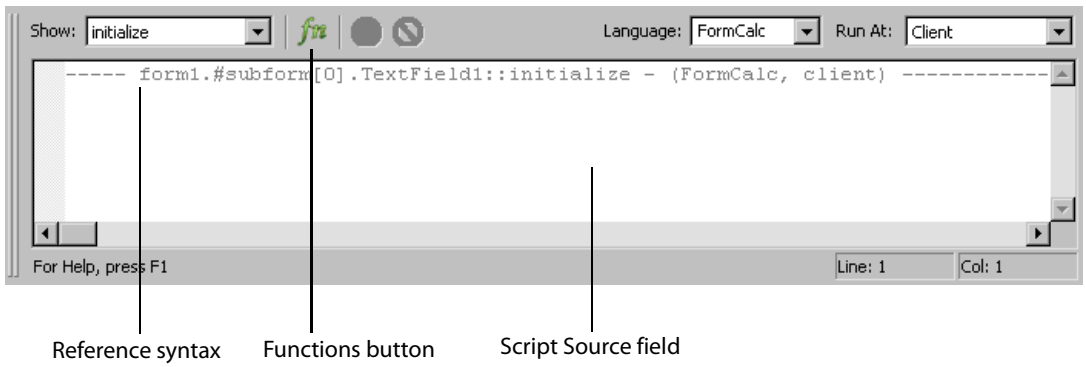
About the Script Editor

The Script Editor within Designer is where you create, modify, and view the calculations and scripts of a particular form.

The editor itself has both a single-line view and a multiline view which you can freely switch between depending on your current needs. Single-line view is designed to maximize the amount of space dedicated to the Layout Editor and other palettes, while multiline view is designed to maximize the amount of space for writing script.

Single-line View

Multiline View



Icon	Command	Description
	Show	Lists all form design events that allow user-defined scripting. Any events that do not apply to a particular object appear dimmed. Events that contain a calculation or script display with an * (asterisk) beside the name of the event.
	Functions	Displays a list of available built-in functions. To place a function onto your script editing field, select a function from the list and press Enter.
	Enter Script Source Changes	Saves your script to your form design. Click this button after you have finished writing your script. If you do not save your script in this manner, your script may not execute correctly or may be lost.

Icon	Command	Description
	Cancel Script Source Changes	Undoes any changes you have made to a particular calculation or script. This applies only to the current editing session and does not act as a multiple undo feature.
	Language	Specifies which scripting language you want to use for the current calculation or script. There are two options: • FormCalc is a native Adobe calculation language ideal for shorter scripts. • JavaScript is a powerful and flexible scripting language for more complex scripts.
	Run At	Specifies where the calculation or script will execute. There are three options: • Client calculations and scripts execute while the client application (for example, Acrobat or web browser) processes the form. • Server calculations and scripts execute while the server application (for example, Form Server) processes the form. • Client and server calculations and scripts execute while being processed by either the client or server application depending on which application process the form.

➤ To show the script editor:

- Select Window > Script Editor.

➤ To change from single-line to multiline view:

- Drag the Script Editor palette bar until the palette is the required size.

Note: Multiline view adds the All Events and Events with Scripts options to the Show list. The All Events option displays all of the events for a particular form design object, even if the events do not contain any calculations or scripts. The Events with Scripts option displays only those events of a particular object that contain calculations or scripts.

➤ To set the default scripting language

1. Select File > Form Properties.
2. Click the Defaults tab.
3. Select your default scripting language from the Default Language list.

➤ To set the default processing application

1. Select File > Form Properties.
2. Click the Defaults tab.
3. Select your default processing application from the Default Run At list.

See also

Attaching a calculation or script to a form design object

Associating a script with a client or server application

About events

Attaching a calculation or script to a form design object

You can attach calculations and scripts to most fields and objects in Designer. Each calculation and script in Designer corresponds to a specific form event. The calculation or script associated with the event runs each time the form event occurs.

➤ **To attach a calculation or script to a form design object:**

1. In the Script Editor, from the Show list, select one of the events that apply to the object. The event you choose specifies when the script will execute. If you are writing a calculation or script that impacts an object that does not support events, you must add your calculation or script to a form design object that supports form events.
2. From the Language list, select your scripting language.
3. From the Run At list, select where you would like the script to execute.

   You can choose to run calculations or scripts on your client-based application (for example Acrobat or web browser) or on your server-based process (for example, Form Server). When set to client, processing of calculations and scripts occurs after the form renders. When set to server, processing of calculations and scripts occurs during the form rendering process.

   Note: Selecting Client And Server from the Run At list causes a script to execute in either the client or server application depending on which application is used to process the form.
4. In the Script Source field, insert your FormCalc calculation or JavaScript script.
5. Click Enter Script Source Changes  to add the script to your form.

See also

About FormCalc

About JavaScript

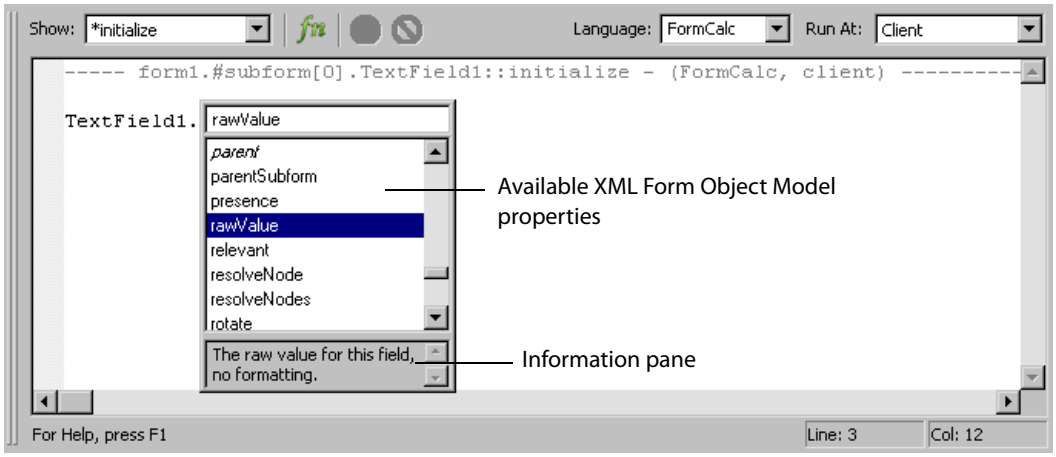
About events

Using object assist to create calculations and scripts

The object assist functionality within the Script Editor allows you to build your calculations and scripts interactively. When writing a calculation or script, each time you enter a "." (period) immediately following a form object or property name, the object assist functionality displays a list of available methods and properties.

➤ **To use object assist to create a calculation or script:**

1. Type the name of a form design object, property, or a valid accessor followed by a "." (period).



2. Select the method or property you want to apply for form design object and continue writing the script. To close the object assist list without selecting a function, press the Esc key.

   The list of available XML Form Object Model properties changes depending on the form design object or property that appears before the "." (period).

See also

About the Object Reference

About accessors

7 Events

About events

Understanding events is key to understanding the role that calculations and scripts play in form design. Every calculation or script is associated with a specific event. An event is defined as a particular change of state in a form. When the particular change of state occurs, the script associated with the event is automatically invoked. By applying calculations and scripts to specific events, you can create sophisticated logic that transforms the data, the presentation of data, or even the form design itself, in response to circumstances.

The object whose change of state triggers the event is called a target. The five general classes of events are distinguished by the type of target. Some events in different classes share the same name because they are similar in function. However, they are distinct events because an event is distinguished by both name and target. In addition, calculations and validations are very much like events and can be treated as special types of events.

See also

Associating a script with an event

Using FormCalc

Using JavaScript

Types of events

Application oriented

Application oriented events are triggered by actions of either the client or server application. Because application events are not directly linked to user actions, they apply to every type of form.

These application events are available from the Show list in the Script Editor:

- docClose
- docReady
- postPrint
- postSave
- prePrint
- preSave

Calculation

Either the client or server application triggers the calculation script when your form design and data merge together to create your form. The calculation triggers again whenever there is a change to any value upon which the calculation is dependent, unless the calculated value has been manually overridden

by the user. It is possible to make unnecessary recalculations but not to omit any necessary ones. Calculations also trigger for all empty fields when the merging of the form design and data completes.

Note: Calculation scripts must not make any changes to the structure of the DOM. Their changes must be limited to the contents of preexisting DOM nodes.

Content inserted by a calculation must satisfy any associated validations.

Calculations and scripts on must not create an infinite loop. For example, a calculation that results in the updating of a value which is also a component of the calculation itself. This causes the form to update the value indefinitely.

This Calculation event is available from the Show list in the Script Editor:

- calculate

DOM oriented

Document Object Model (DOM) oriented events occur when a DOM changes state. Because they are not directly linked to user actions, these events execute in all form types.

These DOM events are available from the Show list in the Script Editor:

- form:ready
- layout:ready
- preSubmit

Exclusion group

Exclusion group events trigger in response to user actions that impact a group of radio buttons on a form.

These Exclusion Group events are available from the Show list in the Script Editor:

- enter
- exit
- initialize

Field oriented

Field oriented events trigger in response to user actions that affect a field. Some field events occur only on interactive forms and some occur on both all form types.

These Field events are available from the Show list in the Script Editor:

- change
- click
- enter
- exit
- full
- initialize
- mouseDown
- mouseEnter

- mouseExit
- mouseUp

Subform oriented

Subform oriented events trigger in response to changes of state of a particular subform on a form.

These Subform events are available from the Show list in the Script Editor:

- enter
- exit
- initialize

Validation

Within Designer, you can perform a number of validations using the Value tab of the Object palette. For example, you can add a validation to a numeric field so that it does not accept letters of the alphabet as input. However, this type of validation only applies to interactive forms. Furthermore, this type of validation is limited because it cannot, for example, compare the content of two fields to validate that one is larger than the other. Validation scripts provide a method to perform validations that are more specific than those available through the Value tab of the Object palette, and that can apply to different types of forms.

There are three separate validations scripts possible for any field. The order of execution of these validations is:

- The form can test the field for null content.
- The form can verify the format of the field value against a specific picture format. For more information on picture formats, see Date and time patterns.
- The form can invoke a validation script.

You can also set the severity level of each validation test to disabled, warning, or error. When any one of these validations fails and its severity is set to error, the application may skip the remaining validations for that field. However, this is not required. A severity of warning means that the user may override the validation and enter data that fails the validation.

An interactive form must trigger the enabled validations upon exit from the field or subform, provided the user has entered data into the field or subform. It is also permitted to trigger redundant validations at other times, although this is not recommended for performance reasons.

In addition, you may choose to trigger validations for all fields and subforms after the merging of data and calculations are complete. Choosing trigger validations is optional because, for example, partially blank forms can print on paper so that a user would fill in blank fields with pen and ink. This print and fill situation occurs when a partially blank form is going to be rendered into an interactive format, such as HTML.

Validation scripts are required to return true or false (expressed in a format appropriate to the scripting language) corresponding to a validation that succeeds or fails. Validation scripts must not make any changes to the DOMs, either to their structure or their content. In addition, they should not attempt to provide feedback to a form user, since that user may not be using the form in a client application such as Acrobat).

Note: Since validations are performed against the content of the form, they cannot be used to verify presentation formatting caused by picture clauses.

The following validation event is available from the Show list in the Script Editor:

- validate

List of events

All Events

This event displays an expanded Script Source field that lists all of the events for the current form design object, even if the events do not contain any calculations or scripts. This event is only available when the Script Editor is in multiline mode.

calculate

Event type: Calculation

This event triggers when your form design and data merge into your finished form. The event also triggers when there is a change to any value upon which the calculation is dependent, such as the value of a particular field, unless the calculated value has been manually overridden by the user. The properties for manually overridden field are located on the Value tab in the Object palette.

change

Event type: Field oriented

This event triggers when a user changes the content of a field. This includes when a user:

- Types a keystroke, as long as the field has keyboard focus
- Pastes data into the field.
- Makes a selection from a list box or drop-down list.
- Selects or de-selects a check box.
- Changes the setting of a group of radio buttons.

This event does not trigger in response to content changes made by an application, such as calculations, or by the merging of form design and data.

click

Event type: Field oriented

Executes when a mouse click occurs within the region.

Note: When a click event occurs for a text or numeric field, calculations or scripts execute immediately. However, the value of the field does not change in response to calculations and scripts until the field loses focus.

docClose

Event type: Application oriented

Executes at the very end of processing a form, if and only if all form validations complete with no errors. This event comes too late to modify a saved document. The purpose is to provide the ability to generate an exit status or completion message.

docReady

Event type: Application oriented

Executes prior to the rendering of the document, but after data binding of the data takes place.

enter

Event types: Subform oriented, Exclusion group, and Field oriented

This event triggers when the field gains keyboard focus, whether caused by a user action (tabbing into the field or clicking on it with the mouse) or by a script programmatically setting the focus.

Events with Scripts

This event displays an expanded Script Source field that lists only those events of a particular object that contain calculations or scripts. This event is only available when the Script Editor is in multiline mode.

exit

Event types: Subform oriented, Exclusion group, and Field oriented

This event triggers when the field loses keyboard focus, whether caused by a user action (tabbing out of the field or clicking away from it with the mouse) or by a script programmatically removing the focus.

form:ready

Event type: DOM oriented

This event triggers after the merging of your form design and data is complete, the finished form exists., and initialize and calculate events are complete.

full

Event type: Field oriented

This event triggers when the user has entered the maximum allowed amount of content into the field. The Limit Length property for a field is located on the Field tab in the Object palette.

initialize

Event types: Subform oriented, Exclusion group, and Field oriented

This event triggers after data binding is complete. A separate event is generated for each instance of the subform in the Form DOM.

layout:ready

Event type: DOM oriented

This event triggers after the merging of form design and data is complete, the form exists, and the form has had its layout applied. At this point no rendering of the finished form has taken place, so a calculation or script set to execute here could modify the layout prior to rendering.

mouseDown

Event type: Field oriented

This event triggers when a user depresses the mouse button at a moment when the mouse pointer is within the region.

Note: When a mouseDown event occurs for a text or numeric field, calculations or scripts execute immediately. However, the value of the field does not change in response to calculations and scripts until the field loses focus.

mouseEnter

Event type: Field oriented

This event triggers when the user moves the mouse pointer into the region of the field, without necessarily pressing the mouse button. It is not triggered when the mouse pointer moves into the field for some other reason, for example because an overlying window closes.

mouseExit

Event type: Field oriented

This event triggers when a user moves the mouse pointer out of the field, even if the user is depressing the mouse button. It is not triggered when the mouse pointer moves out of the field for some other reason, for example because an overlying window opens.

mouseUp

Event type: Field oriented

This event triggers when a user releases the mouse button at a moment when the mouse pointer is within the region.

Note: When a mouseup event occurs for a text or numeric field, calculations or scripts execute immediately. However, the value of the field does not change in response to calculations and scripts until the field loses focus.

postPrint

Event type: Application oriented

Executes immediately after the rendered form is sent to the printer, spooler, or output destination.

postSave

Event type: Application oriented

Executes immediately after a user saves a form in PDF or XDP format. This event does not execute when you export a subset of the form (for example, only form data) to XDP.

prePrint

Event type: Application oriented

Executes immediately before the process of rendering of a form for printing begins.

preSave

Event type: Application oriented

Executes immediately before form data is saved in PDF or XDP format. This event does not execute when the Data DOM or some other subset of the form is exported to XDP.

preSubmit

Event type: DOM oriented

This event triggers whenever a form submits data to the host via the HTTP protocol. At this point the data is organized into a data set, but has not been sent to the host. Scripts associated with this event have the chance to examine and alter the data prior to the form submission. If the script is set to execute at the server, the form sends the data to the server with an indication that it should run the script before performing any additional processing.

The preSubmit event applies only to the Form DOM. Note that the preSubmit event does not distinguish between submissions initiated by different button pushes or to different URLs. Any script that needs to make these distinctions must include code to find out what button was pushed. In general, preSubmit is analogous to preSave and serves a similar purpose.

See also

preSave

validate

Event type: Validation

This event triggers when the form design and data merge to create your form, and again whenever the value of a field changes

Understanding when events occur

About event ordering

Multiple events may be triggered by a single change of state or user action. For example, tabbing from the current field to the next field triggers both the exit event for the current field and the enter event for the next field. If the current and next fields are in different subforms, a total of four events are triggered: namely, exit events for the current field and subform and enter events for the next subform and field. Script authors should know in what order their event scripts will be executed.

Enter, exit, and validation events

Enter, exit, and validation events triggered by the same change of state use the following rules to define their order:

- When focus moves from one field, exclusion group, or subform to another, validations and exit events precede enter events.
- When focus leaves a field, exclusion group, or subform, validation precedes the exit event.
- Validations and exit events for nested elements occur in order from inner to outer element.
- Validations and enter events for nested elements occur in order from outer to inner element.

See also

enter

exit

validate

Full and change events

Full and change events triggered by the same change of state order themselves such that the change event occurs before the full event.

See also

change

full

Merge completion

Calculations, validations, and initialize events triggered by the completion of a merge operation use the following rules to define their order:

- Calculations are first, followed by all validations, and finally all initialize events.
- Calculations occur in order of depth-first traversal of the Form DOM.
- The order of validations is not defined.
- Initialize events occur in order of depth-first traversal of the Form DOM.

See also

calculate

initialize

validate

Events that cause other events

A script may cause changes of state that in turn trigger other events. A script may also directly fire an event. In such cases, the application queues the events and then runs them sequentially according to the rules for the events involved.

See also

Enter, exit, and validation events

Full and change events

Merge completion

Submit

Submit

You cannot place a calculation or script on the click event of a submit button. Instead, place your script on the preSubmit event.

See also

preSubmit

8 Using FormCalc

About FormCalc

FormCalc is a simple yet powerful calculation language modeled on common spreadsheet software. Its purpose is to facilitate fast and efficient form design without requiring a knowledge of traditional scripting techniques or languages. With the use of a few of the built-in functions, users new to FormCalc can expect to quickly create forms that save end users from performing time-consuming calculations, validations, and other verifications. In this manner you can create a basic set of rules for the form design that allows the resulting form to react according to the data it comes into contact with.

Within Designer, FormCalc is the default scripting language in all scripting locations with JavaScript as the alternative. For information on setting your default scripting language, see About the Script Editor.

Caution: If you are developing forms for use with a server-based process (for example, using Form Server), with the intent of rendering your forms in HTML, you should develop your calculations and scripts in JavaScript. FormCalc calculations are not valid in HTML browsers, and are removed prior to the form being rendered in HTML.

FormCalc treats each new line in the Script Editor as a new expression to evaluate.

See also

About JavaScript

Using built-in functions

About built-in functions

The built-in functions that make up FormCalc cover a wide range of areas, including mathematics, dates and times, strings, finance, logic, and the Web. These areas represent the types of functionality that typically occur in forms, and the purpose of the functions is to provide quick and easy manipulation of form data in a useful way.

At the most basic level, a calculation can consist of only a single FormCalc function. However, a single FormCalc function can make use of other FormCalc functions as parameters.

See also

Adding a FormCalc function to an object

You can add a FormCalc function to any form design object that allows calculations and scripts, with the exception of the script object.

➤ **To attach a FormCalc function to an object:**

1. Make sure that you have the multiline version of the Script Editor showing on the Designer workspace.
2. Select a field on your form.

3. From the Show list, select the calculate event.
4. Click Functions  .
5. Select the desired function and press Enter.
6. Replace the default function syntax notation with your own set of values.
7. Click Enter Script Source Changes  to add the FormCalc calculation to your form.

Function syntax

Each FormCalc function uses a specific syntax notation that you must follow in order for the function to execute correctly. The table below describes, very generally, the pieces of syntax notation

Syntax Notation	Replacement Values
d	A valid date string (for example, 03/15/1996)
f	A valid date format string (for example, MM/DD/YYYY)
k	A valid locale identifier (for example, fr_FR)
n	A valid numeric value. Note that the range of valid values varies from function to function.
s	A valid unit of measurement (for example, “in” for inches).
v	A valid accessor value
n1, n2, n3	All values are required.
[[n [, k]]]	No values are required, but you can choose to specify just n, or both n and k.
n1 [, n2 ...]	n1 is required, but you can choose to specify an unlimited number of additional values.
d [, f [, k]]	d is required, but you can choose to also specify f or both f and k.

Creating simple expressions

About simple expressions

Simple expressions are the most basic instances of scripting. These expressions do not involve using FormCalc built-in functions and are never more than a single line in size. You must add simple expressions to the calculate event of a particular field or object in order for the value of the expression to output onto your form.

Examples of simple expressions

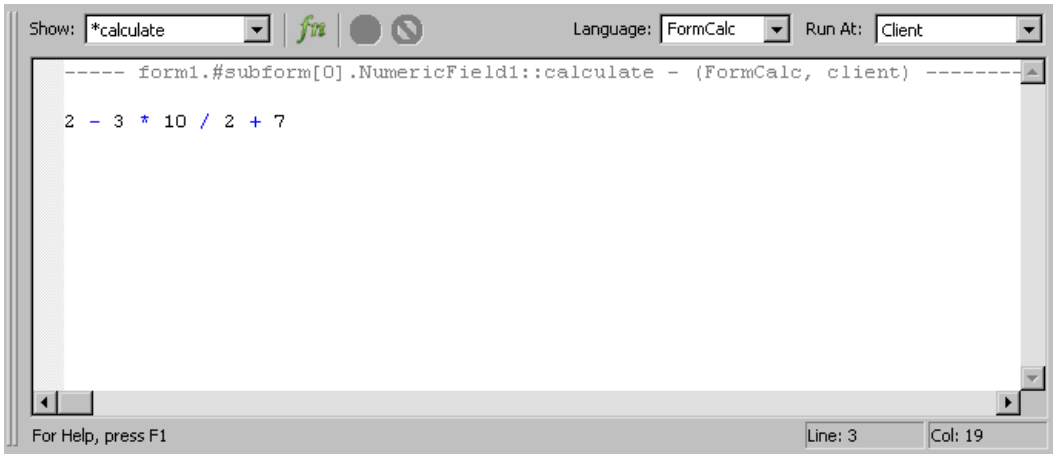
These are all examples of simple expressions:

```
2
"abc"
2 - 3 * 10 / 2 + 7
```

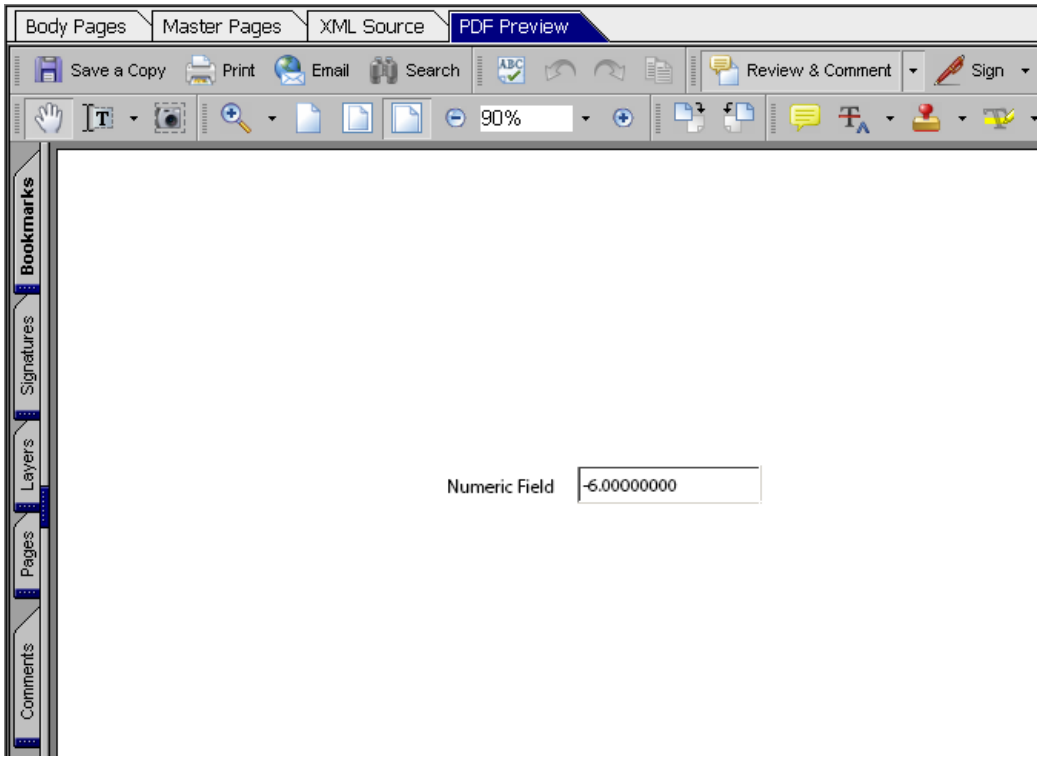
Each simple expression evaluates to a single value by following a traditional order of operations, even if that order is not always obvious from the expression syntax. For example, the following sets of expressions produce equivalent results:

Expression	Equivalent to	Result
`"abc"`	`"abc"`	`abc`
`2 - 3 * 10 / 2 + 7`	`2 - (3 * 10 / 2) + 7`	`-6`
`(10 + 2) * (5 + 4)`	`(10 + 2) * (5 + 4)`	`240`
`0 and 1 or 2 > 1`	`(0 and 1) or (2 >1)`	`1` (true)
`2 < 3 not 1 == 1`	`(2 < 3) not (1 == 1)`	`0` (false)

All of the above examples are valid simple expressions that you can add to a form field or object that will accept calculations and scripts. For example, if you create a new form in Designer with a single numeric field, and add the following calculation to the calculate event in the Script Editor:



Then when you select the Preview tab to view the completed form, the value of the simple expression appears in the text field.



Note: If the value does not appear in the preview, ensure that your simple expression appears in the calculate event of the form design object. You should also ensure that you have correctly installed Designer and the Acrobat 6.0.1 upgrade.

Referencing field values in calculations

The current container

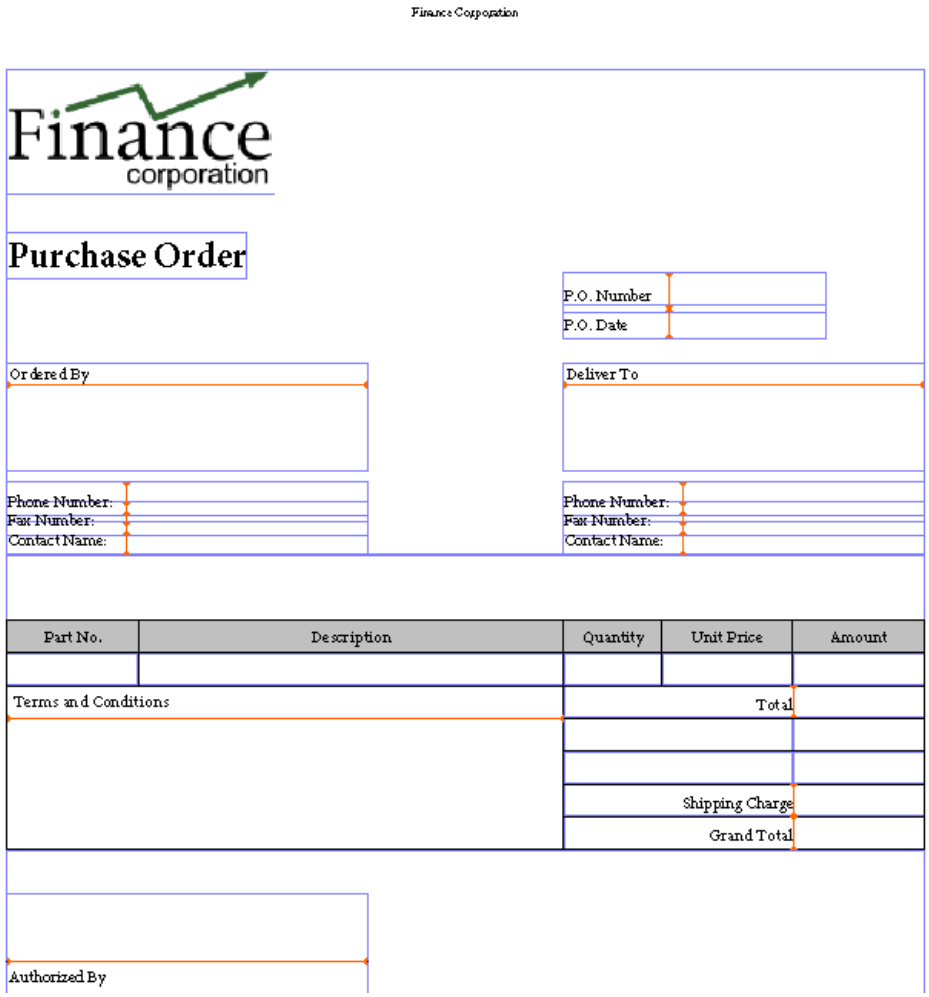
Within a form there is a concept of a container. A container is an object that holds data or values. Simple containers, those that are not capable of holding other containers or objects, include fields (text, numeric, buttons) and drawn objects (static text, circle, line). All containers capable of holding other containers as well as non-container objects are considered complex containers. Subforms are an example of a complex container.

Unqualified references to objects located in the same container

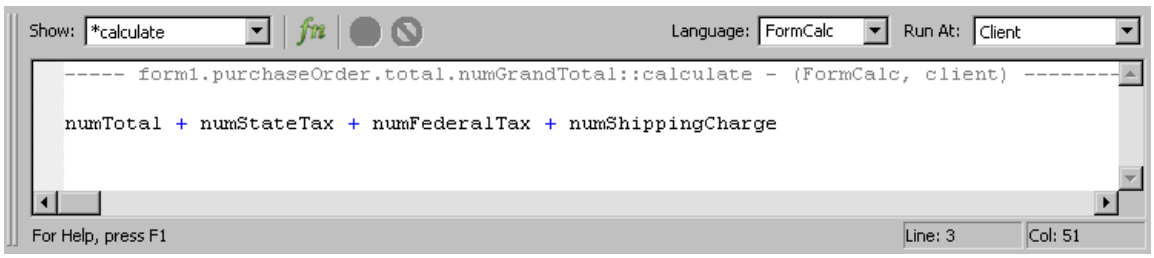
FormCalc allows you to reference the value of a field in a calculation or expression simply by referencing the name of the field, as long as the field containing the calculation is in the same container as the field being referenced. The container of an object is where that object is located within a form hierarchy, with respect to all other fields, subforms, and other objects. In general, an object on a subform is within the same container as every other object on that subform. That same object is considered to be in a different container with respect to any objects located on any other subform.

In the simplest case, if you have a one page form design with no user-created subforms, then all of your fields and objects are considered to be in the same context. This situation allows you to reference any field or object value by stating the name of the that field or object.

The dynamic version of the Purchase Order sample that ships with Designer provides an example of calculations that reference field values. By default, the Purchase Order.xdp file is located in C:\Program Files\Adobe\Designer 6.0\Samples\Purchase Order\Dynamic\Forms. Opening the XDP file in Designer displays the following form design.



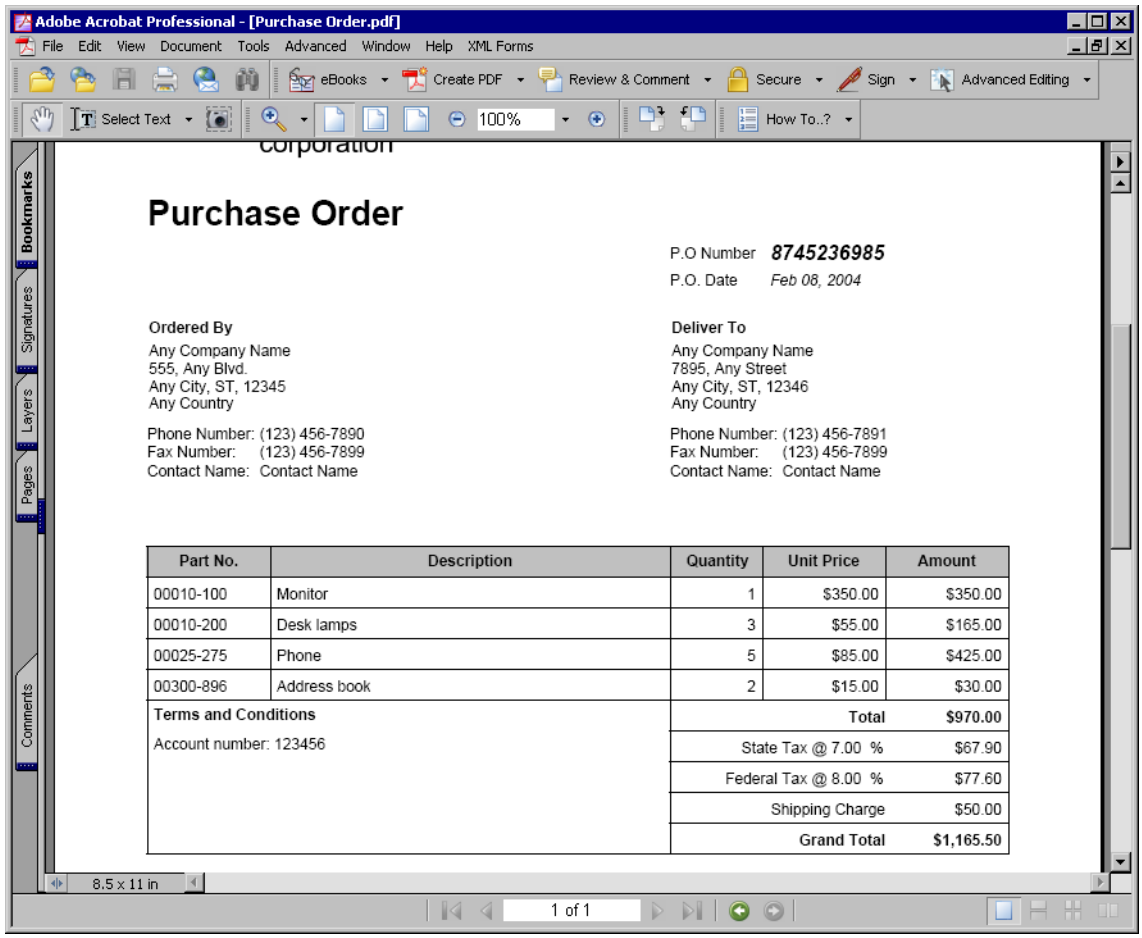
The Grand Total field on the form contains the following script located on the calculate event.



In words, this script adds together the values of the Total field, the State Tax field, the Federal Tax field, and the Shipping Charge field. The result of this entire calculation then displays in the Grand Total field on the form. Notice that only the field names are used to reference the field values.

In this example, the numTotal, numStateTax, numFederalTax, and numShippingCharge fields are all considered to be in the same context because they all exist within the same subform.

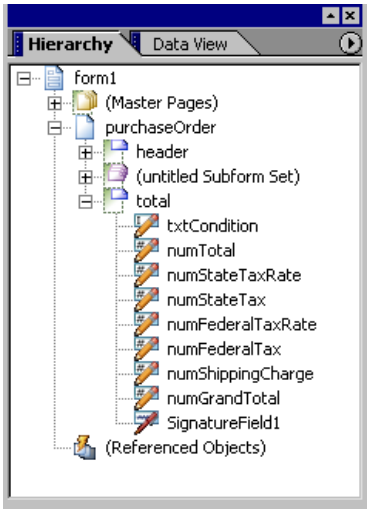
Opening the output of this Purchase Order sample in Acrobat illustrates the effect this calculation has at run time. By default, the example output is located in C:\Program Files\Adobe\Designer 6.0\Samples\Purchase Order\Dynamic\Outputs.



Referencing objects located in different containers

Due to the highly structured nature of form designs created in Designer, a Scripting Object Model (SOM) exists that allows you to easily reference any object on your form. A SOM expression is the representation of the object, value, property, or method you are referencing. In terms of your form design, if you are attempting to reference field or object values located on different subforms or subform sets, then you are referencing values in a different container.

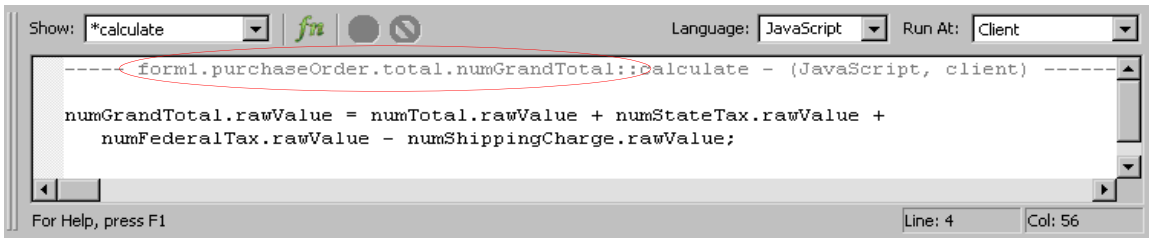
To access values from objects in different containers, you must specify the location of that object as part of your SOM syntax. For example, consider this form hierarchy from the dynamic Purchase Order sample that ships with Designer.



The fully qualified SOM expression of the `numGrandTotal` field is:

```
form1.purchaseOrder.total.numGrandTotal
```

Note: The Script Editor displays a fully qualified SOM expression for all fields and objects in multiline view.



This fully qualified SOM expression points to the value of the `numGrandTotal` field from any point on the form. So if you wanted to create a new field on another subform that divides the grand total into 12 equal payments, your script would look like this:

```
newField = form1.purchaseOrder.total.numGrandTotal / 12
```

Repeated fields

When two or more nodes with the same name occur within a container, a reference to the shared name is taken to refer to the first matching field in the form order. For example, in the dynamic Purchase Order sample, the fields on the detail subform (txtPartNum, txtDescription, numQty, numUnitPrice, and numAmount) can appear multiple times. In this case, referencing the fields using only their name would return the first occurrence of the fields.

```
newField = txtDescription
```

To access the other instances of txtDescription, given that they have the same name, it is necessary to use an array-subscript notation. The syntax [*n*], where *n* represents a number, is used to select one particular field or object out of a group of objects with the same name. For example:

```
newField = txtDescription[3]
```

This array-subscript notation is zero-based, so the above example would reference the fourth instance of txtDescription. The number zero represents the first instance.If you are referencing repeated fields from another container on your form, then the array-subscript notation is applied to your fully qualified SOM expression. For example:

```
newField = form1.purchaseOrder.#subformSet[0].detail.txtDescription[3]
```

Notice in the above example that the subformSet also makes use of the array-subset notation. This is because it is possible to have container objects with the same name in a similar way that you can have fields or objects with the same name.

See also

About accessors

If Expressions

An if expression is a conditional statement that evaluates a given simple expression for truth, and then returns the result of a list of expressions that correspond to the truth value. If the initial simple expression evaluates to false (0), then FormCalc examines any elseif and else conditions for truth and returns the results of their expression lists if appropriate.

Expression	Syntax	Returns

If	`if ( simple expression ) then`
	`  list of expressions`
	`elseif ( simple expression ) then`
	`  list of expressions`
	`else`
	`  list of expressions`
	`endif`
		The result of the list of expressions associated with any valid conditions stated in the if expression.
		Note: You are not required to have any elseif(...) or else statements as part of your if expression, but you must state the end of the expression with endif.

The following are examples of using the if expression:

Expression	Result

`if ( 1 < 2 ) then`
`  1`
`endif`
	`1`

Expression	Result

`if ( "abc" > "def") then`
`  1`
`else`
`  0`
`endif`
	`0`

`if ( numTotal < 1000 ) then`
`  numShippingCharge = 30`
`elseif ( 1000 < numTotal < 2000 ) then`
`  numShippingCharge = 15`
`elseif ( numTotal > 2000 ) then`
`  numShippingCharge = 0`
`endif`
	Varies with the values of `numTotal`. For example, if `numTotal` is 1250, then this expression sets `numShippingCharge` to 15.

9 Using JavaScript

About JavaScript

In order to allow form designers more flexibility and scripting power, Designer supports the use of JavaScript in all situations that support scripting.

Form developers familiar with JavaScript will be able to take their existing expertise and apply it directly to Designer. Designer provides a number of properties and methods that enhance JavaScript to allow you access field and object values. These properties and methods combine with the Designer Scripting Object Model (SOM) to provide you with easy manipulation of form values and data.

Scripting Object Model (SOM)

The current container

Within a form there is a concept of a container. A container is an object that holds data or values. Simple containers, those that are not capable of holding other containers or objects, include fields (text, numeric, buttons) and draws (static text, circle, line). All containers capable of holding other containers as well as non-container objects are considered complex containers. Subforms are an example of a complex container.

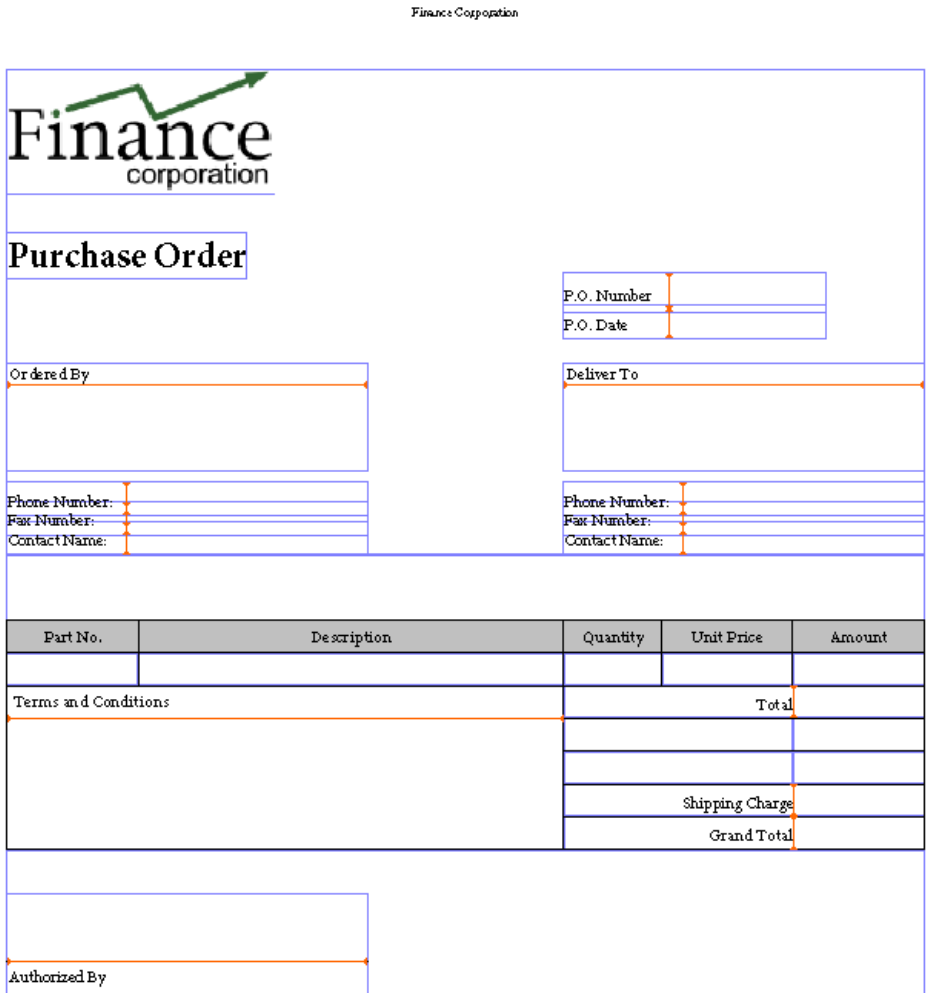
Unqualified references to objects located in the same container

Designer allows you to reference the value of a field in a calculation or expression simply by referencing the name of the field, as long as the field containing the calculation is in the same container as the field being referenced. The container of an object is where that object is located within a form hierarchy, with respect to all other fields, objects, and subforms. In general, an object on a subform is within the same container as every other object on that subform. That same object is considered to be in a different container with respect to any objects located on any other subform.

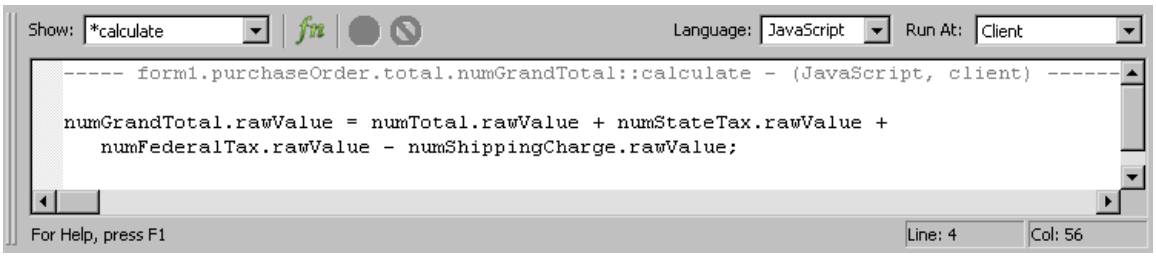
In the simplest case, if you have a one page form design with no user-created subforms, then all of your fields and objects are considered to be in the same context. This allows you to reference any field or object value by stating the name of the that field or object.

The dynamic version of the purchase order sample that ships with Designer provides an example of scripts that reference field values. By default, the Purchase Order.xdp file is located in

C:\Program Files\Adobe\Designer 6.0\Samples\Purchase Order\Dynamic\Forms. Opening the XDP file in Designer loads the following form design.



The Grand Total field on the form contains the following script located on the calculate event.



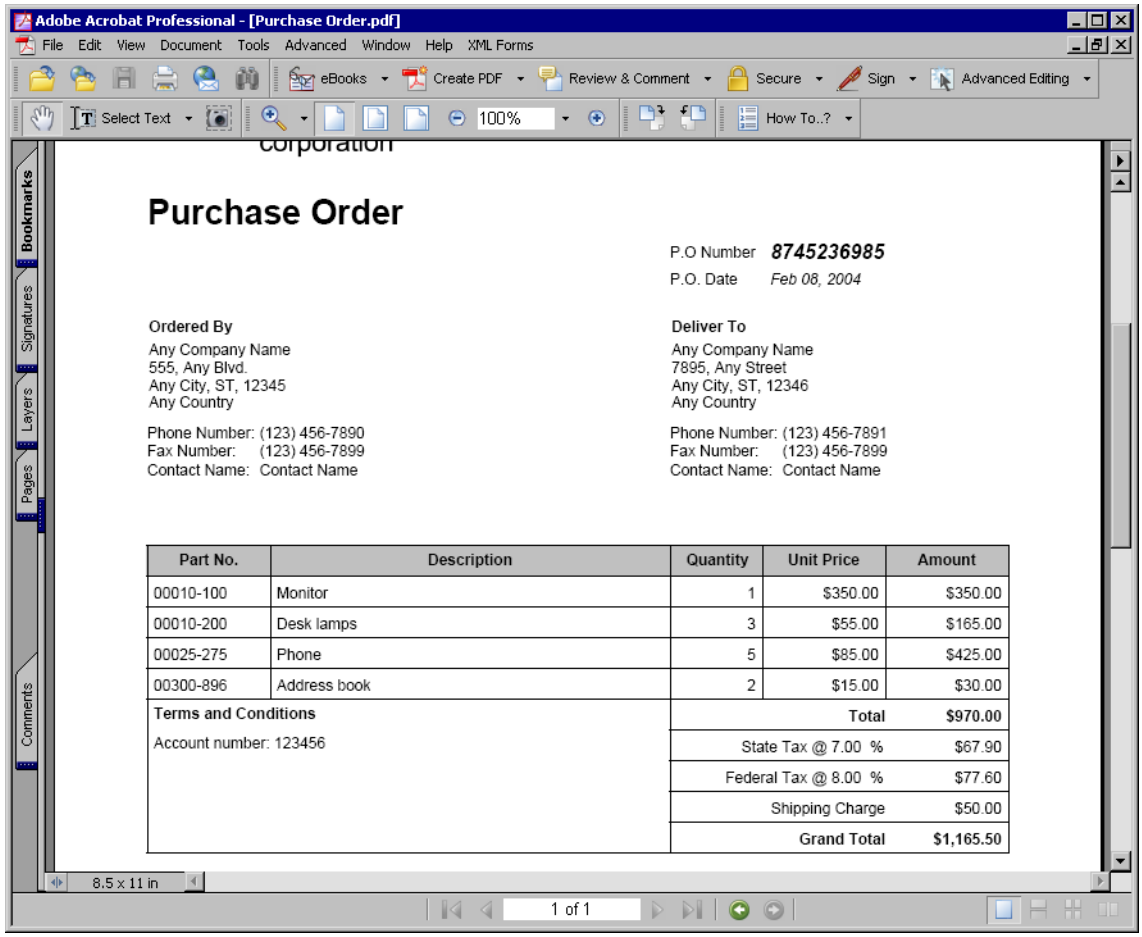
In words, this script adds the values of the Total field, the State Tax field, and the Federal Tax field together, and subtracts the value of the Shipping Charge field. The result of this entire calculation then displays in the Grand Total field on the form.

In this example, the numTotal, numStateTax, numFederalTax, and numShippingCharge fields are all considered to be in the same context because they all exist within the same subform.

Notice the use of the rawValue property. This property references the actual value of the field or object. You could also make use of common JavaScript syntax and reference the Grand Total field using the identifier `this`. For example, the following script returns the same value as the one above:

```
this.rawValue = numTotal.rawValue + numStateTax.rawValue +
  numFederalTax.rawValue - numShippingCharge.rawValue;
```

Opening the output of this purchase order sample in Acrobat illustrates the effect this calculation has at run time. By default, the example output is located in C:\Program Files\Adobe\Designer 6.0\Samples\Purchase Order\Dynamic\Outputs.



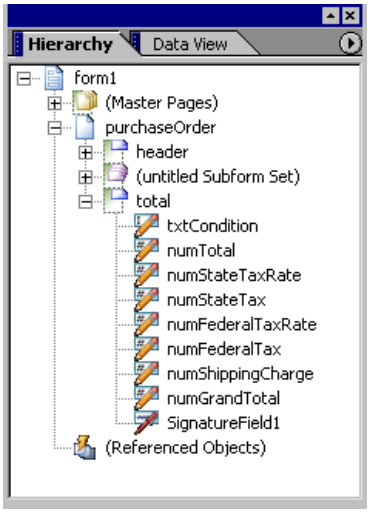
See also

About accessors

Referencing objects located in different containers

Due to the highly structured nature of form designs created in Designer, a Scripting Object Model (SOM) exists that allows you to easily reference any object on your form. A SOM expression is representation of the object, value, property, or method you are referencing. In terms of your form design, if you are attempting to reference field or object values located on different subforms or subform sets, then you are referencing values in a different container.

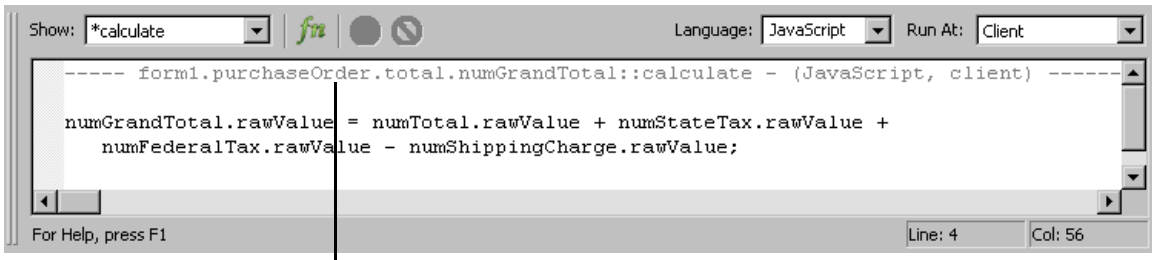
To access values from objects in different containers you must specify the location of that object as part of your SOM syntax. For example, consider this form hierarchy from the dynamic Purchase Order sample that ships with Designer.



The fully qualified SOM expression of the `numGrandTotal` field would be:

```
form1.purchaseOrder.total.numGrandTotal
```

Note that the Script Editor displays a fully qualified SOM expression for all fields and objects in multiline view.



Fully qualified SOM expression for the numGrandTotal field.

This fully qualified SOM expression will point to the value of the `numGrandTotal` field from any point on the form. So if you wanted to create a new field on another subform that divided the grand total into 12 equal payments, your script might look like the following:

```
this.rawValue = form1.purchaseOrder.total.numGrandTotal.rawValue / 12;
```

Repeated fields

When two or more nodes with the same name occur within a container, a reference to the shared name is taken to refer to the first matching field in the form order. For example, in the dynamic purchase order sample that ships with Designer, the fields on the detail subform (txtPartNum, txtDescription, numQty, numUnitPrice, and numAmount) can appear multiple times. In this case, referencing the fields using only their name would return the first occurrence of the fields.

```
newField.rawValue = txtDescription.rawValue;
```

To access the other instances of txtDescription, given that they have the same name, it is necessary to use an array-subscript notation. The syntax [*nnn*], where *nnn* represents a number, is used to select one particular field or object out of a group of objects with the same name. For example:

```
newField.rawValue = txtDescription[3].rawValue
```

This array-subscript notation is zero-based, so the above example would reference the fourth instance of txtDescription. The number zero represents the first sibling. Hence the following two expressions are equivalent:

```
newfield.rawValue = txtDescription.rawValue;
newField.rawValue = txtDescription[0].rawValue;
```

If you are referencing repeated fields from another container on your form then the array-subscript notation is applied to your fully qualified SOM expression. For example:

```
newField.rawValue =
form1.purchaseOrder.#subformSet[0].detail.txtDescription[3].rawValue;
```

Notice in the above example that the subformSet also makes use of the array-subset notation. This is because it is possible to have container objects with the same name in a similar way that you can have fields or objects with the same name.

Differences between FormCalc and JavaScript functions

Although FormCalc and JavaScript are geared towards two different types of users, there is some overlap between the types of built-in functions they offer. The following table lists all available FormCalc functions and lists whether a comparable function exists within JavaScript.

FormCalc function	Description	JavaScript equivalent exists
Abs(n1)	Returns the absolute value of a numeric value or expression.	Yes
Apr(n1, n2, n3)	Returns the annual percentage rate for a loan.	No
At(s1,s2)	Locates the starting character position of a string within another string.	Yes
Avg(n1 [, n2...])	Evaluates a set of number values and/or expressions and returns the average of the non-null elements contained within that set.	No
Ceil(n1)	Returns the whole number greater than or equal to a given number.	Yes
Choose(n1, s1 [, s2...])	Selects a value from a given set of parameters.	No
Concat(s1 [, s2...])	Returns the concatenation of two or more strings.	Yes

FormCalc function	Description	JavaScript equivalent exists
Count(n1 [, n2...])	Evaluates a set of values and/or expressions and returns the number of non-null elements contained within the set.	No
CTerm(n1, n2, n3)	Returns the number of periods needed for an investment earning a fixed, but compounded, interest rate to grow to a future value.	No
Date()	Returns the current system date as the number of days since the epoch.	Yes
Date2Num(d1[, f1[, k1]])	Returns the number of days since the epoch, given a date string.	No
DateFmt([n1[, k1]])	Returns a date format string, given a date format style.	No
Decode(s1 [, s2])	Returns the decoded version of a given string.	Partial support JavaScript only supports URL encoded values that contain no escape characters.
Encode(s1 [, s2])	Returns the encoded version of a given string.	Partial support JavaScript only supports URL encoded values that contain no escape characters.
Eval()	Returns the value of a given form calculation.	Yes
Exists(v1)	Determines whether the given parameter is an accessor to an existing object.	No
Floor(n1)	Returns the largest whole number that is less than or equal to the given value.	Yes
Format(s1, s2)	Formats the given data according to the specified picture format string.	No
FV(n1, n2, n3)	Returns the future value of consistent payment amounts made at regular intervals at a constant interest rate.	No
Get(s1)	Downloads the contents of the given URL.	No

FormCalc function	Description	JavaScript equivalent exists
HasValue(v1)	Determines whether the given parameter is an accessor with a non-null, non-empty, or non-blank value.	No
IPmt(n1, n2, n3, n4, n5)	Returns the amount of interest paid on a loan over a set period of time.	No
IsoDate2Num(d1)	Returns the number of days since the epoch, given an valid date string.	No
IsoTime2Num(d1)	Returns the number of milliseconds since the epoch, given a valid time string.	No
Left(s1, n1)	Extracts a specified number of characters from a string, starting with the first character on the left.	Yes
Len(s1)	Returns the number of characters in a given string.	Yes
LocalDateFmt([n1[, k1]])	Returns a localized date format string, given a date format style.	No
LocalTimeFmt([n1[, k1]])	Returns a localized time format string, given a time format style.	No
Lower(s1[, k1])	Converts all uppercase characters within a specified string to lowercase characters.	No
Ltrim(s1)	Returns a string with all leading white space characters removed.	No
Max(n1 [, n2...])	Returns the maximum value of the non-null elements in the given set of numbers.	No
MessageBox(n1, "s1" [,"s2"])	Returns the minimum value of the non-null elements of the given set of numbers.	Yes
Min(n1 [, n2...])	Returns the modulus of one number divided by another.	No
Mod(n1, n2)	Returns the net present value of an investment based on a discount rate and a series of periodic future cash flows.	Yes
NPV(n1, n2 [, ...])	Returns the null value. The null value means no value.	No
Num2Date(n1[, f1[, k1]])	Returns a date string, given a number of days since the epoch.	No

FormCalc function	Description	JavaScript equivalent exists
Num2GMTime(n1 [,f1 [, k1]])	Returns a GMT time string, given a number of milliseconds from the epoch.	No
Num2Time(n1 [,f1 [, k1]])	Returns a time string, given a number of milliseconds from the epoch.	No
Oneof(s1, s2 [, s3...])	Returns true (1) if a value is in a given set, and false (0) if it is not.	Yes
Parse(s1, s2)	Analyzes the given data according to the given picture format.	No
Pmt(n1, n2, n3)	Returns the payment for a loan based on constant payments and a constant interest rate.	No
Post(s1, s2[, s3[, s4[, s5]]])	Posts the given data to the specified URL.	No
PPmt(n1, n2, n3, n4, n5)	Returns the amount of principal paid on a loan over a period of time.	No
Put(s1, s2[, s3])	Uploads the given data to the specified URL.	No
PV(n1, n2, n3)	Returns the present value of an investment of periodic constant payments at a constant interest rate.	No
Rate(n1, n2, n3)	Returns the compound interest rate per period required for an investment to grow from present to future value in a given period.	No
Ref()	Returns a reference to an existing object.	Yes
Replace(s1, s2[, s3])	Replaces all occurrences of one string with another within a specified string.	Yes
Right(s1, n1)	Extracts a number of characters from a given string, beginning with the last character on the right.	Yes
Round(n1 [, n2])	Evaluates a given numeric value or expression and returns a number rounded to the given number of decimal places.	Yes
Rtrim(s1)	Returns a string with all trailing white space characters removed.	No

FormCalc function	Description	JavaScript equivalent exists
Space(n1)	Returns a string consisting of a given number of blank spaces.	No
Str(n1 [, n2 [, n3]])	Converts a number to a character string. FormCalc formats the result to the specified width and rounds to the specified number of decimal places.	No
Stuff(s1, n1, n2[, s2])	Inserts a string into another string.	No
Substr(s1, n1, n2)	Extracts a portion of a given string.	Yes
Sum(n1 [, n2...])	Returns the sum of the non-null elements of a given set of numbers.	No
Term(n1, n2, n3)	Returns the number of periods needed to reach a given future value from periodic constant payments into an interest-bearing account.	No
Time()	Returns the current system time as the number of milliseconds since the epoch.	Yes
Time2Num(d1[, f1[, k1]])	Returns the number of milliseconds since the epoch, given a time string.	No
TimeFmt([n1[, k1]])	Returns a time format, given a time format style.	No
UnitType(s1)	Returns the units of a unitspan. A unitspan is a string consisting of a number followed by a unit name.	No
UnitValue(s1 [, s2])	Returns the numeric value of a measurement with its associated unitspan, after an optional unit conversion.	No
Upper(s1[, k1])	Converts all lowercase characters within a string to uppercase.	No
Uuid(n1)	Returns a Universally Unique Identifier (UUID) string to use as an identification method.	
Within(s1, s2, s3)	Returns true (1) if the test value is within a given range, and false (0) if it is not.	Yes
WordNum(n1 [, n2 [, k1]])	Returns the English text equivalent of a given number.	No

See also

Using FormCalc

10 Variables

About variables

You can define text variables in Designer in order to store specific information in a central, accessible location. A variable typically acts as a placeholder for text that you might have to change at some point in the future. When the text needs to change, all you have to do is open the affected form or template and update the text once through the variable definition. Designer automatically propagates the new text across all instances of the inserted variable.

You can create, view, and delete variables without requiring the use of scripting. However, you must use scripting in order to access the values stored by variables and manipulate them, or to apply the values to objects on your form.

Creating, viewing, and deleting variables

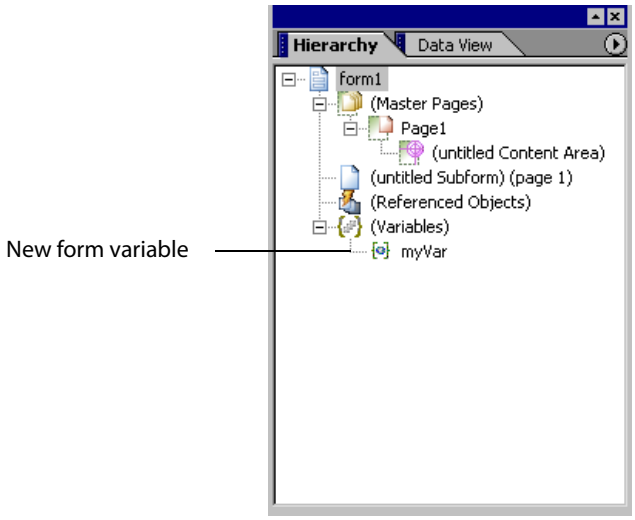
Before you create a variable, decide the name of the variable and the text that it will contain. Variable definitions are saved with the form or template.

➤ **To define a text variable:**

1. Select File > Form Properties.
2. In the Variables tab, click New (Insert)  .
3. In the Variables list, type a unique name for the variable and press Enter. Variable names are not case-sensitive and should not contain any spaces.

   Note: You should use as distinctive a variable name as possible
4. Click once in the box to the right and type the text you want to associate with the variable.

The variable appears in the Hierarchy palette at the form level.



➤ **To view a text variable definition:**

1. Select File > Form Properties.
2. Click the Variables tab. From the Variables list, select the variable. The associated text is displayed in the box to the right.

➤ **To delete a text variable:**

1. Select File > Form Properties.
2. In the Variables tab, select the variable.
3. Click Delete  .

Using variables in calculations and scripts

Once you have created form variables, you only need to reference the variable name in your calculations and scripts in order to obtain the value of the variable.

For example, assume the following form variable definitions:

Variable name	Value
firstName	Tony
lastName	Blue
age	32

Caution: When naming variables you should avoid using names that are identical to the names of any XML Form Object Model properties. For information on XML Form Object Model properties, see Properties in the Object Reference.

In FormCalc you can access the variable values in the same manner that you access field and object values. In this example, the values are assigned to three separate fields.

```
TextField1 = firstName
TextField2 = lastName
NumericField1 = age
```

You can also use variables in FormCalc functions in the same way. For example:

```
Concat( "Dear ", firstName, lastName )
```

In JavaScript, you reference variable values using the .value property, instead of the .rawValue property that is used for field and object values. For example:

```
TextField1.rawValue = firstName.value;
```

11 Accessors

About accessors

For form designs, a qualified hierarchy allows access to all of the object properties and values on the form design. The XML Form Object Model provides access to form design object properties and values through accessors, which provide predefined syntax that makes navigation of this hierarchy easier. An accessor is a mechanism that allows you to either assign or retrieve specific object values and properties without providing the entire location as part of your SOM expression.

Accessor syntax

The following table outlines the correct syntax for all accessors.

Notation	Description

$ (FormCalc)
this (JavaScript)
	Refers to the current field or object. For example:
	`$ = "Tony Blue"`
	`this.rawValue = "Tony Blue"`
	The above examples set the value of the current field or object to `Tony Blue` using both FormCalc and JavaScript.
	Note: These accessors must appear at the beginning of a hierarchy reference, that is, before the first period.

!	Represents the root of the data model, xfa.datasets. For example:
	`!dbresults`
	is equivalent to:
	`xfa.datasets.dbresults`
	Note: This accessor must appear at the beginning of a hierarchy reference, that is, before the first period.

$data (FormCalc)
xfa.datasets.data (JavaScript)
	Represents the root of the data model, xfa.datasets.data. For example:
	`$data.purchaseOrder.total`
	is equivalent to:
	`xfa.datasets.data.purchaseOrder.total`
	Note: This accessor must appear at the beginning of a hierarchy reference, that is, before the first period.

$template (FormCalc)
xfa.template (JavaScript)
	Represents the root of the template model, xfa.template. For example:
	`$template.purchaseOrder.item[1]`
	is equivalent to:
	`xfa.template.purchaseOrder.item[1]`
	Note: This accessor must appear at the beginning of a hierarchy reference, that is, before the first period.

Notation	Description

$form (FormCalc)
xfa.form (JavaScript)
	Represents the root of the form model, xfa.form. For example:
	`$form.purchaseOrder.tax[0]`
	is equivalent to stating:
	`xfa.form.purchaseOrder.tax[0]`
	Note: This accessor must appear at the beginning of a hierarchy reference, that is, before the first period.

$layout (FormCalc)
xfa.layout (JavaScript)
	Represents the root of the layout model, xfa.layout. For example:
	`$layout.purchaseOrder.tax[0]`
	is equivalent to stating:
	`xfa.layout.purchaseOrder.tax[0]`

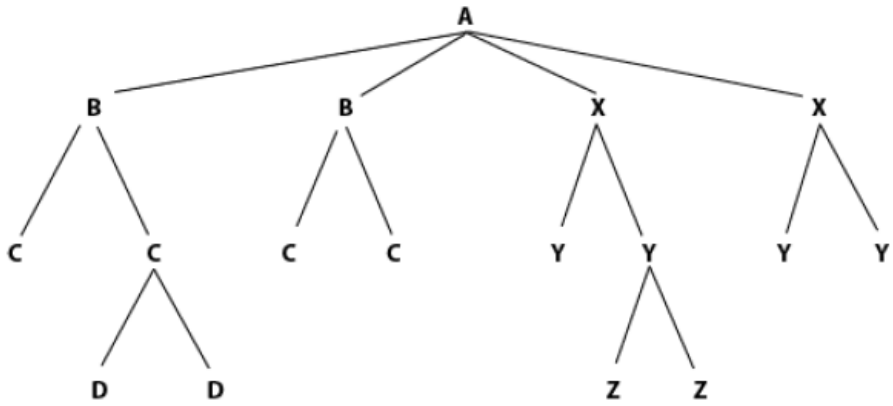
$record (FormCalc) xfa.record (JavaScript)	Represents the current record of a collection of data, such as from an XML file. For example: `$record.header.txtOrderedByCity` references the `txtOrderedByCity` node within the `header` node of the current XML data. **Note:** This accessor must appear at the beginning of a hierarchy reference, that is, before the first period.

$event (FormCalc)
xfa.event (JavaScript)
	Represents the current form object event. For example:
	`$event.name`
	is equivalent to:
	`xfa.event.name`
	Note: This accessor must appear at the beginning of a hierarchy reference, that is, before the first period.

$host (FormCalc)
xfa.host (JavaScript)
	Represents the host object. For example:
	`$host.name`
	is equivalent to:
	`xfa.host.name`
	Note: This accessor must appear at the beginning of a hierarchy reference, that is, before the first period.

*	Selects all form objects within a given container regardless of name, or selects all objects with a similar name. This accessor cannot begin a SOM expression.
	For example, the following expression selects all containers on a form design:
	`$template.*`
	The following expression selects all objects named `item`:
	`form1.item[*]`

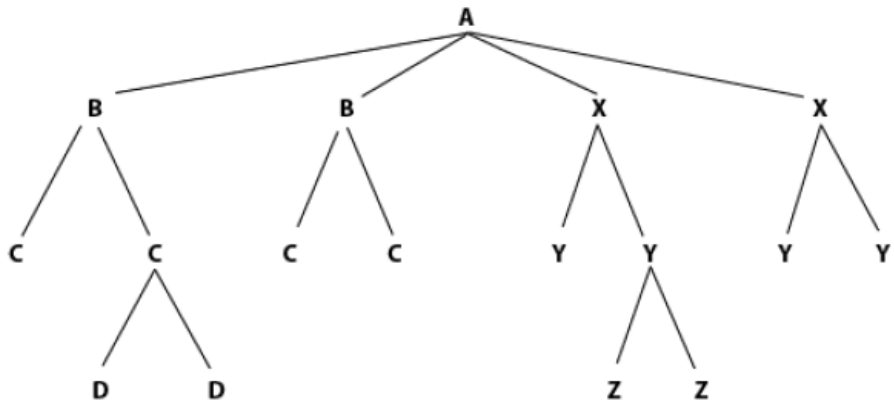
Notation	Description

..	You can use two dots at any point in your SOM expression to search for objects that are a part of any subcontainer of the current container. For example, the expression `A..B` means locate the node `A` (as usual), and find a descendant of `A` called `B`.
	 
	
	Using the example tree above:
	`A..C`
	is equivalent to:
	`A.B[0].C[0]`
	because `C[0]` is the first `C` node FormCalc encounters on its search. As a second example:
	`A..C[*]`
	returns the two `C` nodes on the extreme left, because in this example, `A..C[*]` is equivalent to `A.B[0].C[*]`.

#	Matches an unnamed form design object or property. This accessor is useful if both a property and an object have the same name. The number sign (#) ensures that the script accesses the property value. For example: `purchaseOrder.#name` This expression returns the actual name of the purchase order object, in this case `purchaseOrder`.

Notation	Description

[]	An array referencing syntax. FormCalc treats the collection of accessible objects with the same name as an array. Note that all array references are zero-based.
	In order to construct an array element reference, place square brackets ([ ]) after a qualified accessor name, and enclose within the brackets one of the following:
	• `[ n ]`
	Where `n` is an absolute occurrence index number beginning at 0. An occurrence number that is out of range is an error. Occurrence numbers in SOM syntax are not expressions.
	Only numbers are valid. For example:
	`$xfa.template.Quantity[3]`
	refers to the fourth occurrence of Quantity.
	• `[ +/- n ]`
	Where `n` indicates an occurrence relative to the occurrence of the object making the reference. Positive values yield higher occurrence numbers while negative values yield lower occurrence numbers. For example:
	`$xfa.template.Quantity[+2]`
	This expression yields the occurrence of `Quantity` whose occurrence number is two more than the occurrence number of the container making the reference. For example, if this reference was attached to `Amount[2]`, the reference would be the same as:
	`$xfa.template.Quantity[4]`
	If the computed index number is out of range, it is an error.
	The most common use of this syntax is for locating the previous or next occurrence of a particular field. For example, every occurrence of the field `Amount` (except the first) might use `Amount[-1]` to get the value of the previous amount field.
	• `[*]`
	Indicates multiple occurrences of the object. The first named object is found, and objects of the same name that are siblings to the first are returned. Note that using this notation returns in a collection of objects. For example:
	`Quantity[*]`
	This expression refers to all objects with a name of `Quantity` that are siblings to the first `Quantity` found.

Notation	Description
[ ] (Continued)	 Using the tree for reference, these expressions return the following • `A.B[*]` Both B nodes • `A.B.C[*]` Two `C` nodes on the extreme left. `A.B` resolves to the first `B` node on the left, and the `C[*]` is evaluated relative to that node. • `A.B[*].C` The first and the third `C` nodes from the left. `A.B[*]` resolves to both `B` nodes, and the `C` is evaluated relative to both of those `B` nodes. • `A.B[*].C[1]` The second and fourth `C` nodes from the left. `A.B[*]` resolves to both `B` nodes, and the `C[1]` is evaluated relative to both of those `B` nodes. • `A.B[*].C[*]` All four `C` nodes. • `A.*` Both `B` nodes and both `X` nodes • `A.X.*` Two left-most `Y` nodes.

The host accessor

About the host accessor

The host accessor is a special SOM accessor that provides a direct interface with the hosting application. For example, in the case of an interactive form, the host accessor references the client application (Acrobat or a web browser). Using the host accessor properties and methods, you can retrieve information and execute actions that are not otherwise accessible through calculations and scripts. For example, you can retrieve the name of the host application (such as Acrobat), or advance the current page on an interactive form.

The host accessor is valid on any form design object that has events for scripting. You specify the host accessor by using the following syntax:

```
$host.property_or_method (FormCalc)
xfa.host.property_or_method (JavaScript)
```

See also

Host accessor properties and methods

Comparing the host accessor functionality

About accessors

Host accessor properties and methods

$host.appType

This property determines the type of application in which the document currently exists. For example, in the context of a PDF form viewed in Adobe Reader, this property returns `Reader`.

Syntax

```
$host.appType
```

Parameters

None

Returns

String

$host.beep

This method causes the system to play a sound.

Syntax

```
$host.beep( [ INTEGER param ] )
```

Parameters

Parameter	Description
param	The system code for the appropriate sound. • 0 (Error) • 1 (Warning) • 2 (Question) • 3 (Status) • 4 (Default) This is the default value for this method.

Returns

Empty

$host.currentPage

This property returns and sets the currently active page of a document. Page values are 0-based, so the first page of a document returns a value of 0.

Syntax

```
$host.currentPage
```

Parameters

None

Returns

Integer

$host.exportData

This method exports data in either XDP or XML format to a file.

Syntax

```
$host.exportData( [ STRING param1 [, BOOLEAN param2 ] ])
```

Parameters

Parameter	Description
param1 (*optional*)	Specifies the location and file name of the file where the data will export. If you omit this parameter, a dialog is shown to let the user select the file manually. **Note:** This parameter is only valid on certified documents where the user has sufficient permissions.
param2 (*optional*)	Indicates the format to export the information in. • 0 (default) Export to XDP format. • 1 Export plain XML data. **Note:** In order to change the export type without specifying a file name, you must provide an empty string as the first parameter. For example: `$host.exportData("",0)`

Returns

Empty

$host.gotoURL

This method retrieves the specified URL over the internet.

Syntax

```
$host.gotoURL( STRING param1 [, BOOLEAN param2 ] )
```

Parameters

Parameter	Description
param1	A fully qualified URL.
param2 (*optional*)	Indicates the format to export the information in. • 0 (default) Append the pages from the specified URL to the current document. • 1 The URL opens in a separate document or window.

$host.importData

This method imports data from a specified file.

Syntax

```
$host.importData( [ String param ] )
```

Parameters

Parameter	Description
param (*optional*)	Specifies the location and file name of the file where the data will export. If you omit this parameter, a dialog box appears to let the user select the file manually. **Note:** This parameter is only valid on certified documents where the user has sufficient permissions.

Returns

Empty

$host.language

This property defines the language of the running application.

Syntax

```
$host.language
```

Parameters

None

Returns

String

$host.messageBox

This method displays a dialog box on the screen.

Syntax

```
$host.messageBox( STRING param1 [, STRING param2 [, INTEGER param3
[, INTEGER param4 ] ] ] )
```

Parameters

Parameter	Description
param1	The message to display to the user.
param2 (*optional*)	Optional title to appear in the dialog box title bar.
param3 (*optional*)	The icon type. • 0 (default) - Error • 1 - Warning • 2 - Question • 3 - Status
param4 (*optional*)	The button type. • 1 (default) - OK • 2 - Cancel • 3 - No • 4 - Yes

Returns

Integer

$host.name

Returns the name of the host application. For example, on an interactive PDF form, this property returns `Acrobat`.

Syntax

```
$host.name
```

Parameters

None

Returns

String

$host.numPages

This property returns the number of pages in the current document.

Syntax

```
$host.numPages
```

Parameters

None

Returns

Integer

$host.pageDown()

Go to the next page.

Syntax

```
$host.pageDown()
```

Parameters

None

Returns

Empty

$host.pageUp()

Go to the previous page.

Syntax

```
$host.pageUp()
```

Parameters

None

Returns

Empty

$host.platform

This property returns the platform of the machine running the script. For example, in the case of a PDF form in Acrobat, this property returns: `WIN`, `MAC`, and `UNIX`.

Syntax

```
$host.platform
```

Parameters

None

Returns

String

$host.print

Prints all or the specific number of pages of the document.

Syntax

```
$host.print( BOOLEAN param1, INTEGER param2, INTEGER param3, BOOLEAN param4,
BOOLEAN param5, BOOLEAN param6, BOOLEAN param7, BOOLEAN param8 )
```

Parameters

Parameter	Description
param1	• 0 (default) Do not display a print dialog. • 1 Presents a print dialog to the user to obtain printing information and confirm the action.
param2	Page number of the beginning of the range to print. Page values are 0-based, so you represent page one with a value of 0.
param3	Page number of the end of the range to print. Page values are 0-based, so you represent page one with a value of 0.
param4	• 0 (default) Display a cancel dialog while the document prints. • 1 Do not display a cancel dialog while the document prints.
param5	• 0 (default) Do not shrink the page to fit the printable area. • 1 Shrink page to fit the printable area.
param6	• 0 (default) Do not print the page as a single image. • 1 Print the page as a single image.

Parameter	Description
param7	• 0 (default) Does not print pages from the value specified in *param2* to the value specified in *param3*. • 1 Prints pages from the value specified in *param2* to the value specified in *param3*.
param8	• 0 (default) Do not print the annotations. • 1 Print the annotations.

Returns

Empty

$host.resetData

This method resets field values on a form.

Syntax

```
$host.resetData([ STRING param ])
```

Parameters

Parameter	Description
param (*optional*)	Optional list (separated by commas) of the names of all fields to reset. If not present or empty, all the fields in the form are reset to their default value.

Returns

Empty

$host.response

This method displays a dialog box containing a question and an entry field for the user to reply to the question. The return value is a string containing the user's response. If the user presses the cancel button on the dialog box, the response is null.

Syntax

```
$host.response(STRING param1 [, STRING param2 [, STRING param3
[, BOOLEAN param4 ] ] ])
```

Parameters

Parameter	Description
param1	A question for the user.
param2 (*optional*)	Optional title to appear in the dialog title bar.
param3 (*optional*)	Default value for the answer to the question.
param4 (*optional*)	Indicates if the user's response should show as asterisks or bullets to mask the response. • 0 (default) Do not mask the user's answer. • 1 Mask the user's answer.

Returns

String

$host.setFocus

This method sets the keyboard focus to specified field.

Syntax

```
$host.setFocus( STRING param )
```

Parameters

Parameter	Description
param	Fully qualified SOM expression for the field to make the focus.

Returns

Empty

$host.title

This property is used to set and get the title of the document.

Syntax

```
$host.title
```

Parameters

None

Returns

String

$host.variation

This property indicates the packaging of the running application. For example, in the context of a PDF form in Adobe Reader, the property is one of: `Reader`, `Fill-in`, `Business Tools`, or `Full`.

Syntax

```
$host.variation
```

Parameters

None

Returns

Boolean

$host.version

This property indicates the version number of the current application. For example, in Acrobat 6.0.1 this property returns `6.0.1`.

Syntax

```
$host.version
```

Parameters

None

Returns

String

Comparing the host accessor functionality

This table illustrates the host accessor properties and methods and compares them to the equivalent expressions in Acrobat.

$host properties and methods	Acrobat equivalent
`$host.appType`	`app.viewerType`
`$host.beep( [ INTEGER param ] )`	`app.beep([ nType ])`
`$host.currentPage`	`doc.pageNum`
`$host.exportData([ STRING param1 [, BOOLEAN param2 ] ])`	`doc.exportXFAData(cPath [, bXDP ])`
`$host.gotoURL( STRING param1 [, BOOLEAN param2 ] )`	`doc.getURL(cURL, [ bAppend ])`
`$host.importData( [ STRING param ] )`	`doc.importXFAData(cPath)`
`$host.language`	`app.language`

$host properties and methods	Acrobat equivalent
$host.messageBox(STRING param1 [, STRING *param2* [, INTEGER *param3* [, INTEGER *param4*]]])	doc.alert(*cMsg* [, *nIcon* [, *nType* [, *cTitle*]]])
$host.name	none
$host.numPages	doc.numPages
$host.pageDown()	doc.pageNum++
$host.pageUp()	doc.pageNum--
$host.platform	app.platform
$host.print(*BOOLEAN param1*, INTEGER param2, INTEGER param3, BOOLEAN *param4*, BOOLEAN *param5*, BOOLEAN param6, BOOLEAN param7, BOOLEAN param8)	doc.print([*bUI* [, *nStart* [, *nEnd* [, *bSilent* [, *bShrinkToFit* [, *bPrintAsImage* [, *bReverse* [, *bAnnotations*]]]]]]]])
$host.resetData([STRING *param*])	doc.resetForm([*aFields*])
$host.response(STRING param1 [, STRING param2 [, STRING *param3* [, BOOLEAN param4]]])	app.response(*cQuestion* [, *cTitle* [, *cDefault* [, *bPassword*]]])
$host.setFocus(STRING *param*)	field.setFocus()
$host.title	doc.title
$host.variation	app.viewerVariation
$host.version	app.viewerVersion

See also

About the host accessor

Host accessor properties and methods

The event accessor

About the event accessor

The event accessor queries different object event properties. These properties are useful if you want to access values that are otherwise out of the scope of the events listed in the Show list within the Script Editor. For example, you can retrieve the full value of a field that would otherwise have part of the data stripped out because it is too long or otherwise invalid. This is useful in situations where you have to conduct extensive error checking.

Event accessor properties

$event.change

Specifies the value that a user types or pastes into a field immediately after they perform the action.

Syntax

```
$event.change
```

Returns

String

$event.commitKey

Determines the action that resulted in a field obtaining a value.

Syntax

```
$event.commitKey
```

Returns

Integer

$event.fullText

The full (untruncated) value of a field. The value of the newContentType property determines the content type of this property.

Syntax

```
$event.fullText
```

Returns

String

$event.keyDown

Determines if a user is pressing a keyboard key, including if the arrow key is used to make the selection. This is only available for list boxes and drop-down lists.

Syntax

```
$event.keyDown
```

$event.modifier

Specifies whether the modifier key (for example Ctrl for Microsoft Windows platforms) is down during a particular event.

Syntax

```
$event.modifier
```

Returns

Boolean

$event.name

Name of the current event.

Syntax

```
$event.name
```

Returns

String

$event.newContentType

Content type of the value specified for newText. For example, if prevContentType="text/html", then newText contains an XHTML fragment.

Syntax

```
$event.newContentType
```

Returns

String

$event.newText

Value after the field changes.

Syntax

```
$event.newtext
```

Returns

String

$event.prevContentType

Content type of the value specified for prevText. For example, if prevContentType="text/html", then prevText contains an XHTML fragment.

Syntax

```
$event.prevContentType
```

Returns

String

$event.prevText

Value before the field changes.

Syntax

```
$event.prevText
```

Returns

String

$event.selEnd

Specifies the ending position of the current text selection during a change event.

Syntax

```
$event.selEnd
```

Returns

Integer

$event.selStart

Specifies the starting position of the current text selection during a change event.

Syntax

```
$event.selStart
```

Returns

Integer

$event.shift

Specifies whether the shift key is down during a particular event.

Syntax

```
$event.shift
```

Returns

Boolean

$event.target

Specifies the target object responsible for triggering the event.

Syntax

```
$event.target
```

Returns

Object

12 The script object

About the script object

The script object is an object that you can use to store JavaScript functions and values separately from any particular form object. Typically you use the script object to create custom functions and methods that you want to use as part of scripts in many locations on your form. This technique reduce the overall amount of scripting required to perform repetitive actions.

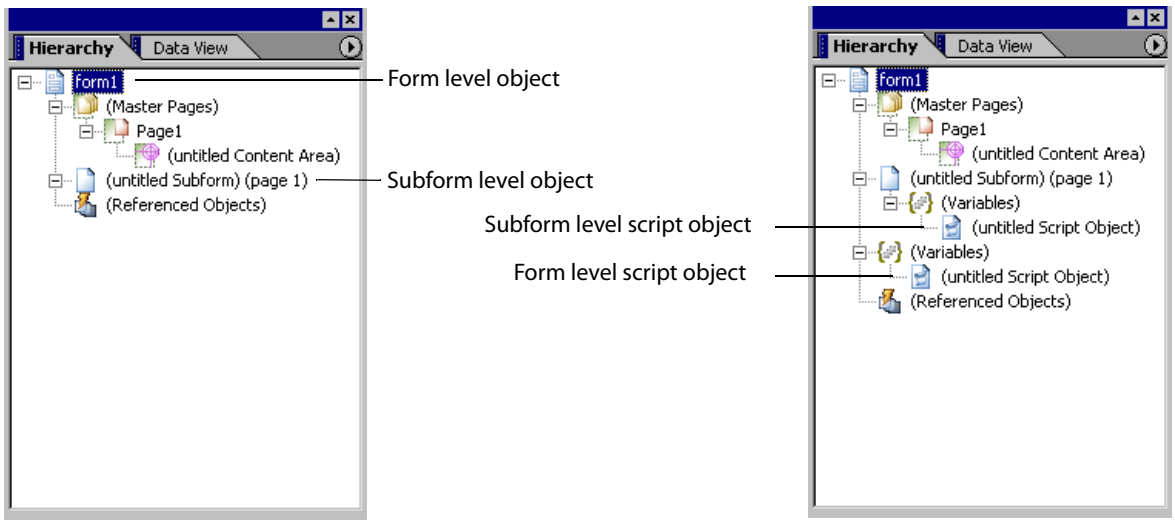
The script object only supports scripting written in JavaScript, but there are no restrictions on where execution of the scripts takes place. Both Acrobat and Form Server process scripting from a script object in the same manner, but both are also distinct. Only scripts set to run on the client can make use of script objects set to run on the client, and vice versa.

Creating a script object

There are two parts to creating a new script object. The first part involves adding the object itself to the form design, and the second part is the actual writing of the script you want to store in the script object.

➤ **To add a new script object to your form:**

1. Create a new form or open an existing form.
2. In the Hierarchy palette, right-click either a form-level object or a subform-level object and select Insert Script Object.



3. (Optional) Right-click the script object and select Rename Object.

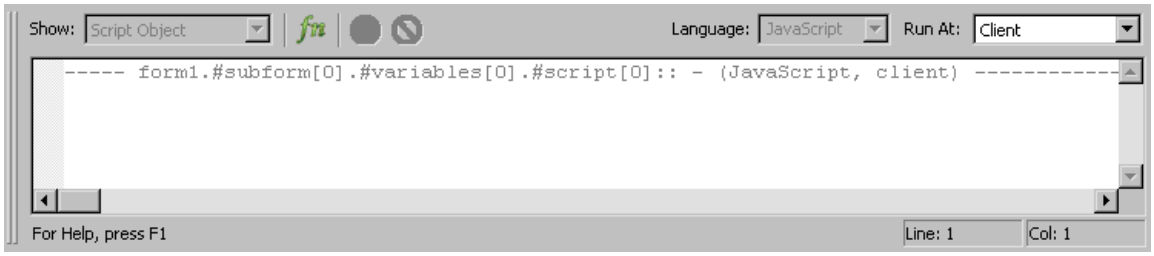
Adding JavaScript to a script object

Once you have a script object on your form, you can add scripts using the Script Editor.

➤ **To add script code to a script object:**

1. Select the script object in the Hierarchy palette.

   The Script Editor displays with both a Script Object value in the Show list, and a JavaScript value in the Language list. You cannot change either of these values.



2. Select where you would like your script object.
3. Enter your JavaScript in the Script Source field.
4. When finished, click Enter Script Source Changes  to add the script to your form.

13 Advanced scripting concepts

About the XML form object model

The Adobe XML form object model, based on the Adobe XML Forms Architecture, represents the underlying technology behind the Adobe XML form solution. This technology enables the construction of robust and flexible form-based applications for use with either the client or the server.

The Adobe XML Forms Architecture heavily leverages XML for the representation of all information and incorporates XML architectural concepts such as Document Object Model(s) (DOM).

The object model provides a form design-based approach to building forms that distinguishes between form layout and content. A form design defines presentation, calculations, and interaction rules. Content is the application data. Any format of XML data is acceptable. Though they are often packaged together, the form design and data are separate entities, and are handled separately by the object model.

Typically, forms are created using one of the following methods:

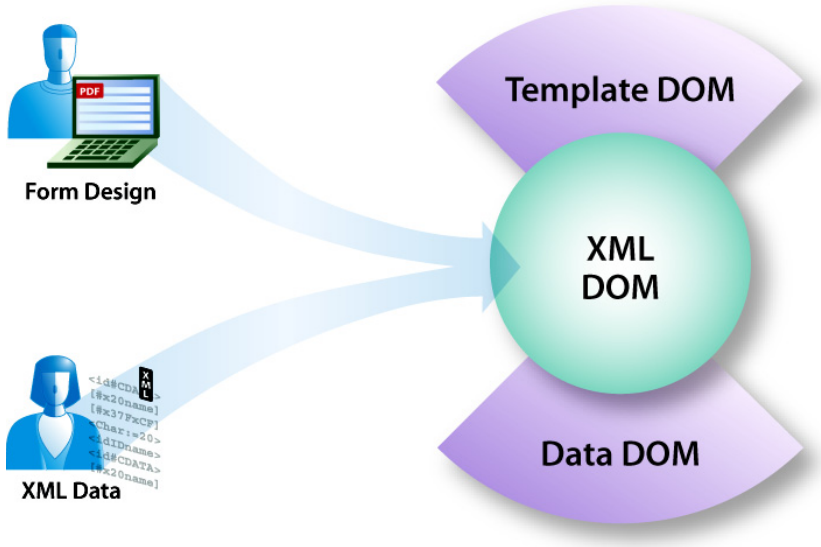
- Using Designer, the interactive graphical forms authoring tool.
- Automatically, through software generating a form based on some input, such as an XML schema.

Designer provides a set of tools that enables the form author to build intelligent business documents. The form author can, optionally, incorporate scripting to create a richer experience for the recipient of the form.

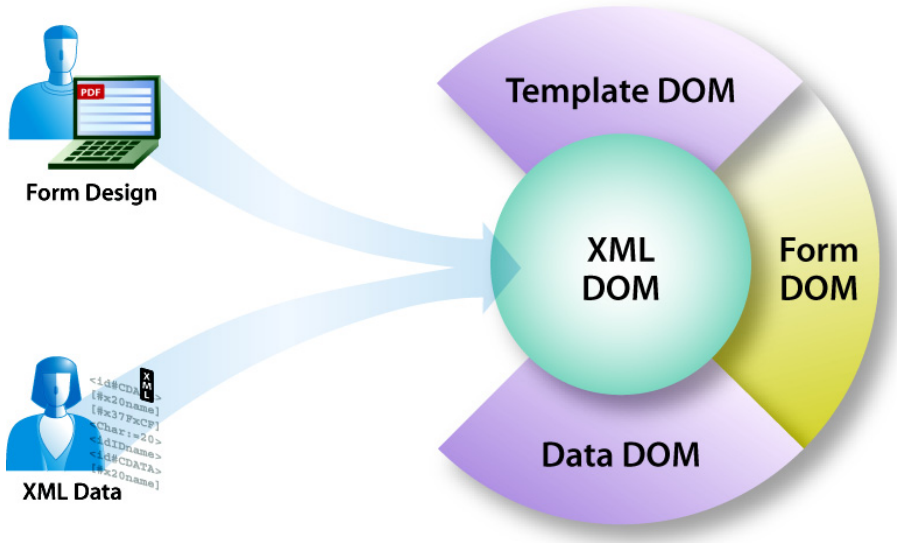
Understanding the XML form object model

Each time a form design is combined with data, the XML form object model is used to facilitate the process of combining template and data to create the resulting form. The process begins by taking the existing XML document object models (DOMs) representations of the form design and the XML data and creating separate DOMs. These separate DOMs store a structured representation of both the original form design

and the original XML data. The Template DOM corresponds to the form design, and the Data DOM corresponds to the user-supplied XML data. The diagram below illustrates this process.



Once creation of the Template and Data DOMs is complete, a third DOM, the Form DOM, is created that represents the merged information. The Form DOM acts as a medium for combining the specific values from the XML data with the presentation rules defined by the form design. The diagram below illustrates the creation of the Form DOM.

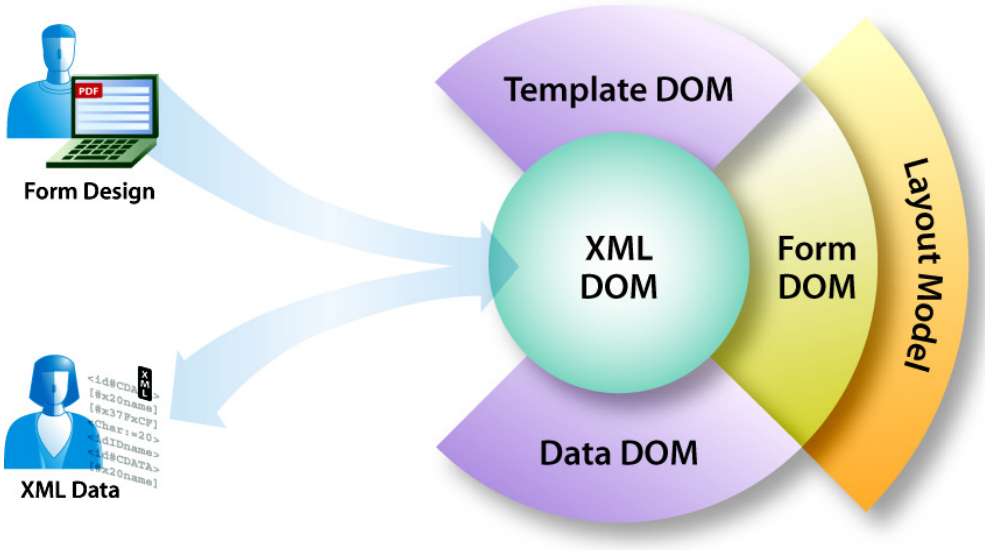


If you are creating an interactive form, it is at this point the form is complete and ready for deployment to users. Interactive form designs may have associated data with which they are merged, but most interactive forms are designed to allow for user-entered data entry.

Up to this point, static and dynamic forms follow the same set of processes that interactive forms do. However, static and dynamic forms always have a set of data to merge with their form templates. In the case of static forms, data merging does not impact the presentation rules for the form. That is, data is plugged into the appropriate fields without the fields themselves reacting to the data. In contrast,

dynamic form designs can make use of fields that will grow or shrink in response to the amount of data they are given.

The Form DOM for both static and dynamic forms looks very similar. It is essentially one long form that contains no pagination. Once the data and presentation rules have been applied to these types of forms, they must be formatted according to the desired layout information. A Layout DOM is created from the Form DOM that structures the form into pages, and applies any other page-based rules such as page numbering, headers, and trailers. The diagram below illustrates the creation of the Layout Model.

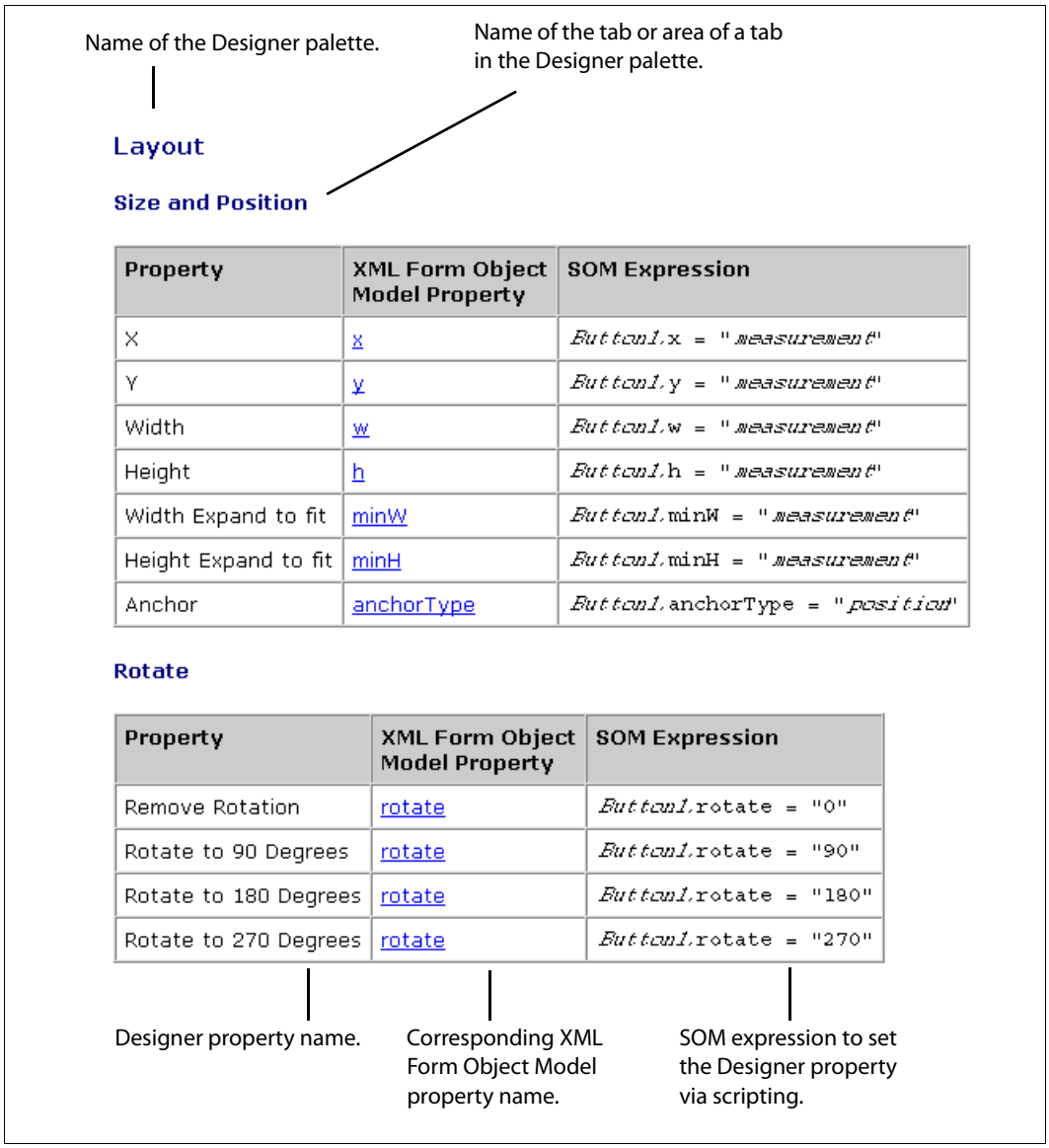


After application of the layout rules, both static and dynamic forms are complete.

Note: Dynamic and static form designs can contain some interactive features, similar to those typically found on an interactive form. In this way, dynamic forms can be thought of as a hybrid of both interactive and static forms.

14 About the Object Reference

The object reference provides the SOM scripting syntax for form object properties that you can manipulate using the Designer interface. The reference is broken down into sections according to the standard form design objects. The information for each form design object is categorized according to the palettes in Designer. For example, the diagram below displays information for the button object.



The Property column displays the object property name as it appears in the palette or tab in Designer.

The XML Form Object Model Property column lists the actual object model property that corresponds to the property name in Designer.

The SOM Expression column gives the scripting syntax you will use to retrieve or set a particular Designer property. Components of a SOM expression shown in italics represent areas that you must change according to your specific needs. For example, you would replace *Button1* in the example diagram

above with the name of the button on your form design. Similarly, you would replace *measurement* with a valid measurement value such as `0.006in`. To learn more about the acceptable values for each SOM expression, see Properties and Methods.

See also

Advanced scripting concepts

15 Barcodes

Layout

Size and Position

Property	XML Form Object Model Property	SOM Expression
X	x	*Barcode1*.x = "*measurement*"
Y	y	*Barcode1*.y = "*measurement*"
Width	w	*Barcode1*.w = "*measurement*"
Height	h	*Barcode1*.h = "*measurement*"
Width Expand to fit	minW	*Barcode1*.minW = "*measurement*"
Height Expand to fit	minH	*Barcode1*.minH = "*measurement*"
Anchor	anchorType	*Barcode1*.anchorType = "*position*"

Rotate

Property	XML Form Object Model Property	SOM Expression
Remove Rotation	rotate	*Barcode1*.rotate = "0"
Rotate to 90 Degrees	rotate	*Barcode1*.rotate = "90"
Rotate to 180 Degrees	rotate	*Barcode1*.rotate = "180"
Rotate to 270 Degrees	rotate	*Barcode1*.rotate = "270"

Margins

Property	XML Form Object Model Property	SOM Expression
Left	leftInset	*Barcode1*.margin.leftInset = "*measurement*"
Right	rightInset	*Barcode1*.margin.rightInset = "*measurement*"
Top	topInset	*Barcode1*.margin.topInset = "*measurement*"
Bottom	bottomInset	*Barcode1*.margin.bottomInset = "*measurement*"

Accessibility

Property	XML Form Object Model Property	SOM Expression
Tooltip	toolTip	*Barcode1*.assist.toolTip = "*text*"
Screen Reader Precedence	priority	*Barcode1*.assist.speak.priority = "*object*"
Custom Screen Reader Text	speak	*Barcode1*.assist.speak = "*text*"

Borders

Property	XML Form Object Model Property	SOM Expression
Left Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Right Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Top Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Bottom Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Rectangle Corner	join	*Barcode1*.border.corner.join = "square"
Inverted Rectangle Corner	inverted	*Barcode1*.border.corner.inverted = "1"
Round Corner	join	*Barcode1*.border.corner.join = "round"
Inverted Round Corner	inverted	*Barcode1*.border.corner.inverted = "1"
Inverted Corner Radius	radius	*Barcode1*.border.corner.radius = "*measurement*"

Background Fill

Property	XML Form Object Model Property	SOM Expression
None	presence	*Barcode1*.border.fill.presence = "invisible"
Solid	presence	*Barcode1*.border.fill.presence = "visible"

Property	XML Form Object Model Property	SOM Expression
Solid - Color 1	value	*Barcode1*.border.fill.color.value = "*R,G,B*"
Linear	type	*Barcode1*.border.fill.linear.type = "*direction*"
Linear - Color 1	value	*Barcode1*.border.fill.color.value = "*R,G,B*"
Linear - Color 2	value	*Barcode1*.border.fill.linear.color.value = "*R,G,B*"
Pattern	type	*Barcode1*.border.fill.pattern.type = "*direction*"
Pattern - Color 1	value	*Barcode1*.border.fill.color.value = "*R,G,B*"
Pattern - Color 2	value	*Barcode1*.border.fill.pattern.color.value = "*R,G,B*"
Radial	type	*Barcode1*.border.fill.radial.type = "*direction*"
Radial - Color 1	value	*Barcode1*.border.fill.color.value = "*R,G,B*"
Radial - Color 2	value	*Barcode1*.border.fill.radial.color.value = "*R,G,B*"

Object

Field

Property	XML Form Object Model Property	SOM Expression
Text	None rawValue	*Barcode1* = "*value*" (FormCalc) *Barcode1*.rawValue = "*value*" (JavaScript)
Location	textLocation	*Barcode1*.ui.#barcode.textLocation = "*location*"
Data length	dataLength	*Barcode1*.ui.#barcode.dataLength = "*length*"
Checksum	checksum	*Barcode1*.ui.#barcode.checksum = "*checksum*"
Wide/narrow ratio	N/A	You cannot change the wide/narrow ratio value for barcodes.
Presence	presence	*Barcode1*.presence = "*visibility*"
Locale	locale	*Barcode1*.locale = "*locale*"

Binding

Property	XML Form Object Model Property	SOM Expression
Name	name	*Barcode1*.name = "*objectname*"
Default Binding (Open, Save, Submit)	match	*Barcode1*.bind.match = "*condition*"
Import/Export Bindings (Execute)	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.

16 Button

Layout

Size and Position

Property	XML Form Object Model Property	SOM Expression
X	x	*Button1*.x = "*measurement*"
Y	y	*Button1*.y = "*measurement*"
Width	w	*Button1*.w = "*measurement*"
Height	h	*Button1*.h = "*measurement*"
Width Expand to fit	minW	*Button1*.minW = "*measurement*"
Height Expand to fit	minH	*Button1*.minH = "*measurement*"
Anchor	anchorType	*Button1*.anchorType = "*position*"

Rotate

Property	XML Form Object Model Property	SOM Expression
Remove Rotation	rotate	*Button1*.rotate = "0"
Rotate to 90 Degrees	rotate	*Button1*.rotate = "90"
Rotate to 180 Degrees	rotate	*Button1*.rotate = "180"
Rotate to 270 Degrees	rotate	*Button1*.rotate = "270"

Margins

Property	XML Form Object Model Property	SOM Expression
Left	leftInset	*Button1*.margin.leftInset = "*measurement*"
Right	rightInset	*Button1*.margin.rightInset = "*measurement*"
Top	topInset	*Button1*.margin.topInset = "*measurement*"
Bottom	bottomInset	*Button1*.margin.bottomInset = "*measurement*"

Caption

Property	XML Form Object Model Property	SOM Expression
Value	#text	*Button1*.caption.value.#text = "*text*"
Position	N/A	Disabled for this form object.
Reserve	N/A	Disabled for this form object.

Font

For button objects, the Font palette only applies to the caption of the button.

Property	XML Form Object Model Property	SOM Expression
Font	typeface	*Button1*.caption.font.typeface = "*font name*"
Size	size	*Button1*.caption.font.size = "*font size*"
Baseline Shift	baselineShift	*Button1*.caption.font.baselineShift = "*measurement*"

Style

Property	XML Form Object Model Property	SOM Expression
Bold	weight	*Button1*.caption.font.weight = "*measurement*"
Italic	posture	*Button1*.caption.font.posture = "*posture*"
Underline	underline	*Button1*.caption.font.underline = "*underline*"
Strikethrough	lineThrough	*Button1*.caption.font.lineThrough = "*linethrough*"
Color	value	*Button1*.caption.font.fill.color.value = "*R,G,B*"

Accessibility

Property	XML Form Object Model Property	SOM Expression
Tooltip	toolTip	*Button1*.assist.toolTip = "*text*"
Screen Reader Precedence	priority	*Button1*.assist.speak.priority = "*object*"
Custom Screen Reader Text	speak	*Button1*.assist.speak = "*text*"

Borders

Property	XML Form Object Model Property	SOM Expression
Left Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Right Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Top Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Bottom Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Rectangle Corner	join	`Button1.border.corner.join = "square"`
Inverted Rectangle Corner	inverted	`Button1.border.corner.inverted = "1"`
Round Corner	join	`Button1.border.corner.join = "round"`
Inverted Round Corner	inverted	`Button1.border.corner.inverted = "1"`
Inverted Corner Radius	radius	`Button1.border.corner.radius = "measurement"`

Background Fill

Property	XML Form Object Model Property	SOM Expression
None	presence	`Button1.border.fill.presence = "invisible"`
Solid	presence	`Button1.border.fill.presence = "visible"`
Solid - Color 1	value	`Button1.border.fill.color.value = "R,G,B"`
Linear	type	`Button1.border.fill.linear.type = "direction"`
Linear - Color 1	value	`Button1.border.fill.color.value = "R,G,B"`
Linear - Color 2	value	`Button1.border.fill.linear.color.value = "R,G,B"`
Pattern	type	`Button1.border.fill.pattern.type = "direction"`

Property	XML Form Object Model Property	SOM Expression
Pattern - Color 1	value	*Button1*.border.fill.color.value = "*R,G,B*"
Pattern - Color 2	value	*Button1*.border.fill.pattern.color.value = "*R,G,B*"
Radial	type	*Button1*.border.fill.radial.type = "*direction*"
Radial - Color 1	value	*Button1*.border.fill.color.value = "*R,G,B*"
Radial - Color 2	value	*Button1*.border.fill.radial.color.value = "*R,G,B*"

Paragraph

For button objects, the Paragraph palette only applies to the caption of the button.

Property	XML Form Object Model Property	SOM Expression
Align Left	hAlign	*Button1*.caption.para.hAlign = "left"
Align Center	hAlign	*Button1*.caption.para.hAlign = "center"
Align Right	hAlign	*Button1*.caption.para.hAlign = "right"
Justify	hAlign	*Button1*.caption.para.hAlign = "justify"
Align Top	vAlign	*Button1*.caption.para.vAlign = "top"
Align Middle	vAlign	*Button1*.caption.para.vAlign = "middle"
Align Bottom	vAlign	*Button1*.caption.para.vAlign = "bottom"
Indent Left	marginLeft	*Button1*.caption.para.marginLeft = "*measurement*"
Indent Right	marginRight	*Button1*.caption.para.marginRight = "*measurement*"
Indent First	textIndent	*Button1*.caption.para.textIndent = "*measurement*"
Indent By	textIndent	*Button1*.caption.para.textIndent = "*measurement*"
Spacing Above	spaceAbove	*Button1*.caption.para.spacingAbove = "*measurement*"
Spacing Below	spaceBelow	*Button1*.caption.para.spacingBelow = "*measurement*"
Line Spacing	lineHeight	*Button1*.caption.para.lineHeight = "*measurement*"

Object

Field

Property	XML Form Object Model Property	SOM Expression
Appearance	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Regular	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Submit	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Execute	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Presence	presence	`Button1.presence = "visibility"`
Locale	locale	`Button1.locale = "locale"`

Submit

Property	XML Form Object Model Property	SOM Expression
Submit Format	format	`Button1.event.submit.format = "dataformat"`
Submit to URL	target	`Button1.event.submit.target = "URL"`

Include

Property	XML Form Object Model Property	SOM Expression
Annotations	xdpContent	`Button1.event.submit.xdpContent = "xfdf"`
Digital Signatures	xdpContent	`Button1.event.submit.xdpContent = "signature"`
PDF	xdpContent	`Button1.event.submit.embedPDF = "1"`
Template	xdpContent	`Button1.event.submit.xdpContent = "template"`
Other	xdpContent	`Button1.event.submit.xdpContent = "contentname"`

Data Options

Property	XML Form Object Model Property	SOM Expression
Encoding	textEncoding	*Button1*.event.submit.textEncoding = "encoding"

Execute

Execution Options

Property	XML Form Object Model Property	SOM Expression
Run At	runAt	*Button1*.event.execute.runAt = "*runAtlocation*"
Re-merge Form Data	executeType	*Button1*.event.execute.executeType = "*type*"

17 Check Box

Layout

Size and Position

Property	XML Form Object Model Property	SOM Expression
X	x	*CheckBox1*.x = "*measurement*"
Y	y	*CheckBox1*.y = "*measurement*"
Width	w	*CheckBox1*.w = "*measurement*"
Height	h	*CheckBox1*.h = "*measurement*"
Width Expand to fit	minW	*CheckBox1*.minW = "*measurement*"
Height Expand to fit	minH	*CheckBox1*.minH = "*measurement*"
Anchor	anchorType	*CheckBox1*.anchorType = "*position*"

Rotate

Property	XML Form Object Model Property	SOM Expression
Remove Rotation	rotate	*CheckBox1*.rotate = "0"
Rotate to 90 Degrees	rotate	*CheckBox1*.rotate = "90"
Rotate to 180 Degrees	rotate	*CheckBox1*.rotate = "180"
Rotate to 270 Degrees	rotate	*CheckBox1*.rotate = "270"

Margins

Property	XML Form Object Model Property	SOM Expression
Left	leftInset	*CheckBox1*.margin.leftInset = "*measurement*"
Right	rightInset	*CheckBox1*.margin.rightInset = "*measurement*"
Top	topInset	*CheckBox1*.margin.topInset = "*measurement*"
Bottom	bottomInset	*CheckBox1*.margin.bottomInset = "*measurement*"

Caption

Property	XML Form Object Model Property	SOM Expression
Value	#text	*CheckBox1*.caption.value.#text = "*text*"
Position	placement	*CheckBox1*.caption.placement = "*placement*"
Reserve	reserve	*CheckBox1*.caption.reserve = "*alignment*"

Font

For check box objects, the Font palette only applies to the caption of the check box.

Property	XML Form Object Model Property	SOM Expression
Font	typeface	*CheckBox1*.caption.font.typeface = "*font name*"
Size	size	*CheckBox1*.caption.font.size = "*font size*"
Baseline Shift	baselineShift	*CheckBox1*.caption.font.baselineShift = "*measurement*"

Style

Property	XML Form Object Model Property	SOM Expression
Bold	weight	*CheckBox1*.caption.font.weight = "*measurement*"
Italic	posture	*CheckBox1*.caption.font.posture = "*posture*"
Underline	underline	*CheckBox1*.caption.font.underline = "*underline*"
Strikethrough	lineThrough	*CheckBox1*.caption.font.lineThrough = "*linethrough*"
Color	value	*CheckBox1*.caption.font.fill.color.value = "*R,G,B*"

Accessibility

Property	XML Form Object Model Property	SOM Expression
Tooltip	toolTip	*CheckBox1*.assist.toolTip = "*text*"

Property	XML Form Object Model Property	SOM Expression
Screen Reader Precedence	priority	*CheckBox1*.assist.speak.priority = "*object*"
Custom Screen Reader text	speak	*CheckBox1*.assist.speak = "*text*"

Borders

Property	XML Form Object Model Property	SOM Expression
Left Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Right Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Top Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Bottom Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Rectangle Corner	join	*CheckBox1*.border.corner.join = "square"
Inverted Rectangle Corner	inverted	*CheckBox1*.border.corner.inverted = "1"
Round Corner	join	*CheckBox1*.border.corner.join = "round"
Inverted Round Corner	inverted	*CheckBox1*.border.corner.inverted = "1"
Inverted Corner Radius	radius	*CheckBox1*.border.corner.radius = "*measurement*"

Background Fill

Property	XML Form Object Model Property	SOM Expression
None	presence	*CheckBox1*.border.fill.presence = "invisible"
Solid	presence	*CheckBox1*.border.fill.presence = "visible"

Property	XML Form Object Model Property	SOM Expression
Solid - Color 1	value	*CheckBox1*.border.fill.color.value = "*R,G,B*"
Linear	type	*CheckBox1*.border.fill.linear.type = "*direction*"
Linear - Color 1	value	*CheckBox1*.border.fill.color.value = "*R,G,B*"
Linear - Color 2	value	*CheckBox1*.border.fill.linear.color.value = "*R,G,B*"
Pattern	type	*CheckBox1*.border.fill.pattern.type = "*direction*"
Pattern - Color 1	value	*CheckBox1*.border.fill.color.value = "*R,G,B*"
Pattern - Color 2	value	*CheckBox1*.border.fill.pattern.color.value = "*R,G,B*"
Radial	type	*CheckBox1*.border.fill.radial.type = "*direction*"
Radial - Color 1	value	*CheckBox1*.border.fill.color.value = "*R,G,B*"
Radial - Color 2	value	*CheckBox1*.border.fill.radial.color.value = "*R,G,B*"

Paragraph

For check box objects, the Paragraph palette only applies to the caption of the check box.

Property	XML Form Object Model Property	SOM Expression
Align Left	hAlign	*CheckBox1*.caption.para.hAlign = "left"
Align Center	hAlign	*CheckBox1*.caption.para.hAlign = "center"
Align Right	hAlign	*CheckBox1*.caption.para.hAlign = "right"
Justify	hAlign	*CheckBox1*.caption.para.hAlign = "justify"
Align Top	vAlign	*CheckBox1*.caption.para.vAlign = "top"
Align Middle	vAlign	*CheckBox1*.caption.para.vAlign = "middle"
Align Bottom	vAlign	*CheckBox1*.caption.para.vAlign = "bottom"
Indent Left	marginLeft	*CheckBox1*.caption.para.marginLeft = "*measurement*"
Indent Right	marginRight	*CheckBox1*.caption.para.marginRight = "*measurement*"

Property	XML Form Object Model Property	SOM Expression
Indent First	textIndent	*CheckBox1*.caption.para.textIndent = "*measurement*"
Indent By	textIndent	*CheckBox1*.caption.para.textIndent = "*measurement*"
Spacing Above	spaceAbove	*CheckBox1*.caption.para.spacingAbove = "*measurement*"
Spacing Below	spaceBelow	*CheckBox1*.caption.para.spacingBelow = "*measurement*"
Line Spacing	lineHeight	*CheckBox1*.caption.para.lineHeight = "*measurement*"

Object

Field

Property	XML Form Object Model Property	SOM Expression
Appearance	shape	*CheckBox1*.ui.checkButton.shape = "*shape*"
States	allowNeutral	*CheckBox1*.ui.checkButton.allowNeutral = "*boolean*"
Size	size	*CheckBox1*.ui.checkButton.size = "*measurement*"
Presence	presence	*CheckBox1*.presence = "*visibility*"
Locale	locale	*CheckBox1*.locale = "*locale*"

Value

Property	XML Form Object Model Property	SOM Expression
Type	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Value	None rawValue	*CheckBox1* = "*integer*" (FormCalc) *CheckBox1*.rawValue = "*integer*" (JavaScript)
Override Message	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.

Property	XML Form Object Model Property	SOM Expression
Validation Script Message	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Error	scriptTest	*CheckBox1*.validate.scriptTest = " "

Binding

Property	XML Form Object Model Property	SOM Expression
Name	name	*CheckBox1*.name = "*objectname*"
Default Binding (Open, Save, Submit)	match	*CheckBox1*.bind.match = "*condition*"
On Value	N/A	You cannot change this property using scripting.
Off Value	N/A	You cannot change this property using scripting.
Neutral Value	N/A	You cannot change this property using scripting.
Import/Export Bindings (Execute)	N/A	You cannot change this property using scripting.

18 Circle

Layout

Size and Position

Property	XML Form Object Model Property	SOM Expression
X	x	*Circle1*.x = "*measurement*"
Y	y	*Circle1*.y = "*measurement*"
Width	w	*Circle1*.w = "*measurement*"
Height	h	*Circle1*.h = "*measurement*"
Width Expand to fit	minW	*Circle1*.minW = "*measurement*"
Height Expand to fit	minH	*Circle1*.minH = "*measurement*"
Anchor	anchorType	*Circle1*.anchorType = "*position*"

Rotate

Property	XML Form Object Model Property	SOM Expression
Remove Rotation	rotate	*Circle1*.rotate = "0"
Rotate to 90 Degrees	rotate	*Circle1*.rotate = "90"
Rotate to 180 Degrees	rotate	*Circle1*.rotate = "180"
Rotate to 270 Degrees	rotate	*Circle1*.rotate = "270"

Margins

Property	XML Form Object Model Property	SOM Expression
Left	leftInset	*Circle1*.margin.leftInset = "*measurement*"
Right	rightInset	*Circle1*.margin.rightInset = "*measurement*"
Top	topInset	*Circle1*.margin.topInset = "*measurement*"
Bottom	bottomInset	*Circle1*.margin.bottomInset = "*measurement*"

Object

Draw

Appearance

Property	XML Form Object Model Property	SOM Expression
Ellipse	circular	*Circle1*.value.arc.circular = "0"
Circle	circular	*Circle1*.value.arc.circular = "1"
Arc	N/A	See the Start and Sweep properties.
Start	startAngle	*Circle1*.value.arc.startAngle = "*angle*"
Sweep	sweepAngle	*Circle1*.value.arc.sweepAngle = "*angle*"
Line Style	stroke	*Circle1*.value.arc.edge.stroke = "*appearance*"
Line Thickness	thickness	*Circle1*.value.arc.edge.thickness = "*thickness*"
Line Color	value	*Circle1*.value.arc.edge.color.value = "*R,G,B*"

Fill style and color

Property	XML Form Object Model Property	SOM Expression
None	presence	*Circle1*.border.fill.presence = "invisible"
Solid	presence	*Circle1*.border.fill.presence = "visible"
Solid - Color 1	value	*Circle1*.border.fill.color.value = "*R,G,B*"
Linear	type	*Circle1*.border.fill.linear.type = "*direction*"
Linear - Color 1	value	*Circle1*.border.fill.color.value = "*R,G,B*"
Linear - Color 2	value	*Circle1*.border.fill.linear.color.value = "*R,G,B*"
Pattern	type	*Circle1*.border.fill.pattern.type = "*direction*"
Pattern - Color 1	value	*Circle1*.border.fill.color.value = "*R,G,B*"
Pattern - Color 2	value	*Circle1*.border.fill.pattern.color.value = "*R,G,B*"
Radial	type	*Circle1*.border.fill.radial.type = "*direction*"
Radial - Color 1	value	*Circle1*.border.fill.color.value = "*R,G,B*"
Radial - Color 2	value	*Circle1*.border.fill.radial.color.value = "*R,G,B*"

Presence

Property	XML Form Object Model Property	SOM Expression
Presence	presence	*Circle1*.presence = "*visibility*"

19 Content Area

Layout

Size and Position

Property	XML Form Object Model Property	SOM Expression
X	x	*ContentArea1*.x = "*measurement*"
Y	y	*ContentArea1*.y = "*measurement*"
Width	w	*ContentArea1*.w = "*measurement*"
Height	h	*ContentArea1*.h = "*measurement*"

Object

Content Area

Property	XML Form Object Model Property	SOM Expression
Name	name	*ContentArea1*.name = "*objectname*"
Flow Control	N/A	This Designer property corresponds to multiple XML Form Object Model properties. You cannot change this property using scripting.

20 Date/Time Field

Layout

Size and Position

Property	XML Form Object Model Property	SOM Expression
X	x	*DateTimeField1*.x = "*measurement*"
Y	y	*DateTimeField1*.y = "*measurement*"
Width	w	*DateTimeField1*.w = "*measurement*"
Height	h	*DateTimeField1*.h = "*measurement*"
Width Expand to fit	minW	*DateTimeField1*.minW = "*measurement*"
Height Expand to fit	minH	*DateTimeField1*.minH = "*measurement*"
Anchor	anchorType	*DateTimeField1*.anchorType = "*position*"

Rotate

Property	XML Form Object Model Property	SOM Expression
Remove Rotation	rotate	*DateTimeField1*.rotate = "0"
Rotate to 90 Degrees	rotate	*DateTimeField1*.rotate = "90"
Rotate to 180 Degrees	rotate	*DateTimeField1*.rotate = "180"
Rotate to 270 Degrees	rotate	*DateTimeField1*.rotate = "270"

Margins

Property	XML Form Object Model Property	SOM Expression
Left	leftInset	*DateTimeField1*.margin.leftInset = "*measurement*"
Right	rightInset	*DateTimeField1*.margin.rightInset = "*measurement*"
Top	topInset	*DateTimeField1*.margin.topInset = "*measurement*"
Bottom	bottomInset	*DateTimeField1*.margin.bottomInset = "*measurement*"

Caption

Property	XML Form Object Model Property	SOM Expression
Value	#text	*DateTimeField1*.caption.value.#text = "*text*"
Position	placement	*DateTimeField1*.caption.placement = "*placement*"
Reserve	reserve	*DateTimeField1*.caption.reserve = "*measurement*"

Font

For date/time field objects, the Font palette applies to either the caption or the value of the field, or both.

Property	XML Form Object Model Property	SOM Expression
Font	typeface	*DateTimeField1*.caption.font.typeface = "*font name*" *DateTimeField1*.font.typeface = "*font name*"
Size	size	*DateTimeField1*.caption.font.size = "*font size*" *DateTimeField1*.font.size = "*font size*"
Baseline Shift	baselineShift	*DateTimeField1*.caption.font.baselineShift = "*measurement*" *DateTimeField1*.font.baselineShift = "*measurement*"

Style

Property	XML Form Object Model Property	SOM Expression
Bold	weight	*DateTimeField1*.caption.font.weight = "*measurement*" *DateTimeField1*.font.weight = "*measurement*"
Italic	posture	*DateTimeField1*.caption.font.posture = "*posture*" *DateTimeField1*.font.posture = "*posture*"
Underline	underline	*DateTimeField1*.caption.font.underline = "*underline*" *DateTimeField1*.font.underline = "*underline*"
Strikethrough	lineThrough	*DateTimeField1*.caption.font.lineThrough = "*linethrough*" *DateTimeField1*.font.lineThrough = "*linethrough*"
Color	value	*DateTimeField1*.caption.font.fill.color.value = "*R,G,B*" *DateTimeField1*.font.fill.color.value = "*R,G,B*"

Accessibility

Property	XML Form Object Model Property	SOM Expression
Tooltip	toolTip	*DateTimeField1*.assist.toolTip = "*text*"
Screen Reader Precedence	priority	*DateTimeField1*.assist.speak.priority = "*object*"
Custom Screen Reader Text	speak	*DateTimeField1*.assist.speak = "*text*"

Borders

Property	XML Form Object Model Property	SOM Expression
Left Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Right Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Top Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Bottom Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Rectangle Corner	join	*DateTimeField1*.border.corner.join = "square"
Inverted Rectangle Corner	inverted	*DateTimeField1*.border.corner.inverted = "1"
Round Corner	join	*DateTimeField1*.border.corner.join = "round"
Inverted Round Corner	inverted	*DateTimeField1*.border.corner.inverted = "1"
Inverted Corner Radius	radius	*DateTimeField1*.border.corner.radius = "*measurement*"

Background Fill

Property	XML Form Object Model Property	SOM Expression
None	presence	*DateTimeField1*.border.fill.presence = "invisible"
Solid	presence	*DateTimeField1*.border.fill.presence = "visible"
Solid - Color 1	value	*DateTimeField1*.border.fill.color.value = "*R,G,B*"
Linear	type	*DateTimeField1*.border.fill.linear.type = "*direction*"
Linear - Color 1	value	*DateTimeField1*.border.fill.color.value = "*R,G,B*"
Linear - Color 2	value	*DateTimeField1*.border.fill.linear.color.value = "*R,G,B*"
Pattern	type	*DateTimeField1*.border.fill.pattern.type = "*direction*"
Pattern - Color 1	value	*DateTimeField1*.border.fill.color.value = "*R,G,B*"
Pattern - Color 2	value	*DateTimeField1*.border.fill.pattern.color.value = "*R,G,B*"
Radial	type	*DateTimeField1*.border.fill.radial.type = "*direction*"
Radial - Color 1	value	*DateTimeField1*.border.fill.color.value = "*R,G,B*"
Radial - Color 2	value	*DateTimeField1*.border.fill.radial.color.value = "*R,G,B*"

Paragraph

For date/time field objects, the Paragraph palette applies to either the caption or the value of the field, or both.

Property	XML Form Object Model Property	SOM Expression
Align left	hAlign	*DateTimeField1*.caption.para.hAlign = "left" *DateTimeField1*.para.hAlign = "left"
Align center	hAlign	*DateTimeField1*.caption.para.hAlign = "center" *DateTimeField1*.para.hAlign = "center"
Align right	hAlign	*DateTimeField1*.caption.para.hAlign = "right" *DateTimeField1*.para.hAlign = "right"

Property	XML Form Object Model Property	SOM Expression
Align Left	hAlign	*DateTimeField1*.caption.para.hAlign = "justify" *DateTimeField1*.para.hAlign = "justify"
Align Center	vAlign	*DateTimeField1*.caption.para.vAlign = "top" *DateTimeField1*.para.vAlign = "top"
Align Right	vAlign	*DateTimeField1*.caption.para.vAlign = "middle" *DateTimeField1*.para.vAlign = "middle"
Justify	vAlign	*DateTimeField1*.caption.para.vAlign = "bottom" *DateTimeField1*.para.vAlign = "bottom"
Align Top	marginLeft	*DateTimeField1*.caption.para.marginLeft = "*measurement*" *DateTimeField1*.para.marginLeft = "*measurement*"
Align Middle	marginRight	*DateTimeField1*.caption.para.marginRight = "*measurement*" *DateTimeField1*.para.marginRight = "*measurement*"
Align Bottom	textIndent	*DateTimeField1*.caption.para.textIndent = "*measurement*" *DateTimeField1*.para.textIndent = "*measurement*"
Indent Left	textIndent	*DateTimeField1*.caption.para.textIndent = "*measurement*" *DateTimeField1*.para.textIndent = "*measurement*"
Indent Right	spaceAbove	*DateTimeField1*.caption.para.spacingAbove = "*measurement*" *DateTimeField1*.para.spacingAbove = "*measurement*"
Indent First	spaceBelow	*DateTimeField1*.caption.para.spacingBelow = "*measurement*" *DateTimeField1*.para.spacingBelow = "*measurement*"
Indent By	lineHeight	*DateTimeField1*.caption.para.lineHeight = "*measurement*" *DateTimeField1*.para.lineHeight = "*measurement*"

Object

Field

Property	XML Form Object Model Property	SOM Expression
Appearance	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Display Pattern	picture	*DateTimeField1*.format.picture = "*pictureformat*"
Edit Pattern	picture	*DateTimeField1*.ui.picture = "*pictureformat*"
Presence	presence	*DateTimeField1*.presence = "*visibility*"
Locale	locale	*DateTimeField1*.locale = "*locale*"

Value

Property	XML Form Object Model Property	SOM Expression
Default	None rawValue	*DateTimeField1* = "*value*" (FormCalc) *DateTimeField1*.rawValue = "*value*" (JavaScript)
Empty Message	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Validation Pattern	picture	*DateTimeField1*.validate.picture = "*pictureformat*"
Validation Pattern Message	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Validation Pattern Message - Error	formatTest	*DateTimeField1*.validate.formatTest = "*test*"
Validation Script Message	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Validation Script Message - Error	scriptTest	*DateTimeField1*.validate.scriptTest = "*test*"

Binding

Property	XML Form Object Model Property	SOM Expression
Name	name	*DateTimeField1*.name = "*objectname*"
Default Binding (Open, Save, Submit)	match	*DateTimeField1*.bind.match = "*condition*"
Data Pattern	picture	*DateTimeField1*.bind.picture = "*pictureformat*"
Data Format	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Import/Export Bindings (Execute)	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.

21 Drop-down List

Layout

Size and Position

Property	XML Form Object Model Property	SOM Expression
X	x	*DropDownList1*.x = "*measurement*"
Y	y	*DropDownList1*.y = "*measurement*"
Width	w	*DropDownList1*.w = "*measurement*"
Height	h	*DropDownList1*.h = "*measurement*"
Anchor	anchorType	*DropDownList1*.anchorType = "*position*"

Rotate

Property	XML Form Object Model Property	SOM Expression
Remove Rotation	rotate	*DropDownList1*.rotate = "0"
Rotate to 90 Degrees	rotate	*DropDownList1*.rotate = "90"
Rotate to 180 Degrees	rotate	*DropDownList1*.rotate = "180"
Rotate to 270 Degrees	rotate	*DropDownList1*.rotate = "270"

Margins

Property	XML Form Object Model Property	SOM Expression
Left	leftInset	*DropDownList1*.margin.leftInset = "*measurement*"
Right	rightInset	*DropDownList1*.margin.rightInset = "*measurement*"
Top	topInset	*DropDownList1*.margin.topInset = "*measurement*"
Bottom	bottomInset	*DropDownList1*.margin.bottomInset = "*measurement*"

Caption

Property	XML Form Object Model Property	SOM Expression
Value	#text	*DropDownList1*.caption.value.#text = "*text*"
Position	placement	*DropDownList1*.caption.placement = "*placement*"
Reserve	reserve	*DropDownList1*.caption.reserve = "*measurement*"

Font

For drop-down list objects, the Font palette applies to either the caption or the value of the field, or both.

Property	XML Form Object Model Property	SOM Expression
Font	typeface	*DropDownList1*.caption.font.typeface = "*font name*" *DropDownList1*.font.typeface = "*font name*"
Size	size	*DropDownList1*.caption.font.size = "*font size*" *DropDownList1*.font.size = "*font size*"
Baseline Shift	baselineShift	*DropDownList1*.caption.font.baselineShift = "*measurement*" *DropDownList1*.font.baselineShift = "*measurement*"

Style

Property	XML Form Object Model Property	SOM Expression
Bold	weight	*DropDownList1*.caption.font.weight = "*measurement*" *DropDownList1*.font.weight = "*measurement*"
Italic	posture	*DropDownList1*.caption.font.posture = "*posture*" *DropDownList1*.font.posture = "*posture*"
Underline	underline	*DropDownList1*.caption.font.underline = "*underline*" *DropDownList1*.font.underline = "*underline*"
Strikethrough	lineThrough	*DropDownList1*.caption.font.lineThrough = "*linethrough*" *DropDownList1*.font.lineThrough = "*linethrough*"
Color	value	*DropDownList1*.caption.font.fill.color.value = "*R,G,B*" *DropDownList1*.font.fill.color.value = "*R,G,B*"

Accessibility

Property	XML Form Object Model Property	SOM Expression
Tooltip	toolTip	*DropDownList1*.assist.toolTip = " "
Screen Reader Precedence	priority	*DropDownList1*.assist.speak.priority = " "
Custom Screen Reader Text	speak	*DropDownList1*.assist.speak = " "

Borders

Property	XML Form Object Model Property	SOM Expression
Left Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Right Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Top Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Bottom Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Rectangle Corner	join	*DropDownList1*.border.corner.join = "square"
Inverted Rectangle Corner	inverted	*DropDownList1*.border.corner.inverted = "1"
Round Corner	join	*DropDownList1*.border.corner.join = "round"
Inverted Round Corner	inverted	*DropDownList1*.border.corner.inverted = "1"
Inverted Corner Radius	radius	*DropDownList1*.border.corner.radius = "*measurement*"

Background Fill

Property	XML Form Object Model Property	SOM Expression
None	presence	*DropDownList1*.border.fill.presence = "invisible"
Solid	presence	*DropDownList1*.border.fill.presence = "visible"
Solid - Color 1	value	*DropDownList1*.border.fill.color.value = "*R,G,B*"
Linear	type	*DropDownList1*.border.fill.linear.type = "*direction*"
Linear - Color 1	value	*DropDownList1*.border.fill.color.value = "*R,G,B*"
Linear - Color 2	value	*DropDownList1*.border.fill.linear.color.value = "*R,G,B*"
Pattern	type	*DropDownList1*.border.fill.pattern.type = "*direction*"
Pattern - Color 1	value	*DropDownList1*.border.fill.color.value = "*R,G,B*"
Pattern - Color 2	value	*DropDownList1*.border.fill.pattern.color.value = "*R,G,B*"
Radial	type	*DropDownList1*.border.fill.radial.type = "*direction*"
Radial - Color 1	value	*DropDownList1*.border.fill.color.value = "*R,G,B*"
Radial - Color 2	value	*DropDownList1*.border.fill.radial.color.value = "*R,G,B*"

Paragraph

For drop-down list objects, the Paragraph palette applies to either the caption or the value of the list, or both.

Property	XML Form Object Model Property	SOM Expression
Align Left	hAlign	*DropDownList1*.caption.para.hAlign = "left" *DropDownList1*.para.hAlign = "left"
Align Center	hAlign	*DropDownList1*.caption.para.hAlign = "center" *DropDownList1*.para.hAlign = "center"
Align Right	hAlign	*DropDownList1*.caption.para.hAlign = "right" *DropDownList1*.para.hAlign = "right"

Property	XML Form Object Model Property	SOM Expression
Justify	hAlign	*DropDownList1*.caption.para.hAlign = "justify" *DropDownList1*.para.hAlign = "justify"
Align Top	vAlign	*DropDownList1*.caption.para.vAlign = "top" *DropDownList1*.para.vAlign = "top"
Align Middle	vAlign	*DropDownList1*.caption.para.vAlign = "middle" *DropDownList1*.para.vAlign = "middle"
Align Bottom	vAlign	*DropDownList1*.caption.para.vAlign = "bottom" *DropDownList1*.para.vAlign = "bottom"
Indent Left	marginLeft	*DropDownList1*.caption.para.marginLeft = "*measurement*" *DropDownList1*.para.marginLeft = "*measurement*"
Indent Right	marginRight	*DropDownList1*.caption.para.marginRight = "*measurement*" *DropDownList1*.para.marginRight = "*measurement*"
Indent First	textIndent	*DropDownList1*.caption.para.textIndent = "*measurement*" *DropDownList1*.para.textIndent = "*measurement*"
Indent By	textIndent	*DropDownList1*.caption.para.textIndent = "*measurement*" *DropDownList1*.para.textIndent = "*measurement*"
Spacing Above	spaceAbove	*DropDownList1*.caption.para.spacingAbove = "*measurement*" *DropDownList1*.para.spacingAbove = "*measurement*"
Spacing Below	spaceBelow	*DropDownList1*.caption.para.spacingBelow = "*measurement*" *DropDownList1*.para.spacingBelow = "*measurement*"
Line Spacing	lineHeight	*DropDownList1*.caption.para.lineHeight = "*measurement*" *DropDownList1*.para.lineHeight = "*measurement*"

Object

Field

Property	XML Form Object Model Property	SOM Expression
Appearance	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Add Item	addItem	*DropDownList1*.addItem( )
Remove Items	clearItems	*DropDownList1*.clearItems( )
Move Up	N/A	You cannot affect this property using scripting.

Property	XML Form Object Model Property	SOM Expression
Move Down	N/A	You cannot affect this property using scripting.
Allow Custom Text Entry	textEntry	*DropDownList1*.ui.#choiceList.textEntry = "*text*"
Presence	presence	*DropDownList1*.presence = "*visibility*"
Locale	locale	*DropDownList1*.locale = "*locale*"

Value

Property	XML Form Object Model Property	SOM Expression
Default	None rawValue	*DropDownList1* = "*value*" (FormCalc) *DropDownList1*.rawValue = "*value*" (JavaScript)
Empty Message	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Validation Pattern	picture	*DropDownList1*.validate.picture = " "
Validation Pattern Message	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Validation Pattern Message - Error	formatTest	*DropDownList1*.validate.formatTest = "*test*"
Validation Script Message	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Validation Script Message - Error	scriptTest	*DropDownList1*.validate.scriptTest = "*test*"

Binding

Property	XML Form Object Model Property	SOM Expression
Name	name	*DropDownList1*.name = "*objectname*"
Default Binding (Open, Save, Submit)	match	*DropDownList1*.bind.match = "*condition*"
Data Pattern	picture	*DropDownList1*.bind.picture = "*pictureformat*"

Property	XML Form Object Model Property	SOM Expression
Specify Item Values	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Move Up	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Move Down	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Import/Export Bindings (Execute)	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.

22 Image Field

Layout

Size and Position

Property	XML Form Object Model Property	SOM Expression
X	x	*ImageField1*.x = "*measurement*"
Y	y	*ImageField1*.y = "*measurement*"
Width	w	*ImageField1*.w = "*measurement*"
Height	h	*ImageField1*.h = "*measurement*"
Width Expand to fit	minW	*ImageField1*.minW = "*measurement*"
Height Expand to fit	minH	*ImageField1*.minH = "*measurement*"
Anchor	anchorType	*ImageField1*.anchorType = "*position*"

Rotate

Property	XML Form Object Model Property	SOM Expression
Remove Rotation	rotate	*ImageField1*.rotate = "0"
Rotate to 90 Degrees	rotate	*ImageField1*.rotate = "90"
Rotate to 180 Degrees	rotate	*ImageField1*.rotate = "180"
Rotate to 270 Degrees	rotate	*ImageField1*.rotate = "270"

Margins

Property	XML Form Object Model Property	SOM Expression
Left	leftInset	*ImageField1*.margin.leftInset = "*measurement*"
Right	rightInset	*ImageField1*.margin.rightInset = "*measurement*"
Top	topInset	*ImageField1*.margin.topInset = "*measurement*"
Bottom	bottomInset	*ImageField1*.margin.bottomInset = "*measurement*"

Caption

Property	XML Form Object Model Property	SOM Expression
Value	#text	*ImageField1*.caption.value.#text = "*text*"
Position	placement	*ImageField1*.caption.placement = "*placement*"
Reserve	reserve	*ImageField1*.caption.reserve = "*measurement*"

Font

For image field objects, the Font palette only applies to the caption of the field.

Property	XML Form Object Model Property	SOM Expression
Font	typeface	*ImageField1*.caption.font.typeface = "*font name*"
Size	size	*ImageField1*.caption.font.size = "*font size*"
Baseline Shift	baselineShift	*ImageField1*.caption.font.baselineShift = "*measurement*"

Style

Property	XML Form Object Model Property	SOM Expression
Bold	weight	*ImageField1*.caption.font.weight = "*measurement*"
Italic	posture	*ImageField1*.caption.font.posture = "*posture*"
Underline	underline	*ImageField1*.caption.font.underline = "*underline*"
Strikethrough	lineThrough	*ImageField1*.caption.font.lineThrough = "*linethrough*"
Color	value	*ImageField1*.caption.font.fill.color.value = "*R,G,B*"

Accessibility

Property	XML Form Object Model Property	SOM Expression
Tooltip	toolTip	*ImageField1*.assist.toolTip = "*text*"
Screen Reader Precedence	priority	*ImageField1*.assist.speak.priority = "*object*"
Custom Screen Reader Text	speak	*ImageField1*.assist.speak = "*text*"

Borders

Property	XML Form Object Model Property	SOM Expression
Left Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Right Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Top Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Bottom Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Rectangle Corner	join	`ImageField1.border.corner.join = "square"`
Inverted Rectangle Corner	inverted	`ImageField1.border.corner.inverted = "1"`
Round Corner	join	`ImageField1.border.corner.join = "round"`
Inverted Round Corner	inverted	`ImageField1.border.corner.inverted = "1"`
Inverted Corner Radius	radius	`ImageField1.border.corner.radius = "measurement"`

Background Fill

Property	XML Form Object Model Property	SOM Expression
None	presence	`ImageField1.border.fill.presence = "invisible"`
Solid	presence	`ImageField1.border.fill.presence = "visible"`
Solid - Color 1	value	`ImageField1.border.fill.color.value = "R,G,B"`
Linear	type	`ImageField1.border.fill.linear.type = "direction"`
Linear - Color 1	value	`ImageField1.border.fill.color.value = "R,G,B"`

Property	XML Form Object Model Property	SOM Expression
Linear - Color 2	value	*ImageField1*.border.fill.linear.color.value = "*R,G,B*"
Pattern	type	*ImageField1*.border.fill.pattern.type = "*direction*"
Pattern - Color 1	value	*ImageField1*.border.fill.color.value = "*R,G,B*"
Pattern - Color 2	value	*ImageField1*.border.fill.pattern.color.value = "*R,G,B*"
Radial	type	*ImageField1*.border.fill.radial.type = "*direction*"
Radial - Color 1	value	*ImageField1*.border.fill.color.value = "*R,G,B*"
Radial - Color 2	value	*ImageField1*.border.fill.radial.color.value = "*R,G,B*"

Paragraph

For image fields, the Paragraph palette only applies to the caption of the field.

Property	XML Form Object Model Property	SOM Expression
Align Left	hAlign	*ImageField1*.caption.para.hAlign = " "
Align Center	hAlign	*ImageField1*.caption.para.hAlign = " "
Align Right	hAlign	*ImageField1*.caption.para.hAlign = " "
Justify	hAlign	*ImageField1*.caption.para.hAlign = " "
Align Top	vAlign	*ImageField1*.caption.para.vAlign = " "
Align Middle	vAlign	*ImageField1*.caption.para.vAlign = " "
Align Bottom	vAlign	*ImageField1*.caption.para.vAlign = " "
Indent Left	marginLeft	*ImageField1*.caption.para.marginLeft = " "
Indent Right	marginRight	*ImageField1*.caption.para.marginRight = " "
Indent First	textIndent	*ImageField1*.caption.para.textIndent = " "
Indent By	textIndent	*ImageField1*.para.textIndent = " "caption.
Spacing Above	spaceAbove	*ImageField1*.caption.para.spacingAbove = " "
Spacing Below	spaceBelow	*ImageField1*.caption.para.spacingBelow = " "
Line Spacing	lineHeight	*ImageField1*.caption.para.lineHeight = " "

Object

Field

Property	XML Form Object Model Property	SOM Expression
URL	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Embed Image Data	name	*ImageField1*.desc.text.name = "embeddedHref"
Sizing	aspect	*ImageField1*.value.image.aspect = "*aspect*"
Presence	presence	*ImageField1*.presence = "*visibility*"
Locale	locale	*ImageField1*.locale = "*locale*"

Binding

Property	XML Form Object Model Property	SOM Expression
Name	name	*ImageField1*.name = "*objectname*"
Default Binding (Open, Save, Submit)	match	*ImageField1*.bind.match = "*condition*"
Import/Export Bindings (Execute)	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.

23 Line

Layout

Size and Position

Property	XML Form Object Model Property	SOM Expression
X	x	*Line1*.x = "*measurement*"
Y	y	*Line1*.y = "*measurement*"
Width	w	*Line1*.w = "*measurement*"
Height	h	*Line1*.h = "*measurement*"
Anchor	anchorType	*Line1*.anchorType = "*position*"

Rotate

Property	XML Form Object Model Property	SOM Expression
Remove Rotation	rotate	*Line1*.rotate = "0"
Rotate to 90 Degrees	rotate	*Line1*.rotate = "90"
Rotate to 180 Degrees	rotate	*Line1*.rotate = "180"
Rotate to 270 Degrees	rotate	*Line1*.rotate = "270"

Margins

Property	XML Form Object Model Property	SOM Expression
Left	leftInset	*Line1*.margin.leftInset = "*measurement*"
Right	rightInset	*Line1*.margin.rightInset = "*measurement*"
Top	topInset	*Line1*.margin.topInset = "*measurement*"
Bottom	bottomInset	*Line1*.margin.bottomInset = "*measurement*"

Object

Draw

Appearance

Property	XML Form Object Model Property	SOM Expression
-, |, /, \	N/A	Line orientation follows the values set in the x, y, and h properties for this object.
Line Style	stroke	*Line1*.value.#line.edge.stroke = " "
Line Thickness	thickness	*Line1*.value.#line.edge.thickness = " "
Line Color	value	*Line1*.value.#line.edge.color.value = " "

Presence

Property	XML Form Object Model Property	SOM Expression
Presence	presence	*Line1*.presence = " "

24 List Box

Layout

Size and Position

Property	XML Form Object Model Property	SOM Expression
X	x	`ListBox1.x = "measurement"`
Y	y	`ListBox1.y = "measurement"`
Width	w	`ListBox1.w = "measurement"`
Height	h	`ListBox1.h = "measurement"`
Width Expand to fit	minW	`ListBox1.minW = "measurement"`
Height Expand to fit	minH	`ListBox1.minH = "measurement"`
Anchor	anchorType	`ListBox1.anchorType = "position"`

Rotate

Property	XML Form Object Model Property	SOM Expression
Remove Rotation	rotate	`ListBox1.rotate = "0"`
Rotate to 90 Degrees	rotate	`ListBox1.rotate = "90"`
Rotate to 180 Degrees	rotate	`ListBox1.rotate = "180"`
Rotate to 270 Degrees	rotate	`ListBox1.rotate = "270"`

Margins

Property	XML Form Object Model Property	SOM Expression
Left	leftInset	`ListBox1.margin.leftInset = "measurement"`
Right	rightInset	`ListBox1.margin.rightInset = "measurement"`
Top	topInset	`ListBox1.margin.topInset = "measurement"`
Bottom	bottomInset	`ListBox1.margin.bottomInset = "measurement"`

Caption

Property	XML Form Object Model Property	SOM Expression
Value	#text	*ListBox1*.caption.value.#text = "*text*"
Position	placement	*ListBox1*.caption.placement = "*placement*"
Reserve	reserve	*ListBox1*.caption.reserve = "*measurement*"

Font

For list box objects, the Font palette applies to either the caption or the value of the field, or both.

Property	XML Form Object Model Property	SOM Expression
Font	typeface	*ListBox1*.caption.font.typeface = "*font name*" *ListBox1*.font.typeface = "*font name*"
Size	size	*ListBox1*.caption.font.size = "*font size*" *ListBox1*.font.size = "*font size*"
Baseline Shift	baselineShift	*ListBox1*.caption.font.baselineShift = "*measurement*" *ListBox1*.font.baselineShift = "*measurement*"

Style

Property	XML Form Object Model Property	SOM Expression
Bold	weight	*ListBox1*.caption.font.weight = "*measurement*" *ListBox1*.font.weight = "*measurement*"
Italic	posture	*ListBox1*.caption.font.posture = "*posture*" *ListBox1*.font.posture = "*posture*"
Underline	underline	*ListBox1*.caption.font.underline = "*underline*" *ListBox1*.font.underline = "*underline*"
Strikethrough	lineThrough	*ListBox1*.caption.font.lineThrough = "*linethrough*" *ListBox1*.font.lineThrough = "*linethrough*"
Color	value	*ListBox1*.caption.font.fill.color.value = "*R,G,B*" *ListBox1*.font.fill.color.value = "*R,G,B*"

Accessibility

Property	XML Form Object Model Property	SOM Expression
Tooltip	toolTip	*ListBox1*.assist.toolTip = "*text*"
Screen Reader Precedence	priority	*ListBox1*.assist.speak.priority = "*object*"
Custom Screen Reader Text	speak	*ListBox1*.assist.speak = "*text*"

Borders

Property	XML Form Object Model Property	SOM Expression
Left Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Right Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Top Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Bottom Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Rectangle Corner	join	*ListBox1*.border.corner.join = "square"
Inverted Rectangle Corner	inverted	*ListBox1*.border.corner.inverted = "1"
Round Corner	join	*ListBox1*.border.corner.join = "round"
Inverted Round Corner	inverted	*ListBox1*.border.corner.inverted = "1"
Inverted Corner Radius	radius	*ListBox1*.border.corner.radius = "*measurement*"

Background Fill

Property	XML Form Object Model Property	SOM Expression
None	presence	*ListBox1*.border.fill.presence = "invisible"
Solid	presence	*ListBox1*.border.fill.presence = "visible"
Solid - Color 1	value	*ListBox1*.border.fill.color.value = "*R,G,B*"
Linear	type	*ListBox1*.border.fill.linear.type = "*direction*"
Linear - Color 1	value	*ListBox1*.border.fill.color.value = "*R,G,B*"
Linear - Color 2	value	*ListBox1*.border.fill.linear.color.value = "*R,G,B*"
Pattern	type	*ListBox1*.border.fill.pattern.type = "*direction*"
Pattern - Color 1	value	*ListBox1*.border.fill.color.value = "*R,G,B*"
Pattern - Color 2	value	*ListBox1*.border.fill.pattern.color.value = "*R,G,B*"
Radial	type	*ListBox1*.border.fill.radial.type = "*direction*"
Radial - Color 1	value	*ListBox1*.border.fill.color.value = "*R,G,B*"
Radial - Color 2	value	*ListBox1*.border.fill.radial.color.value = "*R,G,B*"

Paragraph

For list box objects, the Paragraph palette applies to either the caption or the value of the field, or both.

Property	XML Form Object Model Property	SOM Expression
Align Left	hAlign	*ListBox1*.caption.para.hAlign = "left" *ListBox1*.para.hAlign = "left"
Align Center	hAlign	*ListBox1*.caption.para.hAlign = "center" *ListBox1*.para.hAlign = "center"
Align Right	hAlign	*ListBox1*.caption.para.hAlign = "right" *ListBox1*.para.hAlign = "right"
Justify	hAlign	*ListBox1*.caption.para.hAlign = "justify" *ListBox1*.para.hAlign = "justify"

Property	XML Form Object Model Property	SOM Expression
Align Top	vAlign	*ListBox1*.caption.para.vAlign = "top" *ListBox1*.para.vAlign = "top"
Align Middle	vAlign	*ListBox1*.caption.para.vAlign = "middle" *ListBox1*.para.vAlign = "middle"
Align Bottom	vAlign	*ListBox1*.caption.para.vAlign = "bottom" *ListBox1*.para.vAlign = "bottom"
Indent Left	marginLeft	*ListBox1*.caption.para.marginLeft = "*measurement*" *ListBox1*.para.marginLeft = "*measurement*"
Indent Right	marginRight	*ListBox1*.caption.para.marginRight = "*measurement*" *ListBox1*.para.marginRight = "*measurement*"
Indent First	textIndent	*ListBox1*.caption.para.textIndent = "*measurement*" *ListBox1*.para.textIndent = "*measurement*"
Indent By	textIndent	*ListBox1*.caption.para.textIndent = "*measurement*" *ListBox1*.para.textIndent = "*measurement*"
Spacing Above	spaceAbove	*ListBox1*.caption.para.spacingAbove = "*measurement*" *ListBox1*.para.spacingAbove = "*measurement*"
Spacing Below	spaceBelow	*ListBox1*.caption.para.spacingBelow = "*measurement*" *ListBox1*.para.spacingBelow = "*measurement*"
Line Spacing	lineHeight	*ListBox1*.caption.para.lineHeight = "*measurement*" *ListBox1*.para.lineHeight = "*measurement*"

Object

Field

Property	XML Form Object Model Property	SOM Expression
Appearance	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Add Item	addItem	*ListBox1*.addItem( )
Remove Items	clearItems	*ListBox1*.clearItems( )
Move Up	N/A	You cannot affect this property using scripting.
Move Down	N/A	You cannot affect this property using scripting.
Presence	presence	*ListBox1*.presence = "*visibility*"
Locale	locale	*ListBox1*.locale = "*locale*"

Value

Property	XML Form Object Model Property	SOM Expression
Default	None rawValue	*ListBox1* = "*value*" (FormCalc) *ListBox1*.rawValue = "*value*" (JavaScript)
Empty Message	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Validation Pattern	picture	*ListBox1*.validate.picture = "*pictureformat*"
Validation Pattern Message	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Validation Pattern Message - Error	formatTest	*ListBox1*.validate.formatTest = "*test*"
Validation Script Message	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Validation Script Message - Error	scriptTest	*ListBox1*.validate.scriptTest = "*test*"

Binding

Property	XML Form Object Model Property	SOM Expression
Name	name	*ListBox1*.name = "*objectname*"
Default Binding (Open, Save, Submit)	match	*ListBox1*.bind.match = "*condition*"
Data Pattern	picture	*ListBox1*.bind.picture = "*pictureformat*"
Specify Item Values	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Move Up	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.

Property	XML Form Object Model Property	SOM Expression
Move Down	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Import/Export Bindings (Execute)	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.

25 Numeric Field

Layout

Size and Position

Property	XML Form Object Model Property	SOM Expression
X	x	*NumericField1*.x = "*measurement*"
Y	y	*NumericField1*.y = "*measurement*"
Width	w	*NumericField1*.w = "*measurement*"
Height	h	*NumericField1*.h = "*measurement*"
Width Expand to fit	minW	*NumericField1*.minW = "*measurement*"
Height Expand to fit	minH	*NumericField1*.minH = "*measurement*"
Anchor	anchorType	*NumericField1*.anchorType = "*position*"

Rotate

Property	XML Form Object Model Property	SOM Expression
Remove Rotation	rotate	*NumericField1*.rotate = "0"
Rotate to 90 Degrees	rotate	*NumericField1*.rotate = "90"
Rotate to 180 Degrees	rotate	*NumericField1*.rotate = "180"
Rotate to 270 Degrees	rotate	*NumericField1*.rotate = "270"

Margins

Property	XML Form Object Model Property	SOM Expression
Left	leftInset	*NumericField1*.margin.leftInset = "*measurement*"
Right	rightInset	*NumericField1*.margin.rightInset = "*measurement*"
Top	topInset	*NumericField1*.margin.topInset = "*measurement*"
Bottom	bottomInset	*NumericField1*.margin.bottomInset = "*measurement*"

Caption

Property	XML Form Object Model Property	SOM Expression
Value	#text	*NumericField1*.caption.value.#text = "*text*"
Position	placement	*NumericField1*.caption.placement = "*placement*"
Reserve	reserve	*NumericField1*.caption.reserve = "*measurement*"

Font

For numeric field objects, the Font palette applies to either the caption or the value of the field, or both.

Property	XML Form Object Model Property	SOM Expression
Font	typeface	*NumericField1*.caption.font.typeface = "*font name*" *NumericField1*.font.typeface = "*font name*"
Size	size	*NumericField1*.caption.font.size = "*font size*" *NumericField1*.font.size = "*font size*"
Baseline Shift	baselineShift	*NumericField1*.caption.font.baselineShift = "*measurement*" *NumericField1*.font.baselineShift = "*measurement*"

Style

Property	XML Form Object Model Property	SOM Expression
Bold	weight	*NumericField1*.caption.font.weight = "*measurement*" *NumericField1*.font.weight = "*measurement*"
Italic	posture	*NumericField1*.caption.font.posture = "*posture*" *NumericField1*.font.posture = "*posture*"
Underline	underline	*NumericField1*.caption.font.underline = "*underline*" *NumericField1*.font.underline = "*underline*"
Strikethrough	lineThrough	*NumericField1*.caption.font.lineThrough = "*linethrough*" *NumericField1*.font.lineThrough = "*linethrough*"
Color	value	*NumericField1*.caption.font.fill.color.value = "*R,G,B*" *NumericField1*.font.fill.color.value = "*R,G,B*"

Accessibility

Property	XML Form Object Model Property	SOM Expression
Tooltip	toolTip	*NumericField1*.assist.toolTip = "*text*"
Screen Reader Precedence	priority	*NumericField1*.assist.speak.priority = "*object*"
Custom Screen Reader Text	speak	*NumericField1*.assist.speak = "*text*"

Borders

Property	XML Form Object Model Property	SOM Expression
Left Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Right Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Top Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Bottom Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Rectangle Corner	join	*NumericField1*.border.corner.join = "square"
Inverted Rectangle Corner	inverted	*NumericField1*.border.corner.inverted = "1"
Round Corner	join	*NumericField1*.border.corner.join = "round"
Inverted Round Corner	inverted	*NumericField1*.border.corner.inverted = "1"
Inverted Corner Radius	radius	*NumericField1*.border.corner.radius = "*measurement*"

Background Fill

Property	XML Form Object Model Property	SOM Expression
None	presence	*NumericField1*.border.fill.presence = "invisible"
Solid	presence	*NumericField1*.border.fill.presence = "visible"
Solid - Color 1	value	*NumericField1*.border.fill.color.value = "*R,G,B*"
Linear	type	*NumericField1*.border.fill.linear.type = "*direction*"
Linear - Color 1	value	*NumericField1*.border.fill.color.value = "*R,G,B*"
Linear - Color 2	value	*NumericField1*.border.fill.linear.color.value = "*R,G,B*"
Pattern	type	*NumericField1*.border.fill.pattern.type = "*direction*"
Pattern - Color 1	value	*NumericField1*.border.fill.color.value = "*R,G,B*"
Pattern - Color 2	value	*NumericField1*.border.fill.pattern.color.value = "*R,G,B*"
Radial	type	*NumericField1*.border.fill.radial.type = "*direction*"
Radial - Color 1	value	*NumericField1*.border.fill.color.value = "*R,G,B*"
Radial - Color 2	value	*NumericField1*.border.fill.radial.color.value = "*R,G,B*"

Paragraph

For numeric field objects, the Paragraph palette applies to either the caption or the value of the field, or both.

Property	XML Form Object Model Property	SOM Expression
Align Left	hAlign	*NumericField1*.caption.para.hAlign = "left" *NumericField1*.para.hAlign = "left"
Align Center	hAlign	*NumericField1*.caption.para.hAlign = "center" *NumericField1*.para.hAlign = "center"
Align Right	hAlign	*NumericField1*.caption.para.hAlign = "right" *NumericField1*.para.hAlign = "right"

Property	XML Form Object Model Property	SOM Expression
Justify	hAlign	*NumericField1*.caption.para.hAlign = "justify" *NumericField1*.para.hAlign = "justify"
Align Top	vAlign	*NumericField1*.caption.para.vAlign = "top" *NumericField1*.para.vAlign = "top"
Align Middle	vAlign	*NumericField1*.caption.para.vAlign = "middle" *NumericField1*.para.vAlign = "middle"
Align Bottom	vAlign	*NumericField1*.caption.para.vAlign = "bottom" *NumericField1*.para.vAlign = "bottom"
Indent Left	marginLeft	*NumericField1*.caption.para.marginLeft = "*measurement*" *NumericField1*.para.marginLeft = "*measurement*"
Indent Right	marginRight	*NumericField1*.caption.para.marginRight = "*measurement*" *NumericField1*.para.marginRight = "*measurement*"
Indent First	textIndent	*NumericField1*.caption.para.textIndent = "*measurement*" *NumericField1*.para.textIndent = "*measurement*"
Indent By	textIndent	*NumericField1*.caption.para.textIndent = "*measurement*" *NumericField1*.para.textIndent = "*measurement*"
Spacing Above	spaceAbove	*NumericField1*.caption.para.spacingAbove = "*measurement*" *NumericField1*.para.spacingAbove = "*measurement*"
Spacing Below	spaceBelow	*NumericField1*.caption.para.spacingBelow = "*measurement*" *NumericField1*.para.spacingBelow = "*measurement*"
Line Spacing	lineHeight	*NumericField1*.caption.para.lineHeight = "*measurement*" *NumericField1*.para.lineHeight = "*measurement*"

Object

Field

Property	XML Form Object Model Property	SOM Expression
Appearance	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Display Pattern	picture	*NumericField1*.ui.picture = "*pictureformat*"
Edit Pattern	picture	*NumericField1*.format.picture = "*pictureformat*"
Presence	presence	*NumericField1*.presence = "*visibility*"
Locale	locale	*NumericField1*.locale = "*locale*"

Value

Property	XML Form Object Model Property	SOM Expression
Default	None rawValue	*NumericField1* = "*value*" (FormCalc) *NumericField1*.rawValue = "*value*" (JavaScript)
Empty Message	N/A	You cannot change this property using scripting.
Validation Pattern	picture	*NumericField1*.validate.picture = " "
Validation Pattern Message	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Validation Pattern Message - Error	formatTest	*NumericField1*.validate.formatTest = "*test*"
Validation Script Message	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Validation Script Message - Error	scriptTest	*NumericField1*.validate.scriptTest = "*test*"

Binding

Property	XML Form Object Model Property	SOM Expression
Name	name	*NumericField1*.name = "*objectname*"
Default Binding (Open, Save, Submit)	match	*NumericField1*.bind.match = "*condition*"
Data Pattern	picture	*NumericField1*.bind.picture = "*pictureformat*"
Data Format	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Import/Export Bindings (Execute)	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.

26 Password Field

Layout

Size and Position

Property	XML Form Object Model Property	SOM Expression
X	x	*PasswordField1*.x = "*measurement*"
Y	y	*PasswordField1*.y = "*measurement*"
Width	w	*PasswordField1*.w = "*measurement*"
Height	h	*PasswordField1*.h = "*measurement*"
Width Expand to fit	minW	*PasswordField1*.minW = "*measurement*"
Height Expand to fit	minH	*PasswordField1*.minH = "*measurement*"
Anchor	anchorType	*PasswordField1*.anchorType = "*position*"

Rotate

Property	XML Form Object Model Property	SOM Expression
Remove Rotation	rotate	*PasswordField1*.rotate = "0"
Rotate to 90 Degrees	rotate	*PasswordField1*.rotate = "90"
Rotate to 180 Degrees	rotate	*PasswordField1*.rotate = "180"
Rotate to 270 Degrees	rotate	*PasswordField1*.rotate = "270"

Margins

Property	XML Form Object Model Property	SOM Expression
Left	rotate	*PasswordField1*.margin.leftInset = "*measurement*"
Right	rotate	*PasswordField1*.margin.rightInset = "*measurement*"
Top	rotate	*PasswordField1*.margin.topInset = "*measurement*"
Bottom	rotate	*PasswordField1*.margin.bottomInset = "*measurement*

Caption

Property	XML Form Object Model Property	SOM Expression
Value	#text	*PasswordField1*.caption.value.text = "*text*"
Position	placement	*PasswordField1*.caption.placement = "*placement*"
Reserve	reserve	*PasswordField1*.caption.reserve = "*measurement*"

Font

For password field objects, the Font palette applies to either the caption or the value of the field, or both.

Property	XML Form Object Model Property	SOM Expression
Font	typeface	*PasswordField1*.caption.font.typeface = "*font name*" *PasswordField1*.font.typeface = "*font name*"
Size	size	*PasswordField1*.caption.font.size = "*font size*" *PasswordField1*.font.size = "*font size*"
Baseline Shift	baselineShift	*PasswordField1*.caption.font.baselineShift = "*measurement*" *PasswordField1*.font.baselineShift = "*measurement*"

Style

Property	XML Form Object Model Property	SOM Expression
Bold	weight	*PasswordField1*.caption.font.weight = "*measurement*" *PasswordField1*.font.weight = "*measurement*"
Italic	posture	*PasswordField1*.caption.font.posture = "*posture*" *PasswordField1*.font.posture = "*posture*"
Underline	underline	*PasswordField1*.caption.font.underline = "*underline*" *PasswordField1*.font.underline = "*underline*"
Strikethrough	lineThrough	*PasswordField1*.caption.font.lineThrough = "*linethrough*" *PasswordField1*.font.lineThrough = "*linethrough*"
Color	value	*PasswordField1*.caption.font.fill.color.value = "*R,G,B*" *PasswordField1*.font.fill.color.value = "*R,G,B*"

Accessibility

Property	XML Form Object Model Property	SOM Expression
Tooltip	toolTip	*PasswordField1*.assist.toolTip = "*text*"
Screen Reader Precedence	priority	*PasswordField1*.assist.speak.priority = "*object*"
Custom Screen Reader Text	speak	*PasswordField1*.assist.speak = "*text*"

Borders

Property	XML Form Object Model Property	SOM Expression
Left Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Right Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Top Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Bottom Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Rectangle Corner	join	*PasswordField1*.border.corner.join = "square"
Inverted Rectangle Corner	inverted	*PasswordField1*.border.corner.inverted = "1"
Round Corner	join	*PasswordField1*.border.corner.join = "round"

Property	XML Form Object Model Property	SOM Expression
Inverted Round Corner	inverted	*PasswordField1*.border.corner.inverted = "1"
Inverted Corner Radius	radius	*PasswordField1*.border.corner.radius = "*measurement*"

Background Fill

Property	XML Form Object Model Property	SOM Expression
None	presence	*PasswordField1*.border.fill.presence = "invisible"
Solid	presence	*PasswordField1*.border.fill.presence = "visible"
Solid - Color 1	value	*PasswordField1*.border.fill.color.value = "*R,G,B*"
Linear	type	*PasswordField1*.border.fill.linear.type = "*direction*"
Linear - Color 1	value	*PasswordField1*.border.fill.color.value = "*R,G,B*"
Linear - Color 2	value	*PasswordField1*.border.fill.linear.color.value = "*R,G,B*"
Pattern	type	*PasswordField1*.border.fill.pattern.type = "*direction*"
Pattern - Color 1	value	*PasswordField1*.border.fill.color.value = "*R,G,B*"
Pattern - Color 2	value	*PasswordField1*.border.fill.pattern.color.value = "*R,G,B*"
Radial	type	*PasswordField1*.border.fill.radial.type = "*direction*"
Radial - Color 1	value	*PasswordField1*.border.fill.color.value = "*R,G,B*"
Radial - Color 2	value	*PasswordField1*.border.fill.radial.color.value = "*R,G,B*"

Paragraph

For password field objects, the Paragraph palette applies to either the caption or the value of the field, or both.

Property	XML Form Object Model Property	SOM Expression
Align Left	hAlign	*PasswordField1*.caption.para.hAlign = "left" *PasswordField1*.para.hAlign = "left"
Align Center	hAlign	*PasswordField1*.caption.para.hAlign = "center" *PasswordField1*.para.hAlign = "center"
Align Right	hAlign	*PasswordField1*.caption.para.hAlign = "right" *PasswordField1*.para.hAlign = "right"
Justify	hAlign	*PasswordField1*.caption.para.hAlign = "justify" *PasswordField1*.para.hAlign = "justify"
Align Top	vAlign	*PasswordField1*.caption.para.vAlign = "top" *PasswordField1*.para.vAlign = "top"
Align Middle	vAlign	*PasswordField1*.caption.para.vAlign = "middle" *PasswordField1*.para.vAlign = "middle"
Align Bottom	vAlign	*PasswordField1*.caption.para.vAlign = "bottom" *PasswordField1*.para.vAlign = "bottom"
Indent Left	marginLeft	*PasswordField1*.caption.para.marginLeft = "*measurement*" *PasswordField1*.para.marginLeft = "*measurement*"
Indent Right	marginRight	*PasswordField1*.caption.para.marginRight = "*measurement*" *PasswordField1*.para.marginRight = "*measurement*"
Indent First	textIndent	*PasswordField1*.caption.para.textIndent = "*measurement*" *PasswordField1*.para.textIndent = "*measurement*"
Indent By	textIndent	*PasswordField1*.caption.para.textIndent = "*measurement*" *PasswordField1*.para.textIndent = "*measurement*"
Spacing Above	spaceAbove	*PasswordField1*.caption.para.spacingAbove = "*measurement*" *PasswordField1*.para.spacingAbove = "*measurement*"
Spacing Below	spaceBelow	*PasswordField1*.caption.para.spacingBelow = "*measurement*" *PasswordField1*.para.spacingBelow = "*measurement*"
Line Spacing	lineHeight	*PasswordField1*.caption.para.lineHeight = "*measurement*" *PasswordField1*.para.lineHeight = "*measurement*"

Object

Field

Property	XML Form Object Model Property	SOM Expression
Appearance	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Password Display Character	passwordChar	`PasswordField1.ui.#passwordEdit.passwordChar = "character"`
Edit Pattern	picture	`PasswordField1.ui.picture = "pictureformat"`
Presence	presence	`PasswordField1.presence = "visibility"`
Locale	locale	`PasswordField1.locale = "locale"`

Value

Property	XML Form Object Model Property	SOM Expression
Empty Message	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Validation Pattern	picture	`PasswordField1.validate.picture = "pictureformat"`
Validation Pattern Message	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Validation Pattern Message - Error	formatTest	`PasswordField1.validate.formatTest = "test"`
Validation Script Message	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Validation Script Message - Error	scriptTest	`PasswordField1.validate.scriptTest = "test"`

Binding

Property	XML Form Object Model Property	SOM Expression
Name	name	*PasswordField1*.name = "*objectname*"
Default Binding (Open, Save, Submit)	match	*PasswordField1*.bind.match = "*condition*"
Data Pattern	picture	*PasswordField1*.bind.picture = "*pictureformat*"
Import/Export Bindings (Execute)	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.

27 Radio Button

Layout

Size and Position

Property	XML Form Object Model Property	SOM Expression
X	x	*RadioButton1*.x = "*measurement*"
Y	y	*RadioButton1*.y = "*measurement*"
Width	w	*RadioButton1*.w = "*measurement*"
Height	h	*RadioButton1*.h = "*measurement*"
Width Expand to fit	minW	*RadioButton1*.minW = "*measurement*"
Height Expand to fit	minH	*RadioButton1*.minH = "*measurement*"
Anchor	anchorType	*RadioButton1*.anchorType = "*position*"

Rotate

Property	XML Form Object Model Property	SOM Expression
Remove Rotation	rotate	*RadioButton1*.rotate = "0"
Rotate to 90 Degrees	rotate	*RadioButton1*.rotate = "90"
Rotate to 180 Degrees	rotate	*RadioButton1*.rotate = "180"
Rotate to 270 Degrees	rotate	*RadioButton1*.rotate = "270"

Margins

Property	XML Form Object Model Property	SOM Expression
Left	leftInset	*RadioButton1*.margin.leftInset = "*measurement*"
Right	rightInset	*RadioButton1*.margin.rightInset = "*measurement*"
Top	topInset	*RadioButton1*.margin.topInset = "*measurement*"
Bottom	bottomInset	*RadioButton1*.margin.bottomInset = "*measurement*"

Caption

Property	XML Form Object Model Property	SOM Expression
Value	#text	*RadioButton1*.caption.value.#text = "*text*"
Position	placement	*RadioButton1*.caption.placement = "*placement*"
Reserve	reserve	*RadioButton1*.caption.reserve = "*alignment*"

Font

For radio buttons, the Font palette only applies to the caption of the radio button.

Property	XML Form Object Model Property	SOM Expression
Font	typeface	*RadioButton1*.caption.font.typeface = "*font name*"
Size	size	*RadioButton1*.caption.font.size = "*font size*"
Baseline Shift	baselineShift	*RadioButton1*.caption.font.baselineShift = "*measurement*"

Style

Property	XML Form Object Model Property	SOM Expression
Bold	weight	*RadioButton1*.caption.font.weight = "*measurement*"
Italic	posture	*RadioButton1*.caption.font.posture = "*posture*"
Underline	underline	*RadioButton1*.caption.font.underline = "*underline*"
Strikethrough	lineThrough	*RadioButton1*.caption.font.lineThrough = "*linethrough*"
Color	value	*RadioButton1*.caption.font.fill.color.value = "*R,G,B*"

Accessibility

Property	XML Form Object Model Property	SOM Expression
Tool tip	toolTip	*RadioButton1*.assist.toolTip = "*text*"

Property	XML Form Object Model Property	SOM Expression
Screen Reader Precedence	priority	`RadioButton1.assist.speak.priority = "object"`
Custom Screen Reader Text	speak	`RadioButton1.assist.speak = "text"`

Borders

Property	XML Form Object Model Property	SOM Expression
Left Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Right Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Top Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Bottom Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Rectangle Corner	join	`RadioButton1.border.corner.join = "square"`
Inverted Rectangle Corner	inverted	`RadioButton1.border.corner.inverted = "1"`
Round Corner	join	`RadioButton1.border.corner.join = "round"`
Inverted Round Corner	inverted	`RadioButton1.border.corner.inverted = "1"`
Inverted Corner Radius	radius	`RadioButton1.border.corner.radius = "measurement"`

Background Fill

Property	XML Form Object Model Property	SOM Expression
None	presence	*RadioButton1*.border.fill.presence = "invisible"
Solid	presence	*RadioButton1*.border.fill.presence = "visible"
Solid - Color 1	value	*RadioButton1*.border.fill.color.value = "*R,G,B*"
Linear	type	*RadioButton1*.border.fill.linear.type = "*direction*"
Linear - Color 1	value	*RadioButton1*.border.fill.color.value = "*R,G,B*"
Linear - Color 2	value	*RadioButton1*.border.fill.linear.color.value = "*R,G,B*"
Pattern	type	*RadioButton1*.border.fill.pattern.type = "*direction*"
Pattern - Color 1	value	*RadioButton1*.border.fill.color.value = "*R,G,B*"
Pattern - Color 2	value	*RadioButton1*.border.fill.pattern.color.value = "*R,G,B*"
Radial	type	*RadioButton1*.border.fill.radial.type = "*direction*"
Radial - Color 1	value	*RadioButton1*.border.fill.color.value = "*R,G,B*"
Radial - Color 2	value	*RadioButton1*.border.fill.radial.color.value = "*R,G,B*"

Paragraph

For radio button objects, the Paragraph palette only applies to the caption of the radio button.

Property	XML Form Object Model Property	SOM Expression
Align Left	hAlign	*RadioButton1*.caption.para.hAlign = "left"
Align Center	hAlign	*RadioButton1*.caption.para.hAlign = "center"
Align Right	hAlign	*RadioButton1*.caption.para.hAlign = "right"
Justify	hAlign	*RadioButton1*.caption.para.hAlign = "justify"

Property	XML Form Object Model Property	SOM Expression
Align Top	vAlign	*RadioButton1*.caption.para.vAlign = "top"
Align Middle	vAlign	*RadioButton1*.caption.para.vAlign = "middle"
Align Bottom	vAlign	*RadioButton1*.caption.para.vAlign = "bottom"
Indent Left	marginLeft	*RadioButton1*.caption.para.marginLeft = "*measurement*"
Indent Right	marginRight	*RadioButton1*.caption.para.marginRight = "*measurement*"
Indent First	textIndent	*RadioButton1*.caption.para.textIndent = "*measurement*"
Indent By	textIndent	*RadioButton1*.caption.para.textIndent = "*measurement*"
Spacing Above	spaceAbove	*RadioButton1*.caption.para.spacingAbove = "*measurement*"
Spacing Below	spaceBelow	*RadioButton1*.caption.para.spacingBelow = "*measurement*"
Line Spacing	lineHeight	*RadioButton1*.caption.para.lineHeight = "*measurement*"

Object

Field

Property	XML Form Object Model Property	SOM Expression
Group	name	*RadioButtonList*.name = "*groupname*"
On Value	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Appearance	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Size	size	*RadioButton1*.ui.checkButton.size = "*size*"
Presence	presence	*RadioButton1*.presence = "*visibility*"
Locale	locale	*RadioButton1*.locale = "*locale*"

Group Value

Property	XML Form Object Model Property	SOM Expression
Default	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Override Message	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Validation Script Message	N/A	You cannot change this property using scripting.
Validation Script Message - Error	scriptTest	*RadioButton1*.validate.scriptTest = "*test*"

Group Binding

Property	XML Form Object Model Property	SOM Expression
Name	name	*RadioButton1*.name = "*objectname*"
Default Binding (Open, Save, Submit)	match	*RadioButton1*.bind.match = "*condition*"
Import/Export Bindings (Execute)	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.

28 Rectangle

Layout

Size and Position

Property	XML Form Object Model Property	SOM Expression
X	x	*Rectangle1*.x = "*measurement*"
Y	y	*Rectangle1*.y = "*measurement*"
Width	w	*Rectangle1*.w = "*measurement*"
Height	h	*Rectangle1*.h = "*measurement*"
Anchor	anchorType	*Rectangle1*.anchorType = "*position*"

Rotate

Property	XML Form Object Model Property	SOM Expression
Remove Rotation	rotate	*Rectangle1*.rotate = "0"
Rotate to 90 Degrees	rotate	*Rectangle1*.rotate = "90"
Rotate to 180 Degrees	rotate	*Rectangle1*.rotate = "180"
Rotate to 270 Degrees	rotate	*Rectangle1*.rotate = "270"

Margins

Property	XML Form Object Model Property	SOM Expression
Left	leftInset	*Rectangle1*.margin.leftInset = "*measurement*"
Right	rightInset	*Rectangle1*.margin.rightInset = "*measurement*"
Top	topInset	*Rectangle1*.margin.topInset = "*measurement*"
Bottom	bottomInset	*Rectangle1*.margin.bottomInset = "*measurement*"

Object

Draw

Appearance

Property	XML Form Object Model Property	SOM Expression
Line Style	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Line Thickness	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Line Color	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Rectangle Corner	join	`Rectangle1.border.corner.join = "square"`
Inverted Rectangle Corner	inverted	`Rectangle1.border.corner.inverted = "1"`
Round Corner	join	`Rectangle1.border.corner.join = "round"`
Inverted Round Corner	inverted	`Rectangle1.border.corner.inverted = "1"`
Inverted Corner Radius	radius	`Rectangle1.border.corner.radius = "measurement"`

Fill style and color

Property	XML Form Object Model Property	SOM Expression
None	presence	`Rectangle1.value.rectangle.fill.presence = "invisible"`
Solid	presence	`Rectangle1.value.rectangle.fill.presence = "visible"`
Solid - Color 1	value	`Rectangle1.value.rectangle.fill.color.value = "R,G,B"`
Linear	type	`Rectangle1.value.rectangle.fill.linear.type = "direction"`
Linear - Color 1	value	`Rectangle1.value.rectangle.fill.color.value = "R,G,B"`

Property	XML Form Object Model Property	SOM Expression
Linear - Color 2	value	*Rectangle1*.value.rectangle.fill.linear.color.value = "*R,G,B*"
Pattern	type	*Rectangle1*.value.rectangle.fill.pattern.type = "*direction*"
Pattern - Color 1	value	*Rectangle1*.value.rectangle.fill.color.value = "*R,G,B*"
Pattern - Color 2	value	*Rectangle1*.value.rectangle.fill.pattern.color.value = "*R,G,B*"
Radial	type	*Rectangle1*.value.rectangle.fill.radial.type = "*direction*"
Radial - Color 1	value	*Rectangle1*.value.rectangle.fill.color.value = "*R,G,B*"
Radial - Color 2	value	*Rectangle1*.value.rectangle.fill.radial.color.value = "*R,G,B*"

Presence

Property	XML Form Object Model Property	SOM Expression
Presence	presence	*Rectangle1*.presence = "*visibility*"

29 Signature Field

Layout

Size and Position

Property	XML Form Object Model Property	SOM Expression
X	x	*SignatureField1*.x = "*measurement*"
Y	y	*SignatureField1*.y = "*measurement*"
Width	w	*SignatureField1*.w = "*measurement*"
Height	h	*SignatureField1*.h = "*measurement*"
Width Expand to fit	minW	*SignatureField1*.minW = "*measurement*"
Height Expand to fit	minH	*SignatureField1*.minH = "*measurement*"
Anchor	anchorType	*SignatureField1*.anchorType = "*position*"

Rotate

Property	XML Form Object Model Property	SOM Expression
Remove Rotation	rotate	*SignatureField1*.rotate = "0"
Rotate to 90 Degrees	rotate	*SignatureField1*.rotate = "90"
Rotate to 180 Degrees	rotate	*SignatureField1*.rotate = "180"
Rotate to 270 Degrees	rotate	*SignatureField1*.rotate = "270"

Margins

Property	XML Form Object Model Property	SOM Expression
Left	leftInset	*SignatureField1*.margin.leftInset = "*measurement*"
Right	rightInset	*SignatureField1*.margin.rightInset = "*measurement*"
Top	topInset	*SignatureField1*.margin.topInset = "*measurement*"
Bottom	bottomInset	*SignatureField1*.margin.bottomInset = "*measurement*"

Caption

Property	XML Form Object Model Property	SOM Expression
Value	#text	*SignatureField1*.caption.value.#text = "*text*"
Position	placement	*SignatureField1*.caption.placement = "*placement*"
Reserve	reserve	*SignatureField1*.caption.reserve = "*measurement*"

Font

For signature field objects, the Font palette only applies to the caption of the field.

Property	XML Form Object Model Property	SOM Expression
Font	typeface	*SignatureField1*.caption.font.typeface = "*font name*"
Size	size	*SignatureField1*.caption.font.size = "*font size*"
Baseline Shift	baselineShift	*SignatureField1*.caption.font.baselineShift = "*measurement*"

Style

Property	XML Form Object Model Property	SOM Expression
Bold	weight	*SignatureField1*.caption.font.weight = "*measurement*"
Italic	posture	*SignatureField1*.caption.font.posture = "*posture*"
Underline	underline	*SignatureField1*.caption.font.underline = "*underline*"
Strikethrough	lineThrough	*SignatureField1*.caption.font.lineThrough = "*linethrough*"
Color	value	*SignatureField1*.caption.font.fill.color.value = "*R,G,B*"

Accessibility

Property	XML Form Object Model Property	SOM Expression
Tooltip	toolTip	*SignatureField1*.assist.toolTip = "*text*"
Screen Reader Precedence	priority	*SignatureField1*.assist.speak.priority = "*object*"
Custom Screen Reader Text	speak	*SignatureField1*.assist.speak = "*text*"

Borders

Property	XML Form Object Model Property	SOM Expression
Left Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Right Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Top Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Bottom Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Rectangle Corner	join	*SignatureField1*.border.corner.join = "square"
Inverted Rectangle Corner	inverted	*SignatureField1*.border.corner.inverted = "1"
Round Corner	join	*SignatureField1*.border.corner.join = "round"

Property	XML Form Object Model Property	SOM Expression
Inverted Round Corner	inverted	*SignatureField1*.border.corner.inverted = "1"
Inverted Corner Radius	radius	*SignatureField1*.border.corner.radius = "*measurement*"

Background Fill

Property	XML Form Object Model Property	SOM Expression
None	presence	*SignatureField1*.border.fill.presence = "invisible"
Solid	presence	*SignatureField1*.border.fill.presence = "visible"
Solid - Color 1	value	*SignatureField1*.border.fill.color.value = "*R,G,B*"
Linear	type	*SignatureField1*.border.fill.linear.type = "*direction*"
Linear - Color 1	value	*SignatureField1*.border.fill.color.value = "*R,G,B*"
Linear - Color 2	value	*SignatureField1*.border.fill.linear.color.value = "*R,G,B*"
Pattern	type	*SignatureField1*.border.fill.pattern.type = "*direction*"
Pattern - Color 1	value	*SignatureField1*.border.fill.color.value = "*R,G,B*"
Pattern - Color 2	value	*SignatureField1*.border.fill.pattern.color.value = "*R,G,B*"
Radial	type	*SignatureField1*.border.fill.radial.type = "*direction*"
Radial - Color 1	value	*SignatureField1*.border.fill.color.value = "*R,G,B*"
Radial - Color 2	value	*SignatureField1*.border.fill.radial.color.value = "*R,G,B*"

Paragraph

For signature field objects, the Paragraph palette applies only applies to the caption of the field.

Property	XML Form Object Model Property	SOM Expression
Align Left	hAlign	*SignatureField1*.caption.para.hAlign = "left"
Align Center	hAlign	*SignatureField1*.caption.para.hAlign = "center"
Align Right	hAlign	*SignatureField1*.caption.para.hAlign = "right"
Justify	hAlign	*SignatureField1*.caption.para.hAlign = "justify"
Align Top	vAlign	*SignatureField1*.caption.para.vAlign = "top"
Align Middle	vAlign	*SignatureField1*.caption.para.vAlign = "middle"
Align Bottom	vAlign	*SignatureField1*.caption.para.vAlign = "bottom"
Indent Left	marginLeft	*SignatureField1*.caption.para.marginLeft = "*measurement*"
Indent Right	marginRight	*SignatureField1*.caption.para.marginRight = "*measurement*"
Indent First	textIndent	*SignatureField1*.caption.para.textIndent = "*measurement*"
Indent By	textIndent	*SignatureField1*.caption.para.textIndent = "*measurement*"
Spacing Above	spaceAbove	*SignatureField1*.caption.para.spacingAbove = "*measurement*"
Spacing Below	spaceBelow	*SignatureField1*.caption.para.spacingBelow = "*measurement*"
Line Spacing	lineHeight	*SignatureField1*.caption.para.lineHeight = "*measurement*"

Object

Field

Property	XML Form Object Model Property	SOM Expression
Appearance	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.

Property	XML Form Object Model Property	SOM Expression
Presence	presence	*SignatureField1*.presence = "*visibility*"
Locale	locale	*SignatureField1*.locale = "*locale*"

30 Static Image

Layout

Size and Position

Property	XML Form Object Model Property	SOM Expression
X	x	*StaticImage1*.x = "*measurement*"
Y	y	*StaticImage1*.y = "*measurement*"
Width	w	*StaticImage1*.w = "*measurement*"
Height	h	*StaticImage1*.h = "*measurement*"
Width Expand to fit	minW	*StaticImage1*.minW = "*measurement*"
Height Expand to fit	minH	*StaticImage1*.minH = "*measurement*"
Anchor	anchorType	*StaticImage1*.anchorType = "*position*"

Rotate

Property	XML Form Object Model Property	SOM Expression
Remove Rotation	rotate	*StaticImage1*.rotate = "0"
Rotate to 90 Degrees	rotate	*StaticImage1*.rotate = "90"
Rotate to 180 Degrees	rotate	*StaticImage1*.rotate = "180"
Rotate to 270 Degrees	rotate	*StaticImage1*.rotate = "270"

Margins

Property	XML Form Object Model Property	SOM Expression
Left	leftInset	*StaticImage1*.margin.leftInset = "*measurement*"
Right	rightInset	*StaticImage1*.margin.rightInset = "*measurement*"
Top	topInset	*StaticImage1*.margin.topInset = "*measurement*"
Bottom	bottomInset	*StaticImage1*.margin.bottomInset = "*measurement*"

Accessibility

Property	XML Form Object Model Property	SOM Expression
Tooltip	toolTip	*StaticImage1*.assist.toolTip = "*text*"
Screen Reader precedence	priority	*StaticImage1*.assist.speak.priority = "*object*"
Custom Screen Reader Text	speak	*StaticImage1*.assist.speak = "*text*"

Borders

Property	XML Form Object Model Property	SOM Expression
Left Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Right Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Top Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Bottom Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Rectangle Corner	join	*StaticImage1*.border.corner.join = "square"
Inverted Rectangle Corner	inverted	*StaticImage1*.border.corner.inverted = "1"
Round Corner	join	*StaticImage1*.border.corner.join = "round"
Inverted Round Corner	inverted	*StaticImage1*.border.corner.inverted = "1"
Inverted Corner Radius	radius	*StaticImage1*.border.corner.radius = "*measurement*"

Background Fill

Property	XML Form Object Model Property	SOM Expression
None	presence	*StaticImage1*.border.fill.presence = "invisible"
Solid	presence	*StaticImage1*.border.fill.presence = "visible"
Solid - Color 1	value	*StaticImage1*.border.fill.color.value = "*R,G,B*"
Linear	type	*StaticImage1*.border.fill.linear.type = "*direction*"
Linear - Color 1	value	*StaticImage1*.border.fill.color.value = "*R,G,B*"
Linear - Color 2	value	*StaticImage1*.border.fill.linear.color.value = "*R,G,B*"
Pattern	type	*StaticImage1*.border.fill.pattern.type = "*direction*"
Pattern - Color 1	value	*StaticImage1*.border.fill.color.value = "*R,G,B*"
Pattern - Color 2	value	*StaticImage1*.border.fill.pattern.color.value = "*R,G,B*"
Radial	type	*StaticImage1*.border.fill.radial.type = "*direction*"
Radial - Color 1	value	*StaticImage1*.border.fill.color.value = "*R,G,B*"
Radial - Color 2	value	*StaticImage1*.border.fill.radial.color.value = "*R,G,B*"

Object

Draw

Property	XML Form Object Model Property	SOM Expression
URL	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Embed Image Data	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Sizing	aspect	*StaticImage1*.value.#image.aspect = " "

Property	XML Form Object Model Property	SOM Expression
Presence	presence	*StaticImage1*.presence = " "
Locale	locale	*StaticImage1*.locale = " "

31 Static Text

Layout

Size and Position

Property	XML Form Object Model Property	SOM Expression
X	x	*StaticText1*.x = "*measurement*"
Y	y	*StaticText1*.y = "*measurement*"
Width	w	*StaticText1*.w = "*measurement*"
Height	h	*StaticText1*.h = "*measurement*"
Width Expand to fit	minW	*StaticText1*.minW = "*measurement*"
Height Expand to fit	minH	*StaticText1*.minH = "*measurement*"
Anchor	anchorType	*StaticText1*.anchorType = "*position*"

Rotate

Property	XML Form Object Model Property	SOM Expression
Remove Rotation	rotate	*StaticText1*.rotate = "0"
Rotate to 90 Degrees	rotate	*StaticText1*.rotate = "90"
Rotate to 180 Degrees	rotate	*StaticText1*.rotate = "180"
Rotate to 270 Degrees	rotate	*StaticText1*.rotate = "270"

Margins

Property	XML Form Object Model Property	SOM Expression
Left	leftInset	*StaticText1*.margin.leftInset = " "
Right	rightInset	*StaticText1*.margin.rightInset = " "
Top	topInset	*StaticText1*.margin.topInset = " "
Bottom	bottomInset	*StaticText1*.margin.bottomInset = " "

Caption

Property	XML Form Object Model Property	SOM Expression
Value	#text	*StaticText1*.caption.value.#text = "*text*"
Position	N/A	Disabled for this form object.
Reserve	N/A	Disabled for this form object.

Font

For static text objects, the Font palette only applies to the caption of the object.

Property	XML Form Object Model Property	SOM Expression
Font	typeface	*StaticText1*.caption.font.typeface = "*font name*"
Size	size	*StaticText1*.caption.font.size = "*font size*"
Baseline Shift	baselineShift	*StaticText1*.caption.font.baselineShift = "*measurement*"

Style

Property	XML Form Object Model Property	SOM Expression
Bold	weight	*StaticText1*.caption.font.weight = "*measurement*"
Italic	posture	*StaticText1*.caption.font.posture = "*posture*"
Underline	underline	*StaticText1*.caption.font.underline = "*underline*"
Strikethrough	lineThrough	*StaticText1*.caption.font.lineThrough = "*linethrough*"
Color	value	*StaticText1*.caption.font.fill.color.value = "*R,G,B*"

Accessibility

Property	XML Form Object Model Property	SOM Expression
Tooltip	toolTip	*StaticText1*.assist.toolTip = "*text*"

Property	XML Form Object Model Property	SOM Expression
Screen Reader Precedence	priority	*StaticText1*.assist.speak.priority = "*object*"
Custom Screen Reader Text	speak	*StaticText1*.assist.speak = "*text*"

Borders

Property	XML Form Object Model Property	SOM Expression
Left Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Right Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Top Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Bottom Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Rectangle Corner	join	*StaticText1*.border.corner.join = "square"
Inverted Rectangle Corner	inverted	*StaticText1*.border.corner.inverted = "1"
Round Corner	join	*StaticText1*.border.corner.join = "round"
Inverted Round Corner	inverted	*StaticText1*.border.corner.inverted = "1"
Inverted Corner Radius	radius	*StaticText1*.border.corner.radius = "*measurement*"

Background Fill

Property	XML Form Object Model Property	SOM Expression
None	presence	*StaticText1*.border.fill.presence = "invisible"
Solid	presence	*StaticText1*.border.fill.presence = "visible"
Solid - Color 1	value	*StaticText1*.border.fill.color.value = "*R,G,B*"
Linear	type	*StaticText1*.border.fill.linear.type = "*direction*"
Linear - Color 1	value	*StaticText1*.border.fill.color.value = "*R,G,B*"
Linear - Color 2	value	*StaticText1*.border.fill.linear.color.value = "*R,G,B*"
Pattern	type	*StaticText1*.border.fill.pattern.type = "*direction*"
Pattern - Color 1	value	*StaticText1*.border.fill.color.value = "*R,G,B*"
Pattern - Color 2	value	*StaticText1*.border.fill.pattern.color.value = "*R,G,B*"
Radial	type	*StaticText1*.border.fill.radial.type = "*direction*"
Radial - Color 1	value	*StaticText1*.border.fill.color.value = "*R,G,B*"
Radial - Color 2	value	*StaticText1*.border.fill.radial.color.value = "*R,G,B*"

Paragraph

For static text objects, the Paragraph palette only applies to the caption of the object.

Property	XML Form Object Model Property	SOM Expression
Align Left	hAlign	*StaticText1*.caption.para.hAlign = "left"
Align Center	hAlign	*StaticText1*.caption.para.hAlign = "center"
Align Right	hAlign	*StaticText1*.caption.para.hAlign = "right"
Justify	hAlign	*StaticText1*.caption.para.hAlign = "justify"

Property	XML Form Object Model Property	SOM Expression
Align Top	vAlign	*StaticText1*.caption.para.vAlign = "top"
Align Middle	vAlign	*StaticText1*.caption.para.vAlign = "middle"
Align Bottom	vAlign	*StaticText1*.caption.para.vAlign = "bottom"
Indent Left	marginLeft	*StaticText1*.caption.para.marginLeft = "*measurement*"
Indent Right	marginRight	*StaticText1*.caption.para.marginRight = "*measurement*"
Indent First	textIndent	*StaticText1*.caption.para.textIndent = "*measurement*"
Indent By	textIndent	*StaticText1*.caption.para.textIndent = "*measurement*"
Spacing Above	spaceAbove	*StaticText1*.caption.para.spacingAbove = "*measurement*"
Spacing Below	spaceBelow	*StaticText1*.caption.para.spacingBelow = "*measurement*"
Line Spacing	lineHeight	*StaticText1*.caption.para.lineHeight = "*measurement*"

Object

Draw

Property	XML Form Object Model Property	SOM Expression
Presence	presence	*TextField1*.presence = "*visibility*"
Locale	locale	*TextField1*.locale = "*locale*"

32 Subform

Layout

Size and Position

Property	XML Form Object Model Property	SOM Expression
X	x	*Subform1*.x = "*measurement*"
Y	y	*Subform1*.y = "*measurement*"
Width	w	*Subform1*.w = "*measurement*"
Height	h	*Subform1*.h = "*measurement*"
Width Expand to fit	minW	*Subform1*.minW = "*measurement*"
Height Expand to fit	minH	*Subform1*.minH = "*measurement*"
Anchor	anchorType	*Subform1*.anchorType = "*position*"

Rotate

Property	XML Form Object Model Property	SOM Expression
Remove Rotation	rotate	*Subform1*.rotate = "0"
Rotate to 90 Degrees	rotate	*Subform1*.rotate = "90"
Rotate to 180 Degrees	rotate	*Subform1*.rotate = "180"
Rotate to 270 Degrees	rotate	*Subform1*.rotate = "270"

Margins

Property	XML Form Object Model Property	SOM Expression
Left	leftInset	*Subform1*.margin.leftInset = "*measurement*"
Right	rightInset	*Subform1*.margin.rightInset = "*measurement*"
Top	topInset	*Subform1*.margin.topInset = "*measurement*"
Bottom	bottomInset	*Subform1*.margin.bottomInset = "*measurement*"

Borders

Property	XML Form Object Model Property	SOM Expression
Left Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Right Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Top Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Bottom Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Rectangle Corner	join	`Subform1.border.corner.join = "square"`
Inverted Rectangle Corner	inverted	`Subform1.border.corner.inverted = "1"`
Round Corner	join	`Subform1.border.corner.join = "round"`
Inverted Round Corner	inverted	`Subform1.border.corner.inverted = "1"`
Inverted Corner Radius	radius	`Subform1.border.corner.radius = "measurement"`

Background Fill

Property	XML Form Object Model Property	SOM Expression
None	presence	`Subform1.border.fill.presence = "invisible"`
Solid	presence	`Subform1.border.fill.presence = "visible"`
Solid - Color 1	value	`Subform1.border.fill.color.value = "R,G,B"`
Linear	type	`Subform1.border.fill.linear.type = "direction"`
Linear - Color 1	value	`Subform1.border.fill.color.value = "R,G,B"`
Linear - Color 2	value	`Subform1.border.fill.linear.color.value = "R,G,B"`
Pattern	type	`Subform1.border.fill.pattern.type = "direction"`

Property	XML Form Object Model Property	SOM Expression
Pattern - Color 1	value	*Subform1*.border.fill.color.value = "*R,G,B*"
Pattern - Color 2	value	*Subform1*.border.fill.pattern.color.value = "*R,G,B*"
Radial	type	*Subform1*.border.fill.radial.type = "*direction*"
Radial - Color 1	value	*Subform1*.border.fill.color.value = "*R,G,B*"
Radial - Color 2	value	*Subform1*.border.fill.radial.color.value = "*R,G,B*"

Object

Subform

Property	XML Form Object Model Property	SOM Expression
Type	N/A	You cannot change this property using scripting.
Flow Direction	N/A	You cannot change this property using scripting.
Allow Page Breaks within Content	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Place	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Keep w/ Previous	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Keep w/ Next	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
After	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Presence	presence	*Subform1*.presence = "*visibility*"
Locale	locale	*Subform1*.locale = "*locale*"

Binding

Property	XML Form Object Model Property	SOM Expression
Name	name	*Subform1*.name = "*objectname*"
Default Binding (Open, Save, Submit)	match	*Subform1*.bind.match = "*condition*"
Data Pattern	picture	*Subform1*.bind.picture = "*pictureformat*"
Data Format	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Import/Export Bindings (Execute)	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Repeat Subform for Each Data Item	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Min Count	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Max	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
If Data Set Must Be Paginated - Overflow Leader	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
If Data Set Must Be Paginated - Overflow Trailer	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.

33 Text Field

Layout

Size and Position

Property	XML Form Object Model Property	SOM Expression
X	x	*TextField1*.x = "*measurement*"
Y	y	*TextField1*.y = "*measurement*"
Width	w	*TextField1*.w = "*measurement*"
Height	h	*TextField1*.h = "*measurement*"
Width Expand to fit	minW	*TextField1*.minW = "*measurement*"
Height Expand to fit	minH	*TextField1*.minH = "*measurement*"
Anchor	anchorType	*TextField1*.anchorType = "*position*"

Rotate

Property	XML Form Object Model Property	SOM Expression
Remove Rotation	rotate	*TextField1*.rotate = "0"
Rotate to 90 Degrees	rotate	*TextField1*.rotate = "90"
Rotate to 180 Degrees	rotate	*TextField1*.rotate = "180"
Rotate to 270 Degrees	rotate	*TextField1*.rotate = "270"

Margins

Property	XML Form Object Model Property	SOM Expression
Left	leftInset	*TextField1*.margin.leftInset = "*measurement*"
Right	rightInset	*TextField1*.margin.rightInset = "*measurement*"
Top	topInset	*TextField1*.margin.topInset = "*measurement*"
Bottom	bottomInset	*TextField1*.margin.bottomInset = "*measurement*"

Caption

Property	XML Form Object Model Property	SOM Expression
Value	#text	*TextField1*.caption.value.#text = "*text*"
Position	placement	*TextField1*.caption.placement = "*placement*"
Reserve	reserve	*TextField1*.caption.reserve = "*measurement*"

Font

For text field objects, the Font palette applies to either the caption or the value of the field, or both.

Property	XML Form Object Model Property	SOM Expression
Font	typeface	*TextField1*.caption.font.typeface = "*font name*" *TextField1*.font.typeface = "*font name*"
Size	size	*TextField1*.caption.font.size = "*font size*" *TextField1*.font.size = "*font size*"
Baseline Shift	baselineShift	*TextField1*.caption.font.baselineShift = "*measurement*" *TextField1*.font.baselineShift = "*measurement*"

Style

Property	XML Form Object Model Property	SOM Expression
Bold	weight	*TextField1*.caption.font.weight = "*measurement*" *TextField1*.font.weight = "*measurement*"
Italic	posture	*TextField1*.caption.font.posture = "*posture*" *TextField1*.font.posture = "*posture*"
Underline	underline	*TextField1*.caption.font.underline = "*underline*" *TextField1*.font.underline = "*underline*"
Strikethrough	lineThrough	*TextField1*.caption.font.lineThrough = "*linethrough*" *TextField1*.font.lineThrough = "*linethrough*"
Color	value	*TextField1*.caption.font.fill.color.value = "*R,G,B*" *TextField1*.font.fill.color.value = "*R,G,B*"

Accessibility

Property	XML Form Object Model Property	SOM Expression
Tooltip	toolTip	*TextField1*.assist.toolTip = "*text*"
Screen Reader Precedence	priority	*TextField1*.assist.speak.priority = "*object*"
Custom Screen Reader Text	speak	*TextField1*.assist.speak = "*text*"

Borders

Property	XML Form Object Model Property	SOM Expression
Left Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Right Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Top Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Bottom Edge Style, Thickness, Color	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Rectangle Corner	join	*TextField1*.border.corner.join = "square"
Inverted Rectangle Corner	inverted	*TextField1*.border.corner.inverted = "1"
Round Corner	join	*TextField1*.border.corner.join = "round"
Inverted Round Corner	inverted	*TextField1*.border.corner.inverted = "1"
Inverted Corner Radius	radius	*TextField1*.border.corner.radius = "*measurement*"

Background Fill

Property	XML Form Object Model Property	SOM Expression
None	presence	*TextField1*.border.fill.presence = "invisible"
Solid	presence	*TextField1*.border.fill.presence = "visible"
Solid - Color 1	value	*TextField1*.border.fill.color.value = "*R,G,B*"
Linear	type	*TextField1*.border.fill.linear.type = "*direction*"
Linear - Color 1	value	*TextField1*.border.fill.color.value = "*R,G,B*"
Linear - Color 2	value	*TextField1*.border.fill.linear.color.value = "*R,G,B*"
Pattern	type	*TextField1*.border.fill.pattern.type = "*direction*"
Pattern - Color 1	value	*TextField1*.border.fill.color.value = "*R,G,B*"
Pattern - Color 2	value	*TextField1*.border.fill.pattern.color.value = "*R,G,B*"
Radial	type	*TextField1*.border.fill.radial.type = "*direction*"
Radial - Color 1	value	*TextField1*.border.fill.color.value = "*R,G,B*"
Radial - Color 2	value	*TextField1*.border.fill.radial.color.value = "*R,G,B*"

Paragraph

For text field objects, the Paragraph palette applies to either the caption or the value of the field, or both.

Property	XML Form Object Model Property	SOM Expression
Align Left	hAlign	*TextField1*.caption.para.hAlign = "left" *TextField1*.para.hAlign = "left"
Align Center	hAlign	*TextField1*.caption.para.hAlign = "center" *TextField1*.para.hAlign = "center"
Align Right	hAlign	*TextField1*.caption.para.hAlign = "right" *TextField1*.para.hAlign = "right"

Property	XML Form Object Model Property	SOM Expression
Justify	hAlign	*TextField1*.caption.para.hAlign = "justify" *TextField1*.para.hAlign = "justify"
Align Top	vAlign	*TextField1*.caption.para.vAlign = "top" *TextField1*.para.vAlign = "top"
Align Middle	vAlign	*TextField1*.caption.para.vAlign = "middle" *TextField1*.para.vAlign = "middle"
Align Bottom	vAlign	*TextField1*.caption.para.vAlign = "bottom" *TextField1*.para.vAlign = "bottom"
Indent Left	marginLeft	*TextField1*.caption.para.marginLeft = "*measurement*" *TextField1*.para.marginLeft = "*measurement*"
Indent Right	marginRight	*TextField1*.caption.para.marginRight = "*measurement*" *TextField1*.para.marginRight = "*measurement*"
Indent First	textIndent	*TextField1*.caption.para.textIndent = "*measurement*" *TextField1*.para.textIndent = "*measurement*"
Indent By	textIndent	*TextField1*.caption.para.textIndent = "*measurement*" *TextField1*.para.textIndent = "*measurement*"
Spacing Above	spaceAbove	*TextField1*.caption.para.spacingAbove = "*measurement*" *TextField1*.para.spacingAbove = "*measurement*"
Spacing Below	spaceBelow	*TextField1*.caption.para.spacingBelow = "*measurement*" *TextField1*.para.spacingBelow = "*measurement*"
Line Spacing	lineHeight	*TextField1*.caption.para.lineHeight = "*measurement*" *TextField1*.para.lineHeight = "*measurement*"

Object

Field

Property	XML Form Object Model Property	SOM Expression
Appearance	N/A	This is beyond the scope of this document. Updating these properties using scripting requires a thorough knowledge of the XML Form Object Model.
Allow Multiple Lines	multiLine	*TextField1*.ui.textEdit.multiLine = "*boolean*"

Property	XML Form Object Model Property	SOM Expression
Allow Plain Text Only	allowRichText	*TextField1*.ui.textEdit.allowRichText = "*boolean*"
Limit Length	maxChars	*TextField1*.value.text.maxChars = "*integer*"
Max Chars	maxChars	*TextField1*.value.text.maxChars = "*integer*"
Display Pattern	picture	*TextField1*.format.picture = "*pictureformat*"
Edit Pattern	picture	*TextField1*.ui.picture = "*pictureformat*"
Presence	presence	*TextField1*.presence = "*visibility*"
Locale	locale	*TextField1*.locale = "*locale*"

Value

Property	XML Form Object Model Property	SOM Expression
Default	None rawValue	*TextField1* = "*value*" (FormCalc) *TextField1*.rawValue = "*value*" (JavaScript)
Empty Message	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Validation Pattern	picture	*TextField1*.validate.picture = "*pictureformat*"
Validation Pattern Message	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Validation Pattern Message - Error	formatTest	*TextField1*.validate.formatTest = "*test*"
Validation Script Message	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Validation Script Message - Error	scriptTest	*TextField1*.validate.scriptTest = "*test*"

Binding

Property	XML Form Object Model Property	SOM Expression
Name	name	*TextField1*.name = "*objectname*"
Default Binding (Open, Save, Submit)	match	*TextField1*.bind.match = "*condition*"
Data Pattern	picture	*TextField1*.bind.picture = "*pictureformat*"
Data Format	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.
Import/Export Bindings (Execute)	N/A	This is beyond the scope of this document. Updating this property using scripting requires a thorough knowledge of the XML Form Object Model.

34 Properties

#text

A string of text.

Syntax

```
SOMexpression.#text = "text"
```

Values

Type	Values
String	• Any valid string.

access

Controls user access to the contents of a container.

Syntax

```
SOMexpression.access = "open | protected | readOnly"
```

Values

Type	Values
String	• open (default) Allows updating of a container's contents and navigation into and out of the container without restriction. • protected The application prevents the user from making any direct changes to the container's content. Indirect changes, for example calculations, can occur. The container does not participate in the tabbing sequence, though an application may allow the selection of text for clipboard copying. Protected containers do not generate any events. • readOnly The application does not allow a user to make direct changes to the container's content, but indirect changes, for example calculations, can occur. The container participates in the tabbing sequence and allows users to view the content. The user can select the container's content for clipboard copying. The container also generates a subset of events (those not associated with the user making direct changes to the content).

allowNeutral

Specifies whether the check box or radio button can support an additional third state that represents a neutral value.

Syntax

```
SOMexpression.allowNeutral = "0 | 1"
```

Values

Type	Values
String	• 0 (default) The check box or radio button supports two states representing true or false. • 1 The check box or radio button supports three states. These are true, false, or neutral.

allowRichText

Specifies whether the text may include styling.

Note: The allowRichText property only relays styling information to the application interface. The setting of this property in no way restricts a user from inputting plain text markup that includes styling information. For example, a user could type:

```
<b>hello</b>
```

regardless of the setting of this property.

Syntax

```
SOMexpression.allowRichText = "0 | 1"
```

Values

Type	Values
String	• 0 (default) Text styling is invalid. • 1 Text styling is valid.

anchorType

Specifies the location of the container's anchor point.

Syntax

```
SOMexpression.anchorType = "topLeft | topCenter | topRight | middleLeft
| middleCenter | middleRight | bottomLeft | bottomCenter | bottomRight"
```

Values

Type	Values
String	• topLeft (default) Top left corner of the container. • topCenter Center of the top edge of the container • topRight Top right corner of the container. • middleLeft Middle of the left edge of the container. • middleCenter Middle of the container. • middleRight Middle of the right edge of the container. • bottomLeft Bottom left corner of the container. • bottomCenter Center of the bottom edge of the container. • bottomRight Bottom right corner of the container.

aspect

Specifies how the image is to map to the nominal content region of the image's container.

Syntax

```
SOMexpression.aspect = "fit | none | actual"
```

Values

Type	Values
String	• fit (default) The application scales the image proportionally to the maximum size of the container's content region. • none The application scales the image to the size of entire container's content region. This may result in different scale values being applied to the image's X and Y coordinates. • actual The image renders using the dimensions stored in the image content. The extent of the container's content region plays no role in the sizing of the image.

baselineShift

Specifies a positive measurement specifying a font shift up from the baseline or negative measurement specifying a font shift down from the baseline.

Syntax

```
SOMexpression.baselineShift = "0in | measurement"
```

Values

Type	Values
String	• 0in (default) • Any valid measurement.

bottomInset

A measurement specifying the size of the bottom inset.

Syntax

```
SOMexpression.bottomInset = "0in | measurement"
```

Values

Type	Values
String	• 0in (default) • Any valid measurement.

checksum

Algorithm for the checksum to insert into the barcode. Calculation of checksums is based upon the barcode data.

Syntax

```
SOMexpression.checksum = "none | auto"
```

Values

Type	Values
String	• none (default) Do not insert a checksum. • auto Insert the default checksum.

circular

Allows for an arc to convert into a circle.

Syntax

```
SOMexpression.circular = "0 | 1"
```

Values

Type	Values
Boolean	• 0 (Default) Do not adjust the arc to a circular path. • 1 Adjust the arc to a circular path. **Note:** You can convert an arc into a circle even if the content region where the arc is located is not square. If necessary the size of the circle is adjusted to match the size of the content area.

dataLength

The maximum number of characters for this instance of the barcode.

Syntax

```
SOMexpression.dataLength = "length"
```

Values

Type	Values
String	• Default Each barcode type has its own default length value.

executeType

Specifies whether to simply import new data into your existing form or merge new data with the original form design to create a new form.

Syntax

```
SOMexpression.executeType = "import | remerge"
```

Values

Type	Values
String	• import (default) Imports data into the current form without merging that data with the form design. • remerge Merges the data in the connectionData dataset with your form design. The merge process creates dynamic subforms if necessary, depending on the data returned by the Web Service.

format

Determines the format in which to submit the data.

Syntax

```
SOMexpression.format = "xdp | formdata | pdf | xml"
```

Values

Type	Values
String	• xdp (default) The data is packaged in XDP format. • formdata The data is packaged in URL-encoded format as described in Uniform Resource Locators (URL). • pdf The data is packaged in PDF format as described in the Adobe PDF Specifications. • xml The data is packaged in XML format.

formatTest

Controls validation against the display picture clause.

Syntax

```
SOMexpression.formatTest = "warning | disabled | error"
```

Values

Type	Values
String	• warning (default) Emits a message if the data does not fit the picture clause, but a user can proceed to the next field. • disabled Do not perform any test. • error Emits a message and does not accept data that does not fit the picture clause.

h

A measurement of the height for the layout. When height is specified as a measurement, that value overrides any growth range allowed by the minH and maxH properties. When this property is omitted or set to an empty string, the growth range is set by the minH and maxH properties.

Syntax

```
SOMexpression.h = "0in | measurement"
```

Values

Type	Values
String	• 0in (default) • Any valid measurement.

hAlign

Horizontal text alignment control.

Syntax

```
SOMexpression.hAlign = "left | center | right | justifyAll | justify"
```

Values

Type	Values
String	• left (default) Align with left edge of the available region. • center Center horizontally within the available region. • right Align with right edge of the available region. • justifyAll Spread-justify all lines to fill the available region. • justify Left-align the last line and spread-justify the rest.

href

Specifies a reference to an external file or resource.

Syntax

```
SOMexpression.href = "URL"
```

Values

Type	Values
String	• A valid HTML reference. For example: http://www.adobe.com/data ftp://255.255.0.0/dataFiles

inverted

Specifies whether the corner appears convex (it joins the edges tangentially) or is inverted and appears concave (it joins the edges at right angles).

Syntax

```
SOMexpression.inverted = "0 | 1"
```

Values

Type	Values
String	• 0 (default) Corners appear convex. • 1 Corners appear concave.

join

Specifies the shape of the corner.

Syntax

```
SOMexpression.join = "square | round"
```

Values

Type	Values
String	• square (default) The corner has the shape of a right-angle between the adjoining edges. • round The corner has the shape of a round curve between the adjoining edges.

leftInset

A measurement specifying the size of the left inset.

Syntax

```
SOMexpression.leftInset = "0in | measurement"
```

Values

Type	Values
String	• 0in (default) • Any valid measurement.

lineHeight

A measurement specifying the line height to apply to the paragraph content. Omitting a value or specifying an empty value indicates that the font size determines the line height.

Syntax

```
SOMexpression.lineHeight = "0pt | measurement"
```

Values

Type	Values
String	• 0pt (default) • Any valid measurement.

lineThrough

Specifies the activation of a single or double line extending through the text (also known as strikethrough).

Syntax

```
SOMexpression.lineThrough = "0 | 1 | 2"
```

Values

Type	Values
String	• 0 (default) The font renders without a line through the text. • 1 The font renders with a single line through the text. • 2 The font renders with a double line through the text.

locale

Language, currency, and time/date formatting to use for the content of the object.

Syntax

```
SOMexpression.locale = "ambient | locale"
```

Values

Type	Values
String	• ambient (Default) Application uses its own ambient locale. • A valid locale name, for example `en_US`. For a complete list of valid locale values, refer to the IETF RFC 1766 and ISO 639/ISO 3166 specifications. If the form does not specify a locale, the locale derives from the ambient locale of the operating system. If the operating system does not supply a locale, `en_US` is used.

marginLeft

A measurement representing the left indentation of the paragraph.

Syntax

```
SOMexpression.marginLeft = "0in | measurement"
```

Values

Type	Values
String	• 0in (default) • Any valid measurement.

marginRight

A measurement representing the right indentation of the paragraph.

Syntax

```
SOMexpression.marginRight = "0in | measurement"
```

Values

Type	Values
String	• 0in (default) • Any valid measurement.

match

Controls the role played by the enclosing object in a data-binding (merge) operation.

Syntax

```
SOMexpression.match = "once | none | global | dataRef"
```

Values

Type	Values
String	• once (default) The node representing the enclosing object binds to a node in the Data DOM in accordance with the standard matching rules. • none The node representing the enclosing object is transient. It is not be bound to any node in the Data DOM. • global The containing field is global. If the normal matching rules fail to provide a match for it, the data-binding process looks outside the current record for data to bind to the field. • dataRef The containing field binds to the node in the Data DOM specified by the accompanying ref property.

maxChars

Specifies the maximum number of characters that this text value can enclose.

Syntax

```
SOMexpression.maxChars = "0 | integer"
```

Values

Type	Values
String	• 0 (default) • Any valid integer value. **Note:** If you do not specify a value for this property, or if the value is an empty string, there is no maximum.

maxH

Maximum height for layout purposes. If you do not specify a value for this property, then there is no upper limit. If you specify a value for the h property then the container is not vertically growable and this property is ignored.

Syntax

```
SOMexpression.maxH = "0in | measurement"
```

Values

Type	Values
String	• 0in (default) • Any valid measurement.

maxW

Maximum width for layout purposes. If you do not specify a value for this property, then there is no maximum. If you specify a value for the w property then the container is not horizontally growable and this property is ignored.

Syntax

```
SOMexpression.maxW = "0in | measurement"
```

Values

Type	Values
String	• 0in (default) • Any valid measurement.

minH

Minimum height for layout purposes. If you supply a value for the h property then the container is not vertically growable and this property is ignored.

Syntax

```
SOMexpression.minH = "0in | measurement"
```

Values

Type	Values
String	• 0in (default) • Any valid measurement.

minW

Minimum width for layout purposes. If you supply a value for the w property then the container is not horizontally growable and this property is ignored.

Syntax

```
SOMexpression.minW = "0in | measurement"
```

Values

Type	Values
String	• 0in (default) • Any valid measurement.

multiLine

Specifies whether the text may span multiple lines. This is useful for clients such as HTML browsers that have two types of text editing interfaces.

Syntax

```
SOMexpression.multiLine = "1 | 0"
```

Values

Type	Values
String	• 1 (default) The text may span multiple lines. • 0 The text is limited to a single line.

name

An identifier that may be used to specify this object in script expressions.

Syntax

```
SOMexpression.name = "string"
```

Values

Type	Values
String	A string up to 255 characters in length.

passwordChar

The character the form displays for each password character a user enters.

Syntax

```
SOMexpression.passwordChar = "* | character"
```

Values

Type	Values
String	• "*" (asterisk) (default) • Any valid single character.

picture

A rendering element that describes input and output formatting information.

Syntax

SOMexpression.picture = "*string*"

Values

Type	Values
String	• Any string representing a valid picture-data format.

placement

Specifies the placement of the caption.

Syntax

```
SOMexpression.placement = "left | right | top | bottom | inline"
```

Values

Type	Values
String	• left (default) Locates caption to the left of the content. • right Locates caption to the right of the content. • top Locates caption above the content. • bottom Locates caption below of the content. • inline Locates caption inline immediately prior to the content.

posture

Specifies the posture of the font.

Syntax

```
SOMexpression.posture = "normal | italic"
```

Values

Type	Values
String	• normal (default) The font has a normal posture. • italic The font is italicized.

presence

Specifies an object's visibility.

Syntax

```
SOMexpression.presence = "visible | invisible | hidden"
```

Values

Type	Values
String	• visible (default) Object is visible. • invisible Object is transparent. Although invisible, the object still takes up space. • hidden Object is hidden. The form does not display the object and the object does not take up space on the form's layout.

priority

Alters the search path for text to speak. Whichever object is named in this property moves to the front of the search path. The other objects retain their relative order.

Syntax

```
SOMexpression.priority = "custom | caption | name | tooltip"
```

Values

Type	Values
String	• custom (default) The search order is `speak`, `tooltip`, `caption`, the container's name. • caption The search order is `caption`, `speak`, `tooltip`, the container's name. • name The search order is the container's name, `speak`, `tooltip`, `caption`. • tooltip The search order is `tooltip`, `speak`, `caption`, the container's name.

radius

Specifies the radius of the corner. This property always influences the appearance of round corners, but also determines the depth of an inverted square corner. Each edge is trimmed from its end points by the corner radius, irrespective of the values of the inverted and join attributes. In general, this is of no consequence, as the corner will visibly join with the edges at their trim points. However, if the corner specifies a presence if invisible, the trimming of the edges will become apparent, even when the corner is square and not inverted.

Syntax

```
SOMexpression.radius = "0in | measurement"
```

Values

Type	Values
String	• 0in (default) • Any valid measurement.

rawValue

The actual value of a field or object.

Syntax

```
SOMexpression.rawValue = "value"
```

Values

Type	Values
String	• Any value that is valid for the type of field or object.

reserve

A measurement value that specifies the height or width of the caption.

The effect of this property is determined by the placement property. When the caption is placed at the left or right, the reserve property specifies the width of the caption region. When the caption is placed at the top or bottom, the reserve property specifies the height. When the caption is placed inline, the reserve property is ignored.

There is no meaningful default for this attribute. If you do not supply a reserve value, the height or width is determined by the content of the caption and text auto-wrapping does not occur.

Syntax

```
SOMexpression.reserve = "measurement"
```

Values

Type	Values
String	• Any valid measurement.

rightInset

A measurement specifying the size of the right inset.

Syntax

```
SOMexpression.rightInset = "0in | measurement"
```

Values

Type	Values
String	• 0in (default) • Any valid measurement.

rotate

Rotates the object about its anchor point by the specified angle.

The angle represents degrees counter-clockwise with respect to the default position. The value must be a non-negative multiple of 90.

Syntax

```
SOMexpression.rotate = "0 | measurement"
```

Values

Type	Values
String	• 0 (default) • Any valid angle measurement.

runAt

Specifies what application can execute the script. This setting is enforced even if the script is called by another script.

Syntax

```
SOMexpression.runAt = "client | server | both"
```

Values

Type	Values
String	• client (default) The script runs only on the client. • server The script runs only on the server. • both The script runs on both client and server.

scriptTest

Controls validation by the script.

Syntax

```
SOMexpression.scriptTest = "error | disabled | warning"
```

Values

Type	Values
String	• error (default) Emits a message and refuse to accept data that the script reports is erroneous (default). • disabled Do not perform this test. • warning Emits a message if the script reports the data is erroneous, but allow the user to proceed to the next field.

shape

Specifies whether the check box or radio button displays with a square or round outline.

Syntax

```
SOMexpression.shape = "square | round"
```

Values

Type	Values
String	• square (default) The button appears with a square outline. • round The button appears with a round outline.

size

A measurement specifying the size of the check box or radio button outline representing either the height and width for a check box, or the diameter for a radio button.

Syntax

```
SOMexpression.size = "10pt | measurement"
```

Values

Type	Values
String	• 10pt (default) • Any valid measurement.

spaceAbove

A measurement representing the vertical spacing in addition to the maximum font leading of the first line of the paragraph.

Syntax

```
SOMexpression.spaceAbove = "0in | measurement"
```

Values

Type	Values
String	• 0in (default) • Any valid measurement.

spaceBelow

A measurement representing the vertical spacing that appears after a paragraph.

Syntax

```
SOMexpression.spaceBelow = "0in | measurement"
```

Values

Type	Values
String	• 0in (default) • Any valid measurement.

speak

An audible prompt describing the contents of a container.

Syntax

```
SOMexpression.speak = "string"
```

Values

Type	Values
String	• Any valid string.

startAngle

Specifies the angle where the beginning of the arc renders.

Syntax

```
SOMexpression.startAngle = "0 | angle"
```

Values

Type	Values
String	• 0 (Default) • A value greater than 0 and less than or equal to 360.

stroke

Specifies the appearance of the line.

Syntax

```
SOMexpression.stroke = "solid | dashed | dotted | dashDot | dashDotDot
| lowered | raised | etched | embossed"
```

Values

Type	Values
String	• solid (default) Solid. • dashed A series of rectangular dashes. • dotted A series of round dots. • dashDot Alternating rectangular dashes and dots. • dashDotDot A series of a single rectangular dash followed by two round dots. • lowered The line appears to enclose a lowered region. • raised The line appears to enclose a raised region. • etched The line appears to be a groove lowered into the drawing surface. • embossed The line appears to be a ridge raised out of the drawing surface.

sweepAngle

Specifies the length of the arc as an angle.

Syntax

```
SOMexpression.sweepAngle = "360 |angle"
```

Values

Type	Values
String	• 360 (Default) • A value less than 360 and greater than or equal to 0.

target

The URL where data is sent.

Syntax

```
SOMexpression.target = "URL"
```

Values

Type	Values
String	• A valid URL.

textEncoding

The encoding of text content in the document.

Syntax

```
SOMexpression.textEncoding = "UTF-8 | UTF-16 | Shift-JIS | Big-Five | GB-2312"
```

Values

Type	Values
String	• UTF-8 (default) The characters are encoded using Unicode code points as defined by [Unicode], and UTF-8 serialization as defined by ISO/IEC 10646. • UTF-16 The characters are encoded using Unicode code points as defined by [Unicode], and UTF-16 serialization as defined by ISO/IEC 10646. • Shift-JIS The characters are encoded using JIS X 0208, more commonly known as Shift-JIS. • Big-Five The characters are encoded using Traditional Chinese (Big-Five). There is no official standard for Big-Five and several variants are in use. The Adobe form object model uses the variant implemented by Microsoft as code page 950. • GB-2312 The characters are encoded using Simplified Chinese.

textEntry

Determines if a user can type a value into a drop-down list.

Syntax

```
SOMexpression.textEntry = "0 | 1"
```

Values

Type	Values
String	• 0 (default) Prevents the user from typing in the current field. The value is chosen by selecting a value from the drop-down list. • 1 Allows a user to type a value into a drop-down list or select from the drop-down list.

textIndent

A measurement representing the horizontal positioning of the first line relative to the remaining lines in the paragraph. A negative value indicates a hanging indent whereas a positive value indicates first line indent.

Syntax

```
SOMexpression.textIndent = "0in | measurement"
```

Values

Type	Values
String	• 0in (default) • Any valid measurement.

textLocation

The location of any text associated with the barcode.

Syntax

```
SOMexpression.textLocation = "below | none | above | aboveEmbedded
| belowEmbedded"
```

Values

Type	Values
String	• below (default) Places text below the barcode. • none Displays no text. • above Places text above the barcode. • aboveEmbedded Partially embeds text at the top of the barcode aligned with the top of the bars. • belowEmbedded Partially embeds text at the bottom of the barcode aligned with the bottom of the bars.

thickness

Thickness or weight of the line.

Syntax

```
SOMexpression.thickness = "0.5pt | measurement"
```

Values

Type	Values
String	• 0.5pt (default) • Any valid measurement.

toolTip

Supplies text for a tool tip for a particular field or object.

Syntax

```
SOMexpression.toolTip = "text"
```

Values

Type	Values
String	• Any valid string of text.

topInset

A measurement specifying the size of the top inset.

Syntax

```
SOMexpression.topInset = "0in | measurement"
```

Values

Type	Values
String	• 0in (default) • Any valid measurement.

type

Specifies the direction of the color transition.

Syntax

```
SOMexpression.type = "toRight | toLeft | toTop | toBottom"
```

Values

Type	Values
String	• toRight (Default) The start color appears at the left side of the object and transitions into the end color at the right side. • toLeft The start color appears at the right side of the object and transitions into the end color at the left side. • toTop The start color appears at the bottom side of the object and transitions into the end color at the top side. • toBottom The start color appears at the top side of the object and transitions into the end color at the bottom side.

typeface

Specifies the name of the typeface.

Syntax

```
SOMexpression.typeface = "Courier | typeface"
```

Values

Type	Values
String	• Courier (default) • Any valid typeface identifier.

underline

Specifies the activation and type of underlining.

Syntax

```
SOMexpression.underline = "0 | 1 | 2"
```

Values

Type	Values
String	• 0 (default) The font renders without underlining. • 1 The font renders with a single underline. • 2 The font renders with a double underline.

vAlign

Vertical text alignment control.

Syntax

```
SOMexpression.vAlign = "top | middle | bottom"
```

Values

Type	Values
String	• top (default) Align with top of the available region. • middle Center vertically within the available region. • bottom Align with bottom of the available region.

value

Specifies a comma separated list of values for each color component of the color space.

Syntax

```
SOMexpression.value = "R,G,B"
```

Values

Type	Values
String	For the color-space of SRGB, the component values must be r,g,b, where r is the red component value, g is the green component value, and b is the blue component value. Each component value must be in the range 0 through 255, inclusive. 255 represents maximum display intensity. For example, 255,0,0 specifies the color red. The default is dependent upon the context of where the color is used; the default color is determined by the object enclosing the color object.

w

A measurement of the width for the layout. When you specify a width, that value overrides any growth range specified by the minW and maxW properties. Omitting this property or specifying an empty string indicates that the minW and maxW properties define the width for the object.

Syntax

```
SOMexpression.w = "0in | measurement"
```

Values

Type	Values
String	• 0in (default) • Any valid measurement.

weight

Controls the weight of the font typeface.

Syntax

```
SOMexpression.weight = "bold | normal"
```

Values

Type	Values
String	• bold (default) The typeface renders with a bold typeface. • normal The typeface renders at the default typeface weight.

x

X coordinate of the container's anchor point.

Syntax

```
SOMexpression.x = "0in | measurement"
```

Values

Type	Values
String	• 0in (default) • Any valid measurement value.

xdpContent

Controls what subset of the data is submitted. This property is used only when the format property is `xdp`.

Syntax

```
SOMexpression.xdpContent = "0in | measurement"
```

Values

Type	Values
String	• datasets pdf (default) Submits elements with the tags datasets, and pdf to the host. • datasets pdf xfdf Submits the default content types as well as annotations. • datasets pdf xfdf signature Submits the default content types, annotations, and digital signatures. • datasets pdf xfdf signature template Submits the default content types, annotations, digital signatures, and the form design. • * (asterisk) Submits all data objects to the host.

y

Y coordinate of the container's anchor point.

Syntax

```
SOMexpression.y = "0in | measurement"
```

Values

Type	Values
String	• 0in (default) • Any valid measurement value.

35 Methods

addItem

Adds an item to the list of items contained within a drop-down list or a list box.

Syntax

SOM_expression.addItem(STRING *param1* [, STRING *param2*])

Parameters

param1	The string representing the display value.
param2 (Optional)	The string representing the bound value.

Returns

Empty

Examples

- DropDownList1.addItem("Human Resources")
- DropDownList1.additem("Human Resources", "10")
- ListBox1.addItem("Yellow")
- ListBox1.addItem("Yellow", "2")
- ListBox1.addItem(DropDownList1.rawValue, "3")

clearItems

Remove all the items contained within a drop-down list or a list box.

Syntax

SOM_expression.clearItems( )

Parameters

None

Returns

Empty

Examples

- DropDownList1.clearItems()
- ListBox1.clearItems()

Index

D

E

F

H

I

J

L

M

T

U

V

W

X

Y