

# 5

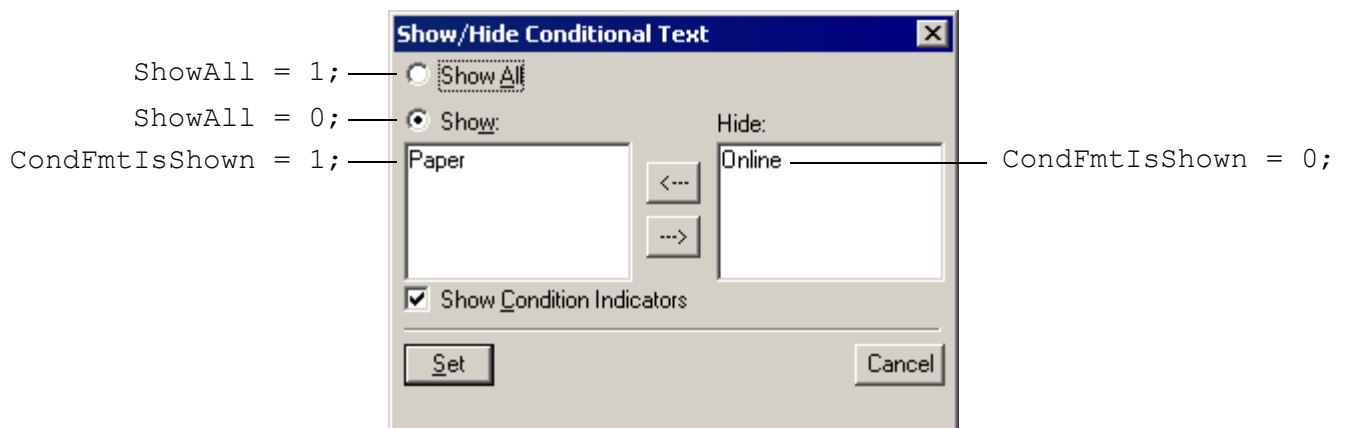
## Working With Text

---

### Conditional Text

#### Showing and Hiding Conditions

Each condition format in a FrameMaker document has a **CondFmtIsShown** property that determines the visibility of the condition in the document. This property must be used in conjunction with the document's **ShowAll** property. If the document's **ShowAll** property is set to 1 (or **True**), all conditions will be shown, regardless of their **CondFmtIsShown** value. Therefore, to hide a condition, set the document's **ShowAll** property to 0 (or **False**), and the condition's **CondFmtIsShown** property to 0. The screenshot below shows how each FrameScript property corresponds to settings in the FrameMaker interface.



The following script hides the Online condition format in the active document.

#### Code Listing 5-86

---

```
// Set a variable for the active document.
Set vCurrentDoc = ActiveDoc;

// Set the document's ShowAll property to 0.
Set vCurrentDoc.ShowAll = 0;

// Get the condition format object.
Get Object Type(CondFmt) Name('Online') DocObject(vCurrentDoc)
  NewVar(vCondFmt);

// Hide the condition format.
Set vCondFmt.CondFmtIsShown = 0;
```

Remember that **ShowAll** is a **Doc** property, while **CondFmtIsShown** is a **CondFmt** property.

## Showing and Hiding Condition Indicators

Condition formats can have colors and formatting to distinguish them in a document; these are known as “condition indicators.” You can show or hide condition indicators by setting the document’s **ShowCondIndicators** property. This is the same as checking or unchecking the Show Condition Indicators checkbox in the Show/Hide Conditional Text dialog box (see previous screenshot). This script shows the condition indicators in the active document.

### Code Listing 5-87

---

```
// Set a variable for the active document.
Set vCurrentDoc = ActiveDoc;

// Show the condition indicators in the document.
Set vCurrentDoc.ShowCondIndicators = 1;
```

## Applying Condition Formats

You can use the **Apply TextProperties** command to apply a **CondFmt** to a range of text. Use the **CondFmt** parameter with the name of the condition format. The following code applies the “Online” condition to the paragraph containing the insertion point.

### Code Listing 5-88

---

```
// Make a variable for the active document.
Set vCurrentDoc = ActiveDoc;

// Set a variable for the paragraph containing the insertion point.
Set vPgf = TextSelection.Begin.Object;

// Make a text range for the entire paragraph.
New TextRange NewVar(vTextRange) Object(vPgf) Offset(0)
  Offset(ObjEndOffset);

// Apply the Online condition format to the text range.
Apply TextProperties CondFmt('Online') TextRange(vTextRange);
```

You can use the **CondFmt** parameter more than once to apply multiple conditions to a text range. If you use a null string on the **CondFmt** parameter, all condition formats will be removed from the text range.

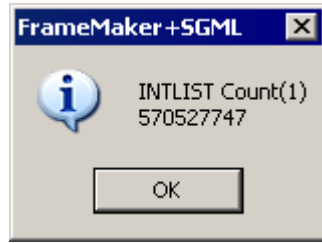
The **Apply TextProperties** command is limited in that it does not preserve any existing conditions that may be applied to the text range. For example, if you have “Condition1” applied to a range of text and use the **Apply TextProperties** command to apply “Condition2” to the same text, it will remove the “Condition1” format from the text.

You can overcome this limitation by manipulating the text’s **InCond** property. The **InCond** property is an integer list of condition format integers that are applied to text. Table rows also have an **InCond** property that indicates which conditions are applied to the row. To see the **InCond** property, apply a condition format to a word in a document, select the word and run this code.

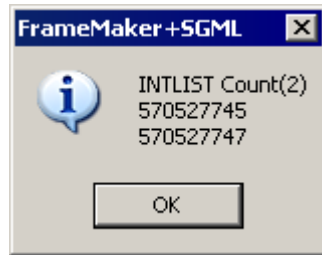
#### Code Listing 5-89

---

```
// See the InCond property of the selected text.  
Display TextSelection.Begin.InCond;
```

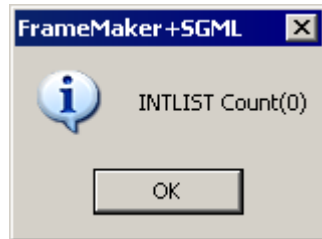


Now apply a second condition format to the word and rerun the code.



The numbers you will see will differ from those in the screenshots. These numbers are integer representations of **CondFmt** objects. You will see how to identify the condition formats represented by these numbers shortly.

If you run the code on text that has no condition formats applied, the **InCond** integer list will be 0 (empty).



To convert the integers to objects, you must first extract them from the integer list. Then you use the **New Object** command to convert each integer to a **CondFmt** object. Select the text that is formatted with two conditions and run the following script.

#### Code Listing 5-90

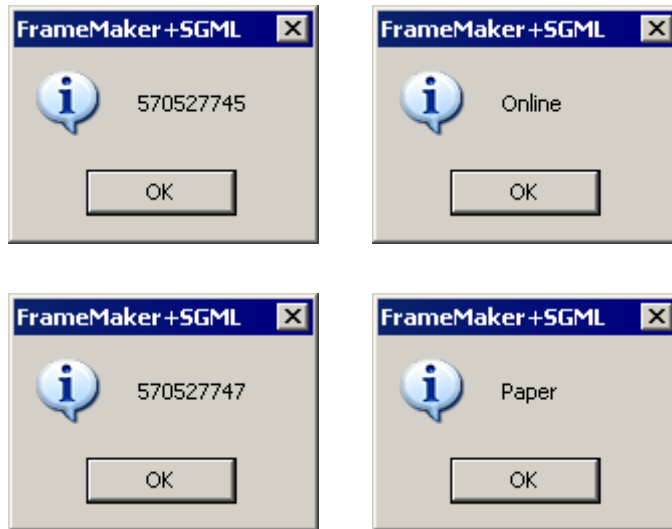
---

```
// Set a variable for the active document.  
Set vCurrentDoc = ActiveDoc;  
  
// Set a variable for the InCond list at the text selection.  
Set vInCond = TextSelection.Begin.InCond;
```

```

// Loop through the InCond list of the selected text.
Loop While(n <= vInCond.Count) LoopVar(n) Init(1) Incr(1)
  Get Member Number(n) From(vInCond) NewVar(vCondInt);
  Display vCondInt; // Display the integer.
  // Make an condition format object from the integer.
  New Object IntValue(vCondInt) DocObject(vCurrentDoc)
    NewVar(vCondFmt);
  Display vCondFmt.Name; // Display the condition format name.
EndLoop

```



You can use this method to identify condition formats that are applied at a text location. You can also use this with a Row's **InCond** property to identify conditions that are applied to a table row.

To apply conditions to text or table rows, you work in the opposite direction; you start with the **CondFmt** object, convert it to an integer, add it to an integer list, and apply the list to the **InCond** property. Here is a script that illustrates this by applying a single condition to the selected text in the active document. *The following two example scripts do not attempt to preserve existing conditions that may be applied to the text or table rows.* See "Preserving Existing Conditions" on page 5-57.

#### Code Listing 5-91

---

```

// Make a variable for the active document.
Set vCurrentDoc = ActiveDoc;

// Make an integer list.
New IntList NewVar(vInCond);

// Get the Online condition format object.
Get Object Type(CondFmt) Name('Online') DocObject(vCurrentDoc)
  NewVar(vCondFmt);
// Convert the condition format object to an integer.
New Integer NewVar(vCondInt) IntValue(vCondFmt);

// Add the integer to the integer list.
Add Member(vCondInt) To(vInCond);

```

```
// Make a property list containing the vInCond list.
New PropertyList NewVar(vProps) InCond(vInCond);

// Apply the properties to the text selection.
Apply TextProperties TextRange(TextSelection) Properties(vProps);
```

For table rows, you use the **Row's InCond** property to apply conditions to the row. The following example, applies two conditions to the last row in the selected table.

#### Code Listing 5-92

---

```
// Make a variable for the active document.
Set vCurrentDoc = ActiveDoc;

// Set a variable for the selected table.
Set vTbl = vCurrentDoc.SelectedTbl;

// Make an integer list.
New IntList NewVar(vInCond);

// Get the PaperUS condition object.
Get Object Type(CondFmt) Name('PaperUS') DocObject(vCurrentDoc)
  NewVar(vCondFmt);
// Convert the condition format object to an integer.
New Integer NewVar(vCondInt) IntValue(vCondFmt);
// Add the integer to the integer list.
Add Member(vCondInt) To(vInCond);

// Get the PaperUK condition object.
Get Object Type(CondFmt) Name('PaperUK') DocObject(vCurrentDoc)
  NewVar(vCondFmt);
// Convert the condition format object to an integer.
New Integer NewVar(vCondInt) IntValue(vCondFmt);
// Add the integer to the integer list.
Add Member(vCondInt) To(vInCond);

// Apply the list to the last row in the table.
Set vTbl.LastRowInTbl.InCond = vInCond;
```

**IMPORTANT:** The previous code examples leave out some important tests: there is no test for an active document, a selected table, and we did not test for the existence of the the condition formats. Make sure you have sufficient testing in your production scripts.

### Preserving Existing Conditions

The previous two examples used the **InCond** property to apply conditions to text and table rows. *However, existing conditions would be overwritten using those scripts.* To preserve existing conditions, you must add the new conditions to the existing ones. We will demonstrate this first with table rows. If you want to try this, follow these steps.

1. Make a new FrameMaker document.
2. Create two new conditions, Condition1 and Condition2. Make sure you assign a different color to each condition.
3. Insert a Format A table.

4. Apply Condition1 to the first row in the table.
5. Click in the table and run the following script.

#### Code Listing 5-93

---

```
// Make a variable for the active document.
Set vCurrentDoc = ActiveDoc;

// Get the Condition2 object.
Get Object Type(CondFmt) Name('Condition2') DocObject(vCurrentDoc)
  NewVar(vCondFmt);
If vCondFmt = 0
  // If the condition doesn't exist, warn the user and exit.
  MsgBox 'Condition2 format does not exist in this document.  ';
  LeaveSub;
EndIf

// Convert the condition object to an integer.
New Integer NewVar(vCondInt) IntValue(vCondFmt);

// Set a variable for the selected table.
Set vTbl = vCurrentDoc.SelectedTbl;

// Set a variable for the first row in the table.
Set vRow = vTbl.FirstRowInTbl;

// Make a variable for the current conditions in the row.
Set vInCond = vRow.InCond;

// See if the Condition2 integer is already in the list.
Find Member(vCondInt) InList(vInCond) ReturnStatus(vFound);
If vFound = 0
  // If the integer is not already in the list, add it.
  Add Member(vCondInt) To(vInCond);
EndIf

// Apply the list to the table row.
Set vRow.InCond = vInCond;
```

You should see Condition2 applied to the table row in addition to the existing Condition1. If you get an error, make sure your cursor is in the table and run the script again.

Preserving existing conditions in text is a little more complicated because existing conditions may be applied anywhere in the text. You must use the **Get TextList** command to identify and preserve existing conditions in the text range or object that you are applying the new condition to. To test the next script, follow these steps.

1. Make a new FrameMaker document.
2. Create two conditions in the document: Condition1 and Condition2. Make sure you assign a different color to each condition.
3. Enter a paragraph of text with a few sentences.
4. Apply Condition1 to a word or two in the middle of the paragraph.

The first example will apply the Condition2 format to the entire paragraph, while preserving the Condition1 format that is already applied. Make sure you click in the paragraph before running the script.

#### Code Listing 5-94

---

```
// Set a variable for the active document.
Set vCurrentDoc = ActiveDoc;

// Set a variable for the paragraph containing the insertion point.
Set vPgf = TextSelection.Begin.Object;

// Get a list of character property changes in the paragraph.
Get TextList InObject(vPgf) CharPropsChange NewVar(vTextList);

// Set a variable for the offset where conditional text changes.
Set vCondOffset = 0;

// Loop through the text list.
Loop While(n <= vTextList.Count) LoopVar(n) Init(1) Incr(1)
  Get Member Number(n) From(vTextList) NewVar(vCharPropsChange);
  // See if the property change is a condition format.
  Find Member('CONDITIONTAG') InList(vCharPropsChange.TextData)
  ReturnStatus(vFound);
  If vFound
    // Make a text range for the existing condition.
    New TextRange NewVar(vTextRange) Object(vPgf)
      Offset(vCondOffset) Offset(vCharPropsChange.TextOffset);
    // Write the text to the Console window.
    Write Console vTextRange.Text;
    // Set the variable indicating the start offset.
    Set vCondOffset = vCharPropsChange.TextOffset;
  EndIf
EndLoop

// Make a text range including the end of the paragraph.
New TextRange NewVar(vTextRange) Object(vPgf) Offset(vCondOffset)
  Offset(ObjEndOffset);
// Write the text to the Console window.
Write Console vTextRange.Text;
```

Look at the Console window. At this point, the script simply divides the paragraph up into separate text ranges for each condition format combination, including text that has no condition applied. If you run the script on a paragraph that has no conditions applied, the entire paragraph will be written to the Console. We now have a mechanism to apply the new condition to each text range, while preserving any existing conditions. We will use a subroutine to apply the new condition to the current text range. Here is the finished script.

#### Code Listing 5-95

---

```
// Set a variable for the active document.
Set vCurrentDoc = ActiveDoc;

// Set a variable for the paragraph containing the insertion point.
Set vPgf = TextSelection.Begin.Object;
```

```

// Get the Condition2 object.
Get Object Type(CondFmt) Name('Condition2') DocObject(vCurrentDoc)
  NewVar(vCondFmt);
If vCondFmt = 0
  // If the condition doesn't exist, warn the user and exit.
  MsgBox 'Condition2 format does not exist in this document.  ';
  LeaveSub;
EndIf

// Convert the condition object to an integer.
New Integer NewVar(vCondInt) IntValue(vCondFmt);

// Get a list of character property changes in the paragraph.
Get TextList InObject(vPgfl) CharPropsChange NewVar(vTextList);

// Set a variable for the offset where conditional text changes.
Set vCondOffset = 0;

// Loop through the text list.
Loop While(n <= vTextList.Count) LoopVar(n) Init(1) Incr(1)
  Get Member Number(n) From(vTextList) NewVar(vCharPropsChange);
  // See if the property change is a condition format.
  Find Member('CONDITIONTAG') InList(vCharPropsChange.TextData)
  ReturnStatus(vFound);
  If vFound
    // Make a text range for the existing condition.
    New TextRange NewVar(vTextRange) Object(vPgfl)
      Offset(vCondOffset) Offset(vCharPropsChange.TextOffset);
    // Run a subroutine to apply the new condition.
    Run ApplyCondition;
    // Set the variable indicating the start offset.
    Set vCondOffset = vCharPropsChange.TextOffset;
  EndIf
EndLoop

// Make a text range including the end of the paragraph.
New TextRange NewVar(vTextRange) Object(vPgfl) Offset(vCondOffset)
  Offset(ObjEndOffset);
// Run a subroutine to apply the new condition.
Run ApplyCondition;

Sub ApplyCondition
//
// Set a variable for the current InCond setting for the text range.
Set vInCond = vTextRange.Begin.InCond;
// See if the new condition integer is already in the list.
Find Member(vCondInt) InList(vInCond) ReturnStatus(vFound);
If vFound = 0
  // If the condition is not in the list, add it.
  Add Member(vCondInt) To(vInCond);
  // Make a property list containing the updated integer list.
  New PropertyList NewVar(vProps) InCond(vInCond);
  // Apply the updated integer list to the text range.
  Apply TextProperties TextRange(vTextRange) Properties(vProps);
EndIf
//
EndSub

```

## Removing Conditions from Text and Table Rows

Removing *all* conditions from text is simple; use the **Apply TextProperties** command with a null string in the **CondFmt** parameter.

### Code Listing 5-96

---

```
// Remove ALL conditions from the selected text.  
Apply TextProperties CondFmt('') TextRange(TextSelection);
```

You use a similar technique to remove *all* conditions from a table row. The script below removes all condition formats from all of the rows in the selected table.

### Code Listing 5-97

---

```
// Set a variable for the active document.  
Set vCurrentDoc = ActiveDoc;  
  
// Make a variable for the selected table.  
Set vTbl = SelectedTbl;  
  
// Get the Condition2 object.  
Get Object Type(CondFmt) Name('Condition2') DocObject(vCurrentDoc)  
  NewVar(vCondFmt);  
If vCondFmt = 0  
  // If the condition doesn't exist, warn the user and exit.  
  MsgBox 'Condition2 format does not exist in this document.  ';  
  LeaveSub;  
EndIf  
  
// Convert the condition object to an integer.  
New Integer NewVar(vCondInt) IntValue(vCondFmt);  
  
// Loop through the table rows.  
Loop ForEach(Row) In(vTbl) LoopVar(vRow)  
  // Make a variable for the row's InCond integer list.  
  Set vInCond = vRow.InCond;  
  // See if Condition2 is in the list.  
  Find Member(vCondInt) InList(vInCond) ReturnStatus(vFound);  
  If vFound  
    // Remove the member from the list.  
    Remove Member(vCondInt) From(vInCond);  
    // Apply the modified list to the table row.  
    Set vRow.InCond = vInCond;  
  EndIf  
EndLoop
```

## Preserving Existing Conditions

You can remove condition formats while preserving existing ones. You do this by manipulating the **InCond** property of the text or table rows. Here is a script that removes a condition from the table rows in the selected table and preserves existing conditions.

### Code Listing 5-98

---

```
// Set a variable for the active document.  
Set vCurrentDoc = ActiveDoc;
```

```

// Make a variable for the selected table.
Set vTbl = SelectedTbl;

// Get the Condition2 object.
Get Object Type(CondFmt) Name('Condition2') DocObject(vCurrentDoc)
  NewVar(vCondFmt);
If vCondFmt = 0
  // If the condition doesn't exist, warn the user and exit.
  MsgBox 'Condition2 format does not exist in this document.  ';
  LeaveSub;
EndIf

// Convert the condition object to an integer.
New Integer NewVar(vCondInt) IntValue(vCondFmt);

// Loop through the table rows.
Loop ForEach(Row) In(vTbl) LoopVar(vRow)
  // Make a variable for the row's InCond integer list.
  Set vInCond = vRow.InCond;
  // See if Condition2 is in the list.
  Find Member(vCondInt) InList(vInCond) ReturnStatus(vFound);
  If vFound
    // Remove the member from the list.
    Remove Member(vCondInt) From(vInCond);
    // Apply the modified list to the table row.
    Set vRow.InCond = vInCond;
  EndIf
EndLoop

```

When removing a condition from text, we can use the same technique employed in Code Listing 5-95 on page 5-59. In fact, everything in the following script is the same, except the subroutine and the two lines calling the subroutine.

#### Code Listing 5-99

---

```

// Set a variable for the active document.
Set vCurrentDoc = ActiveDoc;

// Set a variable for the paragraph containing the insertion point.
Set vPgfl = TextSelection.Begin.Object;

// Get the Condition2 object.
Get Object Type(CondFmt) Name('Condition2') DocObject(vCurrentDoc)
  NewVar(vCondFmt);
If vCondFmt = 0
  // If the condition doesn't exist, warn the user and exit.
  MsgBox 'Condition2 format does not exist in this document.  ';
  LeaveSub;
EndIf

// Convert the condition object to an integer.
New Integer NewVar(vCondInt) IntValue(vCondFmt);

// Get a list of character property changes in the paragraph.
Get TextList InObject(vPgfl) CharPropsChange NewVar(vTextList);

```

```

// Set a variable for the offset where conditional text changes.
Set vCondOffset = 0;

// Loop through the text list.
Loop While(n <= vTextList.Count) LoopVar(n) Init(1) Incr(1)
Get Member Number(n) From(vTextList) NewVar(vCharPropsChange);
// See if the property change is a condition format.
Find Member('CONDITIONTAG') InList(vCharPropsChange.TextData)
ReturnStatus(vFound);
If vFound
// Make a text range for the existing condition.
New TextRange NewVar(vTextRange) Object(vPgfl)
Offset(vCondOffset) Offset(vCharPropsChange.TextOffset);
// Run a subroutine to apply the new condition.
Run RemoveCondition;
// Set the variable indicating the start offset.
Set vCondOffset = vCharPropsChange.TextOffset;
EndIf
EndLoop

// Make a text range including the end of the paragraph.
New TextRange NewVar(vTextRange) Object(vPgfl) Offset(vCondOffset)
Offset(ObjEndOffset);
// Run a subroutine to apply the new condition.
Run RemoveCondition;

Sub RemoveCondition
//
// Set a variable for the current InCond setting for the text range.
Set vInCond = vTextRange.Begin.InCond;
// See if the condition integer is in the list.
Find Member(vCondInt) InList(vInCond) ReturnStatus(vFound);
If vFound
// If the condition is not in the list, add it.
Remove Member(vCondInt) From(vInCond);
// Make a property list containing the updated integer list.
New PropertyList NewVar(vProps) InCond(vInCond);
// Apply the updated integer list to the text range.
Apply TextProperties TextRange(vTextRange) Properties(vProps);
EndIf
//
EndSub

```

## Creating Condition Formats

FrameScript allows you to create condition formats by using the **New ConditionFormat** command. You supply the name of the condition format in the **Name** parameter. You should check to see if the format already exists before creating it. The following script creates a condition called "SecondEdition" in the active document.

### Code Listing 5-100

---

```

// Set a variable for the active document.
Set vCurrentDoc = ActiveDoc;

```

```

// See if the condition format already exists.
Get Object Type(CondFmt) Name('SecondEdition')
  DocObject(vCurrentDoc) NewVar(vCondFmt);
If vCondFmt
  LeaveSub; // Exit the script.
EndIf

// Make the condition format.
New ConditionFormat Name('SecondEdition') DocObject(vCurrentDoc);

```

Condition formats usually have a character style and a color to identify them in the document. You can set these properties after you create the condition format, or you can change them in an existing format. The following script creates a condition, and then sets its style to Strikethrough and its color to Green.

---

#### Code Listing 5-101

```

// Set a variable for the active document.
Set vCurrentDoc = ActiveDoc;

// See if the condition format already exists.
Get Object Type(CondFmt) Name('SecondEdition')
  DocObject(vCurrentDoc) NewVar(vCondFmt);
If vCondFmt = 0
  // If it doesn't exist, create the format.
  New ConditionFormat Name('SecondEdition') DocObject(vCurrentDoc)
    NewVar(vCondFmt);
EndIf

// Set the condition's style to Strikethrough.
Set vCondFmt.StyleOverride = CnStrikeThrough;

// Set the condition's color to Green.
Get Object Type(Color) Name('Green') DocObject(vCurrentDoc)
  NewVar(vColor);
Set vCondFmt.SepOverride = vColor;

```

Note that we didn't have to test for the existence of the Green color, because Green is a reserved color that exists in every FrameMaker document. If you are using a non-reserved color, make sure you test for its existence in the document before attempting to use it.

## Deleting Condition Formats

FrameScript allows you to delete condition formats by getting the `CondFmt` object and deleting it.

---

#### Code Listing 5-102

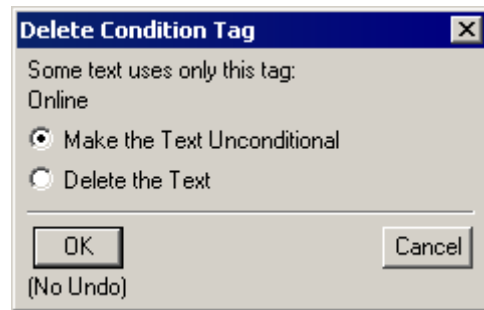
```

// Set a variable for the active document.
Set vCurrentDoc = ActiveDoc;

```

```
// Get the condition format object.  
Get Object Type(CondFmt) Name('PaperUK') DocObject(vCurrentDoc)  
  NewVar(vCondFmt);  
If vCondFmt  
  // Delete the condition format.  
  Delete Object(vCondFmt);  
EndIf
```

When you delete a condition format in the FrameMaker interface, you will get the Delete Condition Tag dialog box if some of the text or table rows in the document is formatted with only that condition. This gives you a choice to keep the text and make it unconditional, or to delete the conditional text and table rows along with the condition.



When you delete a condition format with FrameScript, *any text or table rows with that condition format applied is deleted*. The only exception is text or table rows that have additional conditions applied to it. If you want to delete the condition but keep the text and table rows, you must remove the condition format from the text and table rows before you delete the condition. See “Removing Conditions from Text and Table Rows” on page 5-61.

