Contents

About This Book ........................................................................................................................... ix

CHAPTER 1  DataWindow Operators and Expressions ................................. 1
  Where you use DataWindow expressions ................................. 1
  Operators used in DataWindow expressions ......................... 4
    Arithmetic operators in DataWindow expressions ............... 5
    Relational operators in DataWindow expressions ............... 5
    Logical operators in DataWindow expressions .................. 9
    Concatenation operator in DataWindow expressions .......... 10
  Operator precedence in DataWindow expressions ................ 11
  Evaluating DataWindow expressions in code ...................... 11
  Evaluating conditional DataWindow expressions with current data 12

CHAPTER 2  DataWindow Expression Functions ............................................ 13
  Using DataWindow expression functions ............................. 13
  Four examples ......................................................................... 14
    Example 1: counting null values in a column ................... 14
    Example 2: counting male and female employees ............. 16
    Example 3: creating a row indicator ................................. 19
    Example 4: displaying all data when a column allows nulls ... 21
  Alphabetical list of DataWindow expression functions .......... 23
    Abs .................................................................................. 24
    ACos .............................................................................. 24
    Asc ................................................................................ 25
    AscA .............................................................................. 25
    ASin .............................................................................. 26
    ATan ............................................................................... 27
    Avg .............................................................................. 27
    Bitmap ............................................................................ 30
    Case ............................................................................... 31
    Ceiling .......................................................................... 32
    Char .............................................................................. 33
    CharA ............................................................................ 33
Contents

Cos ................................................................. 34
Count ............................................................ 34
CrosstabAvg ..................................................... 36
CrosstabAvgDec ............................................... 40
CrosstabCount .................................................. 41
CrosstabMax ...................................................... 43
CrosstabMaxDec ............................................... 44
CrosstabMin ...................................................... 45
CrosstabMinDec ............................................... 47
CrosstabSum ...................................................... 48
CrosstabSumDec ............................................... 50
CumulativePercent ............................................ 51
CumulativeSum .................................................. 53
CurrentRow ...................................................... 54
Date ............................................................... 55
DateTime .......................................................... 56
Day ................................................................. 57
DayName .......................................................... 57
DayNumber ...................................................... 58
DaysAfter ......................................................... 59
Dec ................................................................. 59
Describe .......................................................... 60
Exp ................................................................. 61
Fact ................................................................. 61
Fill ................................................................. 62
FillA ............................................................... 63
First ............................................................... 63
GetRow ............................................................ 65
GetText ............................................................ 66
Hour ............................................................... 66
If ................................................................. 67
Int ................................................................. 68
Integer ............................................................ 68
IsDate .............................................................. 69
IsExpanded ...................................................... 70
IsNull ............................................................. 70
IsNumber ........................................................ 71
IsRowModified .................................................. 71
IsRowNew ........................................................ 72
IsSelected ....................................................... 72
IsTime ............................................................. 73
Large ............................................................. 74
Last ............................................................... 76
LastPos ........................................................... 77
<table>
<thead>
<tr>
<th>Function</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Left</td>
<td>78</td>
</tr>
<tr>
<td>LeftA</td>
<td>79</td>
</tr>
<tr>
<td>LeftTrim</td>
<td>80</td>
</tr>
<tr>
<td>Len</td>
<td>80</td>
</tr>
<tr>
<td>LenA</td>
<td>81</td>
</tr>
<tr>
<td>Log</td>
<td>81</td>
</tr>
<tr>
<td>LogTen</td>
<td>82</td>
</tr>
<tr>
<td>Long</td>
<td>82</td>
</tr>
<tr>
<td>LookUpDisplay</td>
<td>83</td>
</tr>
<tr>
<td>Lower</td>
<td>84</td>
</tr>
<tr>
<td>Match</td>
<td>84</td>
</tr>
<tr>
<td>Max</td>
<td>87</td>
</tr>
<tr>
<td>Median</td>
<td>89</td>
</tr>
<tr>
<td>Mid</td>
<td>91</td>
</tr>
<tr>
<td>MidA</td>
<td>92</td>
</tr>
<tr>
<td>Min</td>
<td>92</td>
</tr>
<tr>
<td>Minute</td>
<td>94</td>
</tr>
<tr>
<td>Mod</td>
<td>95</td>
</tr>
<tr>
<td>Mode</td>
<td>95</td>
</tr>
<tr>
<td>Month</td>
<td>98</td>
</tr>
<tr>
<td>Now</td>
<td>98</td>
</tr>
<tr>
<td>Number</td>
<td>99</td>
</tr>
<tr>
<td>Page</td>
<td>100</td>
</tr>
<tr>
<td>PageAbs</td>
<td>100</td>
</tr>
<tr>
<td>PageAcross</td>
<td>101</td>
</tr>
<tr>
<td>PageCount</td>
<td>101</td>
</tr>
<tr>
<td>PageCountAcross</td>
<td>102</td>
</tr>
<tr>
<td>Percent</td>
<td>103</td>
</tr>
<tr>
<td>PI</td>
<td>105</td>
</tr>
<tr>
<td>Pos</td>
<td>106</td>
</tr>
<tr>
<td>PosA</td>
<td>107</td>
</tr>
<tr>
<td>ProfileInt</td>
<td>107</td>
</tr>
<tr>
<td>ProfileString</td>
<td>109</td>
</tr>
<tr>
<td>Rand</td>
<td>110</td>
</tr>
<tr>
<td>Real</td>
<td>110</td>
</tr>
<tr>
<td>RelativeDate</td>
<td>111</td>
</tr>
<tr>
<td>RelativeTime</td>
<td>111</td>
</tr>
<tr>
<td>Replace</td>
<td>112</td>
</tr>
<tr>
<td>ReplaceA</td>
<td>113</td>
</tr>
<tr>
<td>RGB</td>
<td>113</td>
</tr>
<tr>
<td>Right</td>
<td>115</td>
</tr>
<tr>
<td>RightA</td>
<td>115</td>
</tr>
<tr>
<td>RightTrim</td>
<td>116</td>
</tr>
<tr>
<td>Round</td>
<td>116</td>
</tr>
</tbody>
</table>
CHAPTER 3  DataWindow Object Properties ................................. 143
  Overview of DataWindow object properties ............................. 143
  Controls in a DataWindow and their properties ...................... 145
    Properties for the DataWindow object .................................. 145
    Properties for Button controls in DataWindow objects .......... 154
    Properties for Column controls in DataWindow objects ....... 155
    Properties for Computed Field controls in DataWindow objects ......................................................... 156
    Properties for Graph controls in DataWindow objects ........ 157
    Properties for GroupBox controls in DataWindow objects ... 159
    Properties for the Group keyword ....................................... 160
    Properties for InkPicture controls in DataWindow objects .... 160
    Properties for Line controls in DataWindow objects .......... 161
    Properties for OLE Object controls in DataWindow objects .. 161
    Properties for Oval, Rectangle, and RoundRectangle controls in DataWindow objects ...................... 162
    Properties for Picture controls in DataWindow objects ....... 163
    Properties for Report controls in DataWindow objects ....... 164
    Properties for the Style keyword ....................................... 165
    Properties for TableBlob controls in DataWindow objects .... 165
Contents

Properties for Text controls in DataWindow objects.............. 166
Title keyword ......................................................................... 167
Alphabetical list of DataWindow object properties ................. 167

Index .................................................................................................................. 365
About This Book

Subject
This book provides reference information for the DataWindow® object. It describes DataWindow expressions and the functions you use with them, and properties of DataWindow objects.

Audience
This book is for anyone developing applications that use DataWindow .NET™ in the .NET Framework. It assumes that:

- You are familiar with the DataWindow painter. If not, see the DataWindow Designer User's Guide.
- You are an experienced user of Visual Basic or C# and your development environment. If not, see the documentation for your development environment and your language of choice.

Related books
For a complete list of books and online documentation, see the preface of the DataWindow .NET Programmer's Guide.

Code samples
You can find sample code on the Sybase CodeXchange Web site at http://datawindownet.codexchange.sybase.com and in the Sybase\DataWindow .NET 2.5\Code Examples directory.

Other sources of information
Use the Sybase Getting Started CD, the SyBooks CD, and the Sybase Product Manuals Web site to learn more about your product:

- The Getting Started CD contains release bulletins and installation guides in PDF format, and may also contain other documents or updated information not included on the SyBooks CD. It is included with your software. To read or print documents on the Getting Started CD, you need Adobe Acrobat Reader, which you can download at no charge from the Adobe Web site using a link provided on the CD.
- The SyBooks CD contains product manuals and is included with your software. The Eclipse-based SyBooks browser allows you to access the manuals in an easy-to-use, HTML-based format.

Some documentation may be provided in PDF format, which you can access through the PDF directory on the SyBooks CD. To read or print the PDF files, you need Adobe Acrobat Reader.
Refer to the *SyBooks Installation Guide* on the Getting Started CD, or the *README.txt* file on the SyBooks CD for instructions on installing and starting SyBooks.

- The Sybase Product Manuals Web site is an online version of the SyBooks CD that you can access using a standard Web browser. In addition to product manuals, you will find links to EBFs/Maintenance, Technical Documents, Case Management, Solved Cases, newsgroups, and the Sybase Developer Network.

  To access the Sybase Product Manuals Web site, go to Product Manuals at [http://www.sybase.com/support/manuals/](http://www.sybase.com/support/manuals/).

**If you need help**

Each Sybase installation that has purchased a support contract has one or more designated people who are authorized to contact Sybase Technical Support. If you cannot resolve a problem using the manuals or online help, please have the designated person contact Sybase Technical Support or the Sybase subsidiary in your area.
CHAPTER 1

DataWindow Operators and Expressions

About this chapter
You use an expression to request that a DataWindow object perform a computational operation. This chapter explains how expressions work and how to write them.

Contents

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Where you use DataWindow expressions</td>
<td>1</td>
</tr>
<tr>
<td>Operators used in DataWindow expressions</td>
<td>4</td>
</tr>
<tr>
<td>Operator precedence in DataWindow expressions</td>
<td>11</td>
</tr>
<tr>
<td>Evaluating DataWindow expressions in code</td>
<td>11</td>
</tr>
<tr>
<td>Evaluating conditional DataWindow expressions with current data</td>
<td>12</td>
</tr>
</tbody>
</table>

Where you use DataWindow expressions

A DataWindow expression is a combination of data, operators, and functions that, when evaluated, results in a value. An expression can include column names, operators, DataWindow expression functions, and constants such as numbers and text strings.

In painters

DataWindow expressions are associated with DataWindow objects and reports. You specify them in the DataWindow painter. You can also specify expressions in the Database painter, although these expressions have a slightly different format and are used only in validation rules.

For information about DataWindow expression functions that you can use in expressions, see “Using DataWindow expression functions” on page 13, or look up the function you want in online Help.
In painters, you use expressions in these ways:

**Table 1-1: Using DataWindow expressions in painters**

<table>
<thead>
<tr>
<th>In this painter</th>
<th>Expressions are used in</th>
</tr>
</thead>
<tbody>
<tr>
<td>DataWindow painter</td>
<td>Computed fields</td>
</tr>
<tr>
<td></td>
<td>Conditional expressions for property values</td>
</tr>
<tr>
<td></td>
<td>Validation rules</td>
</tr>
<tr>
<td></td>
<td>Filters</td>
</tr>
<tr>
<td></td>
<td>Sorting</td>
</tr>
<tr>
<td></td>
<td>Series and values in graphs</td>
</tr>
<tr>
<td></td>
<td>Columns, rows, and values in crosstabs</td>
</tr>
<tr>
<td>Database painter</td>
<td>Validation rules</td>
</tr>
</tbody>
</table>

**Other types of expressions you use**

You also use expressions in Quick Select, SQL Select, and the Query painter to specify selection criteria, and in SQL Select and the Query painter to create computed columns. In these painters you are using SQL operators and DBMS-specific functions, not DataWindow expression operators and functions, to create expressions.

Some of the specific places where you use expressions are described here.

**In computed fields**

Expressions for computed fields can evaluate to any value. The datatype of the expression becomes the datatype of the computed field:

**Table 1-2: Using expressions in computed fields**

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Today ()</td>
<td>Displays the date using the Today function</td>
</tr>
<tr>
<td>Salary/12</td>
<td>Computes the monthly salary</td>
</tr>
<tr>
<td>Sum (Salary for group 1)</td>
<td>Computes the salary for the first group using the Sum aggregate function</td>
</tr>
<tr>
<td>Price*Quantity</td>
<td>Computes the total cost</td>
</tr>
</tbody>
</table>

**Expressions for graphs and crosstabs**

You can use similar expressions for series and values in graphs and for columns, rows, and values in crosstabs.
In filters

Filter expressions are boolean expressions that must evaluate to true or false:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Academics = &quot;*****&quot; AND Cost = &quot;$$&quot;</td>
<td>Displays data only for colleges with both a 5-star academic rating and a $$$ cost rating</td>
</tr>
<tr>
<td>Emp_sal &lt; 50000</td>
<td>Displays data for employees with salaries less than $50,000</td>
</tr>
<tr>
<td>Salary &gt; 50000 AND Dept_id BETWEEN 400 AND 700</td>
<td>Displays data for employees in departments 400, 500, 600, and 700 with salaries greater than $50,000</td>
</tr>
<tr>
<td>Month(Bdate) = 9 OR Month(Bdate) = 2</td>
<td>Displays data for people with birth dates in September or February</td>
</tr>
<tr>
<td>Match ( Lname, &quot;^[^ABC ]&quot; )</td>
<td>Displays data for people whose last name begins with A, B, or C</td>
</tr>
</tbody>
</table>

In validation rules for table columns

Validation rules are boolean expressions that compare column data with values and that use relational and logical operators. When the validation rule evaluates to false, the data in the column is rejected.

**In the DataWindow painter**  When you specify a validation rule in the DataWindow painter, you should validate the newly entered value. To refer to the newly entered value, use the GetText function. Because GetText returns a string, you also need a data conversion function (such as Integer or Real) if you compare the value to other types of data.

If you include the column name in the expression, you get the value that already exists for the column instead of the newly entered value that needs validating.

**In the Database painter**  When you specify the validation rule in the Database painter, you are defining a general rule that can be applied to any column. Use @placeholder to stand for the newly entered value. The name you use for @placeholder is irrelevant. You can assign the rule to any column that has a datatype appropriate for the comparison.

When you define a DataWindow object, a validation rule assigned to a column is brought into the DataWindow object and converted to DataWindow object syntax. @placeholder is converted to GetText and the appropriate datatype conversion function.
Operators used in DataWindow expressions

Other columns in the rule You can refer to values in other columns for the current row by specifying their names in the validation rule:

Table 1-4: Using expressions with values from other columns

<table>
<thead>
<tr>
<th>Expression in Database painter</th>
<th>Expression in DataWindow painter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>@column &gt;= 10000</td>
<td>Integer(GetText())&gt;= 10000</td>
<td>If a user enters a salary below $10,000, an error message displays.</td>
</tr>
<tr>
<td>@column IN (100, 200, 300)</td>
<td>Integer(GetText()) IN (100, 200, 300)</td>
<td>If a user does not enter a department ID of 100, 200, or 300, an error message displays.</td>
</tr>
<tr>
<td>@salary &gt; 0</td>
<td>Long(GetText()) &gt; 0</td>
<td>If a user does not enter a positive number, an error message displays.</td>
</tr>
<tr>
<td>Match(@disc_price, &quot;^[0-9]+$&quot;) and @disc_price &lt; Full_Price</td>
<td>Match(GetText( ), &quot;^[0-9]+$&quot;) and Real(GetText()) &lt; Full_Price</td>
<td>If a user enters any characters other than digits, or the resulting number is greater than or equal to the value in the Full_Price column, an error message displays.</td>
</tr>
</tbody>
</table>

Operators used in DataWindow expressions

An operator is a symbol or word in an expression that performs an arithmetic calculation or logical operation; compares numbers, text, or values; or manipulates text strings.

Four types of operators are available:

- **Arithmetic** for numeric datatypes. See “Arithmetic operators in DataWindow expressions” on page 5.
- **Relational** for all datatypes. See “Relational operators in DataWindow expressions” on page 5.
- **Logical** for all datatypes. See “Logical operators in DataWindow expressions” on page 9.
- **Concatenation** for string datatypes. See “Concatenation operator in DataWindow expressions” on page 10.
Arithmetic operators in DataWindow expressions

When you write an expression, you can use the following arithmetic operators:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>+</td>
<td>Addition</td>
<td>SubTotal + Tax</td>
</tr>
<tr>
<td>-</td>
<td>Subtraction</td>
<td>Price - Discount</td>
</tr>
<tr>
<td>*</td>
<td>Multiplication</td>
<td>Quantity * Price</td>
</tr>
<tr>
<td>/</td>
<td>Division</td>
<td>Discount / Price</td>
</tr>
<tr>
<td>^</td>
<td>Exponentiation</td>
<td>Rating ^ 2.5</td>
</tr>
</tbody>
</table>

Multiplication and division are carried out to full precision (16–18 digits). Values are rounded:

<table>
<thead>
<tr>
<th>Expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>20.0/3</td>
<td>6.666666666666667</td>
</tr>
<tr>
<td>3*(20.0/3)</td>
<td>20</td>
</tr>
<tr>
<td>Truncate(20.0/3,4)</td>
<td>6.6666</td>
</tr>
</tbody>
</table>

Calculations with null

When you form an arithmetic expression that contains a null value, the expression becomes null. Thinking of null as undefined makes this easier to understand. For example, when a null column is multiplied by 5, the entire expression also evaluates to null. Use the IsNull function to explicitly check for the null value.

Boolean expressions that contain a null value evaluate to false rather than to null. For more information, see “Relational operators in DataWindow expressions” next.

Relational operators in DataWindow expressions

You use relational operators to compare a value with other values. The result is a boolean expression whose value is always true or false.

Since the result of a boolean expression is always true or false, a relational operator that compares a value to null evaluates to false. For example, the expression “column > 5” evaluates to false (and “NOT column > 5” evaluates to true) when the column value is null.
When you write an expression, you can use the following relational operators (more information about LIKE, IN, and BETWEEN follows the table):

### Table 1-7: Using expressions with relational operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>=</td>
<td>Is equal to</td>
<td>Price = 100</td>
</tr>
<tr>
<td>&gt;</td>
<td>Is greater than</td>
<td>Price &gt; 100</td>
</tr>
<tr>
<td>&lt;</td>
<td>Is less than</td>
<td>Price &lt; 100</td>
</tr>
<tr>
<td>&lt;=</td>
<td>Is not equal to</td>
<td>Price &lt;= 100</td>
</tr>
<tr>
<td>&gt;=</td>
<td>Greater than or equal to</td>
<td>Price &gt;= 100</td>
</tr>
<tr>
<td>NOT =</td>
<td>Is not equal to</td>
<td>Price NOT= 100</td>
</tr>
<tr>
<td>LIKE</td>
<td>Matches this specified pattern.</td>
<td>Emp_lname LIKE 'C%' OR Emp_lname LIKE 'G%'</td>
</tr>
<tr>
<td>IN</td>
<td>Is in this set of values.</td>
<td>Dept_id IN (100, 200, 500)</td>
</tr>
<tr>
<td>BETWEEN</td>
<td>Is within this range of values. The range includes the first and last values.</td>
<td>Price BETWEEN 1000 AND 3000</td>
</tr>
<tr>
<td>NOT LIKE</td>
<td>Does not match this specified pattern.</td>
<td>Emp_lname NOT LIKE 'C%', AND Emp_lname NOT LIKE 'G%'</td>
</tr>
<tr>
<td>NOT IN</td>
<td>Is not in this set of values.</td>
<td>Dept_id NOT IN (100, 200, 500)</td>
</tr>
<tr>
<td>NOT BETWEEN</td>
<td>Is outside this range of values. The range includes the first and last values.</td>
<td>Price NOT BETWEEN 1000 AND 2000</td>
</tr>
</tbody>
</table>

### Special characters for operations with strings

You can use the following special characters with relational operators that take string values:

### Table 1-8: Special characters for use in expressions with relational operators

<table>
<thead>
<tr>
<th>Special character</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>% (percent)</td>
<td>Matches any group of characters.</td>
<td>Good% matches all names that begin with Good.</td>
</tr>
<tr>
<td>_ (underscore)</td>
<td>Matches any single character.</td>
<td>Good _ _ _ matches all 7-letter names that begin with Good.</td>
</tr>
</tbody>
</table>

### LIKE and NOT LIKE operators

Use LIKE to search for strings that match a predetermined pattern. Use NOT LIKE to search for strings that do not match a predetermined pattern. When you use LIKE or NOT LIKE, you can use the % or _ characters to match unknown characters in a pattern.
For example, the following expression for the Background.Color property of the Salary column displays salaries in red for employees with last names beginning with F and displays all other salaries in white:

\[
\text{If(emp\_lname LIKE 'F%', RGB(255,0,0), RGB(255,255,255))}
\]

**Escape keyword**

If you need to use the % or _ characters as part of the string, you can use the escape keyword to indicate that the character is part of the string. For example, the _ character in the following filter string is part of the string to be searched for, but is treated as a wildcard:

\[
\text{comment LIKE '~%o_a15progress%~'}
\]

The escape keyword designates any character as an escape character (do not use a character that is part of the string you want to match). In the following example, the asterisk (*) character is inserted before the _ character and designated as an escape character, so that the _ character is treated as part of the string to be matched:

\[
\text{comment like ~'\%o\*_a15progress\%~' escape ~'*~'}
\]

Use BETWEEN to check if a value is within a range of values. Use NOT BETWEEN to check if a value is not in a range of values. The range of values includes the boundary values that specify the range.

For example, the following expression for the Background.Color property of the Salary column displays salaries in red when an employee’s salary is between $50,000 and $100,000 and displays all other salaries in white:

\[
\text{If(salary BETWEEN 50000 AND 100000, RGB(255,0,0), RGB(255,255,255))}
\]

You can use the BETWEEN and NOT BETWEEN operators with string values. For example, if the following expression is used for the Visual property of a column, column values display only for departments listed alphabetically between Finance and Sales:

\[
\text{If(dept\_name BETWEEN 'Finance' AND 'Sales', 1, 0)}
\]

The % or _ characters can be used when you are using string values with the BETWEEN and NOT BETWEEN operators. This example might include more department listings than the previous example:

\[
\text{If(dept\_name BETWEEN 'F\%' AND 'S\%', 1, 0)}
\]

You can also use the BETWEEN and NOT BETWEEN operators with methods. For example:

\[
\text{GetRow( ) BETWEEN 5 AND 8}
\]
Operators used in DataWindow expressions

IN and NOT IN operators

Use IN to check if a value is in a set of values. Use NOT IN to check if a value is not in a set of values.

For example, the following expression for the Background.Color property of the Salary column displays salaries in red for employees in department 300 or 400 having a salary between $50,000 and $100,000, and displays all other salaries in white:

\[
\text{If}(\text{dept\_id} \text{ IN } (300,400) \text{ and salary BETWEEN } 50000 \text{ AND } 100000, \text{ RGB}(255,0,0), \text{ RGB}(255,255,255))
\]

Comparing strings in DataWindow expressions

When you compare strings, the comparison is case sensitive. Leading blanks are significant, but trailing blanks are not.

Case-sensitivity examples

Assume City1 is “Austin” and City2 is “AUSTIN”. Then:

\[
\text{City1}=\text{City2}
\]

returns false.

To compare strings regardless of case, use the \textit{Upper} or \textit{Lower} function. For example:

\[
\text{Upper(City1)}=\text{Upper(City2)}
\]

returns true.

For information about these functions, see “Using DataWindow expression functions” on page 13.

Blanks examples

Assume City1 is "Austin" and City2 is " Austin ". Then the expression:

\[
\text{City1}=\text{City2}
\]

returns false. DataWindow .NET removes the trailing blank before making the comparison, but it does not remove the leading blank.

To prevent leading blanks from affecting a comparison, remove them with one of the trim functions: \textit{Trim} or \textit{LeftTrim}.

For example:

\[
\text{Trim(City1)}=\text{Trim(City2)}
\]

returns true.
To compare strings when trailing blanks are significant, use an expression such as the following to ensure that any trailing blanks are included in the comparison:

\[
\text{City}_1 + ">" = \text{City}_2 + ">"
\]

For information about these functions, see “Using DataWindow expression functions” on page 13.

**Logical operators in DataWindow expressions**

You use logical operators to combine boolean expressions into a larger boolean expression. The result is always **true** or **false**:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>NOT</strong></td>
<td>Logical negation. If A is true, NOT A is false. If A is false, NOT A is true.</td>
<td>NOT Price = 100</td>
</tr>
<tr>
<td><strong>AND</strong></td>
<td>Logical and. A AND B is true if both are true. A AND B is false if either is false.</td>
<td>Tax &gt; 3 AND Ship &lt; 5</td>
</tr>
<tr>
<td><strong>OR</strong></td>
<td>Logical or. A OR B is true if either is true or both are true. A OR B is false only if both are false.</td>
<td>Tax &gt; 3 OR Ship &lt; 5</td>
</tr>
</tbody>
</table>

When you combine two or more boolean expressions to form a new expression, the new expression is either **true** or **false**. The following truth table shows how **true** and **false** expressions are evaluated to form an expression that is either **true** or **false**.

For example, if "My dog has fleas" is **true** and "My hair is brown" is **false**, then "My dog has fleas OR my hair is brown" is **true**, and "My dog has fleas AND my hair is brown" is **false**:

<table>
<thead>
<tr>
<th>If one expression has this value</th>
<th>And the logical operator is</th>
<th>And if another expression has this value</th>
<th>The resulting expression has this value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>TRUE</strong></td>
<td>AND</td>
<td><strong>TRUE</strong></td>
<td><strong>TRUE</strong></td>
</tr>
<tr>
<td><strong>TRUE</strong></td>
<td>AND</td>
<td><strong>FALSE</strong></td>
<td><strong>FALSE</strong></td>
</tr>
<tr>
<td><strong>FALSE</strong></td>
<td>AND</td>
<td><strong>TRUE</strong></td>
<td><strong>FALSE</strong></td>
</tr>
</tbody>
</table>
Operators used in DataWindow expressions

<table>
<thead>
<tr>
<th>If one expression has this value</th>
<th>And the logical operator is</th>
<th>And if another expression has this value</th>
<th>The resulting expression has this value</th>
</tr>
</thead>
<tbody>
<tr>
<td>FALSE</td>
<td>AND</td>
<td>FALSE</td>
<td>FALSE</td>
</tr>
<tr>
<td>TRUE</td>
<td>OR</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
<tr>
<td>TRUE</td>
<td>OR</td>
<td>FALSE</td>
<td>TRUE</td>
</tr>
<tr>
<td>FALSE</td>
<td>OR</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
<tr>
<td>FALSE</td>
<td>OR</td>
<td>FALSE</td>
<td>TRUE</td>
</tr>
<tr>
<td>NOT TRUE</td>
<td>AND</td>
<td>TRUE</td>
<td>FALSE</td>
</tr>
<tr>
<td>NOT TRUE</td>
<td>AND</td>
<td>TRUE</td>
<td>FALSE</td>
</tr>
<tr>
<td>NOT FALSE</td>
<td>AND</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
<tr>
<td>NOT TRUE</td>
<td>OR</td>
<td>FALSE</td>
<td>TRUE</td>
</tr>
<tr>
<td>NOT TRUE</td>
<td>OR</td>
<td>FALSE</td>
<td>TRUE</td>
</tr>
<tr>
<td>NOT FALSE</td>
<td>OR</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
<tr>
<td>NOT TRUE</td>
<td>OR</td>
<td>FALSE</td>
<td>TRUE</td>
</tr>
<tr>
<td>NOT FALSE</td>
<td>OR</td>
<td>TRUE</td>
<td>TRUE</td>
</tr>
</tbody>
</table>

If you use a logical operator with a boolean function that returns null, the term with the null return value is evaluated as false. If you use the NOT logical operator with a boolean function that returns null, the complete term evaluates to true. For example, NOT gf_boolean () evaluates to true when gf_boolean returns null.

Concatenation operator in DataWindow expressions

The concatenation operator joins the contents of two variables of the same type to form a longer value. You can concatenate strings and blobs.

To concatenate values, you use the plus sign (+) operator.

**Table 1-11: Using expressions with concatenation operator**

<table>
<thead>
<tr>
<th>String expression</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;over&quot; + &quot;stock&quot;</td>
<td>overstock</td>
</tr>
<tr>
<td>Lname + ', ' + Fname</td>
<td>If Lname is Hill and Fname is Craig, then &quot;Hill, Craig&quot;</td>
</tr>
</tbody>
</table>

Using quotes

You can use either single or double quotes in string expressions. For example, the expression “over” + “stock” is equivalent to the expression ‘over’ + ‘stock’.
Operator precedence in DataWindow expressions

To ensure predictable results, operators in DataWindow expressions are evaluated in a specific order of precedence. When operators have the same precedence, they are evaluated from left to right.

The following table lists the operators in descending order of precedence:

<table>
<thead>
<tr>
<th>Operator</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>(</td>
<td>Grouping</td>
</tr>
<tr>
<td>^</td>
<td>Exponentiation</td>
</tr>
<tr>
<td>*, /</td>
<td>Multiplication and division</td>
</tr>
<tr>
<td>+, -</td>
<td>Addition and subtraction; string concatenation</td>
</tr>
<tr>
<td>IN, LIKE, BETWEEN</td>
<td>SQL SELECT statement conditions</td>
</tr>
<tr>
<td>=, &gt;, &lt;, &lt;=, &gt;=, &lt;&gt;</td>
<td>Relational operators</td>
</tr>
<tr>
<td>AND, OR</td>
<td>Logical and and logical or</td>
</tr>
<tr>
<td>NOT</td>
<td>Logical negation</td>
</tr>
</tbody>
</table>

Overriding the precedence order

Since expressions in parentheses are evaluated first, to override the precedence order, enclose expressions in parentheses. You can also use parentheses to clarify the order of evaluation. Within each set of parentheses, precedence order applies.

In the expression \( x + y \times a + b \), \( y \) is first multiplied by \( a \) (because multiplication has a higher precedence than addition). The result of the multiplication is then added to \( x \) and this result is then added to \( b \) (because the + operators are evaluated left to right).

To force evaluation in a different order, group expressions with parentheses. For example, in the expression \( x + (y \times (a + b)) \), \( a + b \) is evaluated first. The sum \( a + b \) is then multiplied by \( y \), and this product is added to \( x \).

Evaluating DataWindow expressions in code

In code, you use methods, properties, and data expressions for the DataWindow control to get information about the state of the DataWindow: the current row, the highlighted row, values of particular items. You can get other information by accessing properties of the DataWindow object, either with the Describe function or with property expressions.
Evaluating conditional DataWindow expressions with current data

For example, if you need to find the current row in a DataWindow, use the DataWindow control property CurrentRow:

\[\text{rownum} = \text{dw1.CurrentRow}\]

If you need to find the first row on the current page in a DataWindow, there is no DataWindow control function to return this information, but you can find it in the appropriate DataWindow object property:

\[\text{strFirst} = \text{dw1.Describe("DataWindow.FirstRowOnPage")}\]
\[\text{strLast} = \text{dw1.Describe("DataWindow.LastRowOnPage")}\]
\[\text{dw1.Text} = "Rows " + \text{strFirst} + " to " + \text{strLast}\]

In some cases, however, information you need might not be available either by using DataWindow control functions or by accessing DataWindow object properties.

DataWindow expression functions sometimes provide information that is available in no other way. These functions, which are available within a DataWindow expression, are documented in “Using DataWindow expression functions” on page 13.

Evaluating conditional DataWindow expressions with current data

Querying a property for a column

Values for column properties normally apply to all the rows in the column. For example, if you set the Protect property to “1” for the Emp_Id column, the user will be unable to modify Emp_Id for any of the rows. If you query the property value for this column at runtime, it will return “1”.

When the column has a conditional expression

Instead of a constant, you can assign a conditional expression to some column properties. Such properties are set on a row-by-row basis at runtime.

For example, you might wish to allow users to enter an employee id for new rows but protect this value for existing rows. The conditional expression for this column’s Protect property would be:

\[\text{If(IsRowNew(), 0, 1)}\]

When you query the Protect property at runtime, the result in this case would be the actual expression (preceded by a default value and a tab character and enclosed in quotes) instead of the property value. The value for the Protect property would be:

"0 <tab> If(IsRowNew(), 0, 1)"
CHAPTER 2
DataWindow Expression Functions

About this chapter
This chapter provides syntax, descriptions, and examples of the functions you can use in expressions in the DataWindow painter.

Contents
<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Using DataWindow expression functions</td>
<td>13</td>
</tr>
<tr>
<td>Four examples</td>
<td>14</td>
</tr>
<tr>
<td>Alphabetical list of DataWindow expression functions</td>
<td>23</td>
</tr>
</tbody>
</table>

Using DataWindow expression functions

In the DataWindow painter, you can use DataWindow expression functions in expressions for computed fields, filters, validation rules, and graphed data, with some exceptions.

The dialog boxes in which you define expressions include a list box that lists the available functions and their arguments. The dialog boxes make it easy to insert a function into the expression.

For information about expressions, see Chapter 1, “DataWindow Operators and Expressions.”

Return values for functions and expressions

DataWindow expression functions can return the following datatypes:
- Double
- Decimal
- String
- DateTime
- Time

Within an expression, a function can return other datatypes (such as boolean, date, or integer), but the final value of an expression is converted to one of these datatypes.
An aggregate function is a function (such as Avg, Max, StDev, and Sum) that operates on a range of values in a column. When you use an aggregate function, some restrictions apply. You cannot use an aggregate function:

- In a filter
- In a validation rule
- As an argument for another aggregate function

When you use aggregate functions, they cancel the effect of setting Retrieve Rows As Needed. To do the aggregation, the DataWindow object always retrieves all rows.

No matter what country you are creating objects and developing an application in, you must use U.S. number notation in numbers or number masks in display formats, edit masks, and DataWindow expressions. This means that when you specify a number or number mask, use a comma as the thousands delimiter and period for the decimal place.

Numbers display appropriately in whatever countries you deploy applications in. At runtime, the locally correct symbols for numbers display (because the international Control Panel settings are used) when numbers are interpreted. For example, in countries where comma represents the decimal place and period represents thousands, users see numbers in those formats at runtime.

For information about the locally correct display of dates and day names, see String on page 129 and DayName on page 57.

### Example 1: counting null values in a column

A null value is a marker used to fill a place in a column where data is missing for any reason. The value might not be applicable, or it might be missing or unknown. When a database table is created, each column in the table either allows null values or does not allow them. The column or set of columns that define the primary key cannot allow null values. Sometimes it is useful to know how many null values there are in a particular column.
What you want to do

Suppose you are working with the Fin_code table in the Enterprise Application Sample Database. The Fin_code table has three columns:

<table>
<thead>
<tr>
<th>Column</th>
<th>What the column is</th>
<th>Allows null values?</th>
</tr>
</thead>
<tbody>
<tr>
<td>Code</td>
<td>Unique financial identifier (primary key)</td>
<td>No</td>
</tr>
<tr>
<td>Type</td>
<td>Code type: expense or revenue</td>
<td>No</td>
</tr>
<tr>
<td>Description</td>
<td>Code description: the department incurring the expense or getting the revenue</td>
<td>Yes</td>
</tr>
</tbody>
</table>

You create a DataWindow object using the Code and Description columns. You want to know the number of null values in the Description column.

How to do it

In the DataWindow object, you create a computed field that uses functions to display the number of null values in the Description column.

For the sake of demonstrating the use of functions, the following computed fields are created in the Summary band of the DataWindow object (with text objects that tell you what information each computed field is providing):

- \( \text{Count(description for all)} \)
  counts the number of descriptions (that are not null);  
  \( \text{Sum(If(IsNull(description), 1, 0))} \)
  returns a 1 if the description column is null, a 0 if the description column is not null, and then adds the total;  
- \( \text{Count(id for all)} \)
  counts the number of IDs (which is also the number of rows);  
  \( \text{Sum(If(IsNull(description), 1, 1))} \)
  adds the number of nulls and not nulls in the description column (which is the total number of rows) and should match the result of the \( \text{Count(id for all)} \) function; and  
  \( \text{IsNull(description)} \)
  evaluates whether the last row in the table has a description that is null. The return value of the \( \text{IsNull} \) function is true or false.
What you get

Here is the design for the DataWindow object.

<table>
<thead>
<tr>
<th>Id</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>AAAA</td>
</tr>
<tr>
<td>2</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>XXXXXXX</td>
</tr>
<tr>
<td>4</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of descriptions</th>
<th>Number of NULLs</th>
<th>Number of rows</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>3</td>
<td>8</td>
</tr>
</tbody>
</table>

Last value NULL? = true

Here is the DataWindow object showing eight descriptions, three of which are null and five of which are not null. The last description for Id=8 is null.

Example 2: counting male and female employees

Example 1 demonstrates the use of the Sum and Count functions. Sum and Count are two examples of a class of functions called aggregate functions.

An aggregate function is a function that operates on a range of values in a column. The aggregate functions are:

<table>
<thead>
<tr>
<th>Function</th>
<th>Aggregate Function</th>
</tr>
</thead>
<tbody>
<tr>
<td>Avg</td>
<td>Large, Mode, Sum</td>
</tr>
<tr>
<td>Count</td>
<td>Last, Percent, Var</td>
</tr>
<tr>
<td>CumulativePercent</td>
<td>Max, Small, VarP</td>
</tr>
<tr>
<td>CumulativeSum</td>
<td>Median, StDev</td>
</tr>
<tr>
<td>First</td>
<td>Min, StDevP</td>
</tr>
</tbody>
</table>

DataWindow .NET
About crosstab functions
Although the crosstab functions (CrosstabAvg, CrosstabAvgDec, CrosstabCount, CrosstabMax, CrosstabMaxDec, CrosstabMin, CrosstabMinDec, CrosstabSum, and CrosstabSumDec) behave like aggregate functions, they are not included on the list because they are for crosstabs only and are designed to work in the crosstab matrix.

A few restrictions apply to the use of aggregate functions. You cannot use an aggregate function:

- In a filter
- In a validation rule
- As an argument for another aggregate function

This example demonstrates the use of the Sum aggregate function.

What you want to do
Using the employee table in the EAS Demo DB as the data source, you create a DataWindow object using at least the Emp_id and the Sex columns. You want the DataWindow object to display the number of male employees and female employees in the company.

How to do it
In the summary band in the workspace, add two computed fields to the DataWindow object that use the Sum and If functions:

- \( \text{Sum(If(sex = "M", 1, 0))} \) counts the number of males in your company;
- \( \text{Sum(If(sex = "F", 1, 0))} \) counts the number of females in your company.

By clicking the Page computed field button, you can also add a Page computed field in the footer band to display the page number and total pages at the bottom of each page of the DataWindow object.

What you get
Here is what the design of the DataWindow object looks like.

<table>
<thead>
<tr>
<th>Employee ID</th>
<th>Sex</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header 1</td>
<td></td>
</tr>
<tr>
<td>Emp_id</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Male</td>
</tr>
<tr>
<td></td>
<td>Female</td>
</tr>
<tr>
<td>Detail 1</td>
<td></td>
</tr>
<tr>
<td>Number of males</td>
<td>Number of females</td>
</tr>
<tr>
<td>( \text{Sum(If(sex = &quot;M&quot;, 1, 0))} )</td>
<td>( \text{Sum(If(sex = &quot;F&quot;, 1, 0))} )</td>
</tr>
<tr>
<td>Summary 1</td>
<td></td>
</tr>
<tr>
<td>Page + page() + &quot;.&quot; + pageCount()</td>
<td></td>
</tr>
<tr>
<td>Footer 1</td>
<td></td>
</tr>
</tbody>
</table>
Here is the last page of the DataWindow object, with the total number of males and females in the company displayed.

<table>
<thead>
<tr>
<th></th>
<th>Male</th>
<th>Female</th>
</tr>
</thead>
<tbody>
<tr>
<td>1684</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1740</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1751</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Number of males** | **Number of females**
---|---
41 | 34

What if you decide that you also want to know the number of males and females in each department in the company?

❖ **To display the males and females in each department:**

1. Select Design>Data Source from the menu bar so that you can edit the data source.

2. Select Design>Select tables from the menu bar and open the Department table in the Select painter workspace, which currently displays the Employee table with the Emp_id and Sex columns selected.

3. Select the department_dept_name column to add it to your data source.

4. Select Rows>Create Group from the menu bar to create a group and group by department name.

5. In the trailer group band, add two additional computed fields:

   - \( \text{Sum(If(sex = "M", 1, 0) for group 1)} \) counts the number of males in each department;
   - \( \text{Sum(If(sex = "F", 1, 0) for group 1)} \) counts the number of females in each department.
Here is what the design of the grouped DataWindow object looks like.

<table>
<thead>
<tr>
<th>Employee ID</th>
<th>Sex</th>
</tr>
</thead>
<tbody>
<tr>
<td>department_dept_name</td>
<td></td>
</tr>
<tr>
<td>1: Header_group_department_dept_name:</td>
<td></td>
</tr>
<tr>
<td>emp_id</td>
<td>Male</td>
</tr>
<tr>
<td></td>
<td>Female</td>
</tr>
</tbody>
</table>

**Detail:**

<table>
<thead>
<tr>
<th>Number of males</th>
<th>Number of females</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sum([iif(emp_id=&quot;M&quot;,1,0) for group 1])</td>
<td>Sum([iif(emp_id=&quot;F&quot;,1,0) for group 1])</td>
</tr>
</tbody>
</table>

**Summary:**

<table>
<thead>
<tr>
<th>Page * page() + 'd' + pageCount()</th>
</tr>
</thead>
</table>

Here is the last page of the DataWindow object with the number of males and females in the shipping department displayed, followed by the total number of males and females in the company.

<table>
<thead>
<tr>
<th>Shipping</th>
</tr>
</thead>
<tbody>
<tr>
<td>181</td>
</tr>
<tr>
<td>703</td>
</tr>
<tr>
<td>759</td>
</tr>
<tr>
<td>568</td>
</tr>
<tr>
<td>921</td>
</tr>
<tr>
<td>1073</td>
</tr>
<tr>
<td>1579</td>
</tr>
<tr>
<td>1615</td>
</tr>
<tr>
<td>188</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of males</th>
<th>Number of females</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>11</td>
</tr>
</tbody>
</table>

**Example 3: creating a row indicator**

This example demonstrates the use of several functions: Bitmap, Case, CurrentRow, GetRow, and RGB.

**What you want to do**

Using the Employee table in the Enterprise Application Sample Database, you create a DataWindow object using the Emp_id, Emp_fname, Emp_lname, and Salary columns.
In the painter, you want to display a number of items such as the number of the current row, an arrow that is an indicator of the current row, and the salary for an employee with a background color that depends on what the salary is.

In the workspace, add the following:

- A computed field `CurrentRow()`, which displays the number of the current row.

- A picture object, which is a right-arrow, for which you define an expression for the arrow’s visible property:
  
  ```javascript
  If(CurrentRow() = GetRow(), 1, 0)
  ```

  The expression causes an arrow to display in the current row and no arrow to display in other rows.

- A computed field using the `If`, `CurrentRow`, and `GetRow` functions:
  
  ```javascript
  If(CurrentRow() = GetRow(), "Current", "Not current")
  ```

  displays the word “Current” when the row is the current row and “Not current” for all other rows.

- A computed field (typed on one line) using the `Bitmap`, `CurrentRow`, and `GetRow` functions:
  
  ```javascript
  Bitmap(If(CurrentRow() = GetRow(), "c:samp1\ex\code\indicatr.bmp", " " ))
  ```

  displays an arrow bitmap for the current row and no bitmap for all other rows.

- An expression for the Background.Color property of the salary column:
  
  ```javascript
  Case(salary WHEN IS >60000 THEN RGB(192,192,192)
  WHEN IS >40000 THEN RGB(0,255,0) ELSE
  RGB(255,255,255))
  ```

  The expression causes a salary above $40,000 to display in green, a salary above $60,000 to display in gray, and all other salaries to display in white.

Here is what the design of the DataWindow object looks like:

<table>
<thead>
<tr>
<th>Current Row</th>
<th>Employee ID</th>
<th>First Name</th>
<th>Last Name</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>CurrentRow()</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

```javascript
If(currentRow() = getrow(), "Current", "Not current")
Bitmap(If(CurrentRow() = GetRow(), "c:samp1\ex\code\indicatr.bmp", " " ))
```
Here is what the data looks like with the second row current.

<table>
<thead>
<tr>
<th>Current Row</th>
<th>Employee ID</th>
<th>First Name</th>
<th>Last Name</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>103</td>
<td>Fran</td>
<td>Whitney</td>
<td>$45,700.00</td>
</tr>
<tr>
<td>Not current</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Current</td>
<td>105</td>
<td>Matthew</td>
<td>Cubb</td>
<td>$62,000.00</td>
</tr>
<tr>
<td></td>
<td>128</td>
<td>Phillip</td>
<td>Chin</td>
<td>$38,500.00</td>
</tr>
</tbody>
</table>

Notice that the number of the current row is 2; the first row and the third row are "Not current" (and therefore display no bitmap); and the second row, which is the current row, displays the arrow row indicator.

On your screen, the salary in the first row has a green background because it is more than $40,000; the salary in the second row has a gray background because it is more than $60,000; and the salary in the third row has a white background, which matches the background of the DataWindow object.

### Example 4: displaying all data when a column allows nulls

When you create an arithmetic expression that has a null value, the value of the expression is null. This makes sense, since null means essentially undefined and the expression is undefined, but sometimes this fact can interfere with what you want to display.

**What you want to do**

A table in your database has four columns: Id, Corporation, Address1, and Address2. The Corporation, Address1, and Address2 columns allow null values. Using this table as the data source, you create a DataWindow object using the four columns. You now want the DataWindow object to display both parts of the address, separated by a comma.

You create a computed field to concatenate Address1 and Address2 with a comma separator. Here is the expression that defines the computed field:

```
address1 + ", " + address2
```
When you preview the DataWindow object, if either Address1 or Address2 is null, no part of the address displays because the value of the expression is null. To display a part of the address, you need to create a computed field that forces evaluation even if Address2 is null. Note that Address2 is assumed to have data only if Address1 has data for a particular row.

How to do it

In the detail band, create a computed field that uses the If and IsNull functions:

\[
\text{If}(\text{IsNull}(\text{address1} + \text{address2}), \text{address1}, \text{address1} + "," + \text{address2})
\]

The computed field says this: if the concatenation of the addresses is null (because address2 is null), then display address1, and if it is not null, display both parts of the address separated by a comma.

What you get

Here is what the design of the DataWindow object looks like. It includes both the computed field that does not work and the one that does.

When you preview the DataWindow object, notice that the first computed field displays null for ABC Corporation and XYZ Corporation. The second computed field displays the first part of the address, which is not null.
Alphabetical list of DataWindow expression functions

The list of DataWindow expression functions follows in alphabetical order.
Abs

Description: Calculates the absolute value of a number.

Syntax: `Abs (n)`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>n</code></td>
<td>The number for which you want the absolute value</td>
</tr>
</tbody>
</table>

Return value: The datatype of `n`. Returns the absolute value of `n`.

Examples: This expression counts all the product numbers where the absolute value of the product number is distinct:

```
Count(product_number for All DISTINCT Abs(product_number))
```

Only data with an absolute value greater than 5 passes this validation rule:

```
Abs(value_set) > 5
```

See also: Count

ACos

Description: Calculates the arc cosine of an angle.

Syntax: `ACos (n)`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>n</code></td>
<td>The ratio of the lengths of two sides of a triangle for which you want a corresponding angle (in radians). The ratio must be a value between -1 and 1.</td>
</tr>
</tbody>
</table>

Return value: Double. Returns the arc cosine of `n` if it succeeds.

Examples: This expression returns 0:

```
ACos (1)
```

This expression returns 3.141593 (rounded to six places):

```
ACos (-1)
```

This expression returns 1.000000 (rounded to six places):

```
ACos (.540302)
```

See also: Cos, ASin, ATan
CHAPTER 2  DataWindow Expression Functions

Asc

Description
Converts the first character of a string to its Unicode code point. A Unicode code point is the numerical integer value given to a Unicode character.

Syntax
\[ \text{Asc (string)} \]

Return value
Unsigned integer. Returns the code point value of the first character in \text{string}.

Usage
Use Asc to test the case of a character or manipulate text and letters.
To find out the case of a character, you can check whether its code point value is within the appropriate range.

Examples
This expression for a computed field returns the string in code_id if the code point value of the first character in code_id is A (65):

\[
\text{If (Asc(code_id) = 65, code_id, "Not a valid code")}
\]

This expression for a computed field checks the case of the first character of lname and if it is lowercase, makes it uppercase:

\[
\text{IF (Asc(lname) > 64 AND Asc(lname) < 91, lname, WordCap(lname))}
\]

See also
Char
WordCap

AscA

Description
Converts the first character of a string to its ASCII integer value.

Syntax
\[ \text{AscA (string)} \]

Return value
Integer. Returns the ASCII value of the first character in \text{string}.

Usage
Use AscA to test the case of a character or manipulate text and letters.
To find out the case of a character, you can check whether its ASCII value is within the appropriate range.
**ASin**

**Description**
Calculates the arc sine of an angle.

**Syntax**

\[ \text{ASin}(n) \]

**Argument | Description**
--- | ---
\( n \) | The ratio of the lengths of two sides of a triangle for which you want a corresponding angle (in radians). The ratio must be a value between -1 and 1.

**Return value**
Double. Returns the arc sine of \( n \) if it succeeds.

**Examples**
This expression returns .999998 (rounded to six places):

\[ \text{ASin}(0.84147) \]

This expression returns .520311 (rounded to six places):

\[ \text{ASin} (\text{LogTen} (\text{Pi} (1))) \]

This expression returns 0:

\[ \text{ASin} (0) \]

**See also**
Sin
ACos
ATan
Pi
CHAPTER 2 DataWindow Expression Functions

ATan

Description
Calculates the arc tangent of an angle.

Syntax
ATan ( n )

Argument | Description
---|---
 n | The ratio of the lengths of two sides of a triangle for which you want a corresponding angle (in radians)

Return value
Double. Returns the arc tangent of n if it succeeds.

Examples
This expression returns 0:
ATan (0)

This expression returns 1.000 (rounded to three places):
ATan (1.55741)

This expression returns 1.267267 (rounded to six places):
ATan (Pi (1))

See also
Tan
ASin
ACos

Avg

Description
Calculates the average of the values of the column.

Syntax
Avg ( column ( FOR range { DISTINCT { expres1 {, expres2 {, ... } } } } ) )

Argument | Description
---|---
column | The column for which you want the average of the data values. Column can be the column name or the column number preceded by a pound sign (#). Column can also be an expression that includes a reference to the column. The datatype of column must be numeric.
Avg

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
</table>
| FOR range (optional) | The data that will be included in the average. For most presentation styles, values for range are:  
|                 |  
|                 | • ALL – (Default) The average of all values in column.  
|                 | • GROUP n – The average of values in column in the specified group. Specify the keyword GROUP followed by the group number: for example, GROUP 1.  
|                 | • PAGE – The average of the values in column on a page.  
|                 | For Crosstabs, specify CROSSTAB for range:  
|                 | • CROSSTAB – (Crosstabs only) The average of all values in column in the crosstab.  
|                 | For Graph and OLE objects, specify one of the following:  
|                 | • GRAPH – (Graphs only) The average of values in column in the range specified for the Rows option.  
|                 | • OBJECT – (OLE objects only) The average of values in column in the range specified for the Rows option.  
| DISTINCT (optional) | Causes Avg to consider only the distinct values in column when calculating the average. For a value of column, the first row found with the value is used and other rows that have the same value are ignored.  
| expresn (optional) | One or more expressions that you want to evaluate to determine distinct rows. Expresn can be the name of a column, a function, or an expression.  

Return value

The numeric datatype of the column. Returns the average of the values of the rows in range.

Usage

If you specify range, Avg returns the average value of column in range. If you specify DISTINCT, Avg returns the average value of the distinct values in column, or if you specify expresn, the average of column for each distinct value of expresn.

For graphs and OLE objects, you do not select the range when you call the function. The range has already been determined by the Rows setting on the Data property page (the Range property), and the aggregation function uses that range. Settings for Rows include the following:

• For the Graph presentation style, Rows is always All.
• For Graph controls, Rows can be All, Page, or Group.
• For OLE controls, Rows can be All, Current Row, Page, or Group. The available choices depend on the layer the control occupies.

In calculating the average, null values are ignored.
Not in validation rules or filter expressions
You cannot use this or other aggregate functions in validation rules or filter expressions.

Using an aggregate function cancels the effect of setting Retrieve Rows As Needed in the painter. To do the aggregation, a DataWindow object always retrieves all rows.

Examples

This expression returns the average of the values in the column named salary:

\[ \text{Avg}(\text{salary}) \]

This expression returns the average of the values in group 1 in the column named salary:

\[ \text{Avg}(\text{salary for group 1}) \]

This expression returns the average of the values in column 5 on the current page:

\[ \text{Avg}(\#5 \text{ for page}) \]

This computed field returns Above Average if the average salary for the page is greater than the average salary:

\[ \text{If}(\text{Avg}(\text{salary for page}) > \text{Avg}(\text{salary}), \text{"Above Average"}, \text{""}) \]

This expression for a graph value sets the data to the average value of the sale_price column:

\[ \text{Avg}(\text{sale_price}) \]

This expression for a graph value sets the data value to the average value of the sale_price column for the entire graph:

\[ \text{Avg}(\text{sale_price for graph}) \]

Assuming a DataWindow object displays the order number, amount, and line items for each order, this computed field returns the average of the order amount for the distinct order numbers:

\[ \text{Avg}(\text{order_amt for all DISTINCT order_nbr}) \]

See also

Median
Mode
**Bitmap**

**Description**
Displays the specified bitmap.

---

**For computed fields only**
You can use the **Bitmap** function *only* in a computed field.

---

**Syntax**

```
Bitmap ( string )
```

**Argument** | **Description**
--- | ---
`string` | A column containing bitmap files, a string containing the name of an image file (a BMP, GIF, JPEG, RLE, or WMF file), or an expression that evaluates to a string containing the name of an image file

---

**Return value**
The special datatype bitmap, which *cannot* be used in any other function.

**Usage**
Use **Bitmap** to dynamically display a bitmap in a computed field. When `string` is a column containing bitmap files, a different bitmap can display for each row.

**Examples**
These examples are all expressions for a computed field.

This expression dynamically displays the bitmap file contained in the column named `employees`:

```
Bitmap ( employees )
```

If the `employees` column is column 3, this next expression gives the same result as the expression above:

```
Bitmap ( #3 )
```

This expression displays the bitmap `tools.bmp`:

```
Bitmap ( "TOOLS.BMP" )
```

This expression tests the value in the column named `password` and then uses the value to determine which bitmap to display:

```
Bitmap ( If( password = "y", "yes.bmp", "no.bmp" ) )
```

**See also**
“Example 3: creating a row indicator” on page 19
CHAPTER 2   DataWindow Expression Functions

Case

Description
Tests the values of a column or expression and returns values based on the results of the test.

Syntax
\[
\text{Case}(\text{column WHEN value1 THEN result1 WHEN value2 THEN result2 ... ELSE resultelse})
\]

Argument | Description
--- | ---
\text{column} | The column or expression whose values you want to test. \text{Column} can be the column name or the column number preceded by a pound sign (#). \text{Column} can also be an expression that includes a reference to the column. \text{Column} is compared to each \text{valuen}.
\text{WHEN} (optional) | Introduces a value-result pair. At least one \text{WHEN} is required.
\text{valuen} | One or more values that you want to compare to values of \text{column}. A value can be:
- A single value
- A list of values separated by commas (for example, 2, 4, 6, 8)
- A TO clause (for example, 1 TO 20)
- IS followed by a relational operator and comparison value (for example, IS>5)
- Any combination of the above with an implied OR between expressions (for example, 1,3,5,7,9,27 TO 33, IS>42)
\text{THEN} | Introduces the result to be returned when \text{column} matches the corresponding \text{valuen}.
\text{resultn} | An expression whose value is returned by Case for the corresponding \text{valuen}. All \text{resultn} values must have the same datatype.
\text{ELSE} (optional) | Specifies that for any values of \text{column} that do not match the values of \text{valuen} already specified, Case returns \text{resultelse}.
\text{resultelse} | An expression whose value is returned by Case when the value of \text{column} does not match any \text{WHEN} \text{valuen} expression.

Return value
The datatype of \text{resultn}. Returns the result you specify in \text{resultn}.

Usage
If more than one \text{WHEN} clause matches \text{column}, Case returns the result of the first matching one.

Examples
This expression for the Background.Color property of a Salary column returns values that represent red when an employee’s salary is greater than $70,000, green when an employee’s salary is greater than $50,000, and blue otherwise:

\[
\text{Case(salary WHEN IS >70000 THEN RGB(255,0,0) WHEN IS >50000 THEN RGB(0,255,0) ELSE RGB(0,0,255))}
\]
This expression for the Background.Color property of an employee Id column returns red for Id 101, gray for Id 102, and black for all other Id numbers:

\[
\text{Case}(\text{emp}\_\text{id} \text{ WHEN } 101 \text{ THEN } 255 \text{ WHEN } 102 \text{ THEN } \text{RGB}(100,100,100) \text{ ELSE } 0)
\]

This expression for the Format property of the Marital_status column returns Single, Married, and Unknown based on the data value of the Marital_status column for an employee:

\[
\text{Case}(\text{marital}\_\text{status} \text{ WHEN } 'S' \text{ THEN } 'Single' \text{ WHEN } 'M' \text{ THEN } 'Married' \text{ ELSE } 'Unknown')
\]

See also “Example 3: creating a row indicator” on page 19

If

### Ceiling

**Description**

Retrieves the smallest whole number that is greater than or equal to a specified limit.

**Syntax**

\[
\text{Ceiling} (n)
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n )</td>
<td>The number for which you want the smallest whole number that is greater than or equal to it</td>
</tr>
</tbody>
</table>

**Return value**

The datatype of \( n \). Returns the smallest whole number that is greater than or equal to \( n \).

**Examples**

These expressions both return -4:

\[
\text{Ceiling}(-4.2)
\]

\[
\text{Ceiling}(-4.8)
\]

This expression for a computed field returns ERROR if the value in discount_amt is greater than the smallest whole number that is greater than or equal to discount_factor times price. Otherwise, it returns discount_amt:

\[
\text{If}(\text{discount}\_\text{amt} \leq \text{Ceiling}(\text{discount}\_\text{factor} * \text{price}), \text{String}(\text{discount}\_\text{amt}), \text{"ERROR"})
\]
To pass this validation rule, the value in discount_amt must be less than or equal to the smallest whole number that is greater than or equal to discount_factor times price:

\[
\text{discount\_amt} \leq \text{Ceiling}(\text{discount\_factor} \times \text{price})
\]

**See also**
- Int
- Round
- Truncate

### Char

**Description**
Converts an integer to a Unicode character.

**Syntax**

```
Char (n)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>The integer you want to convert to a character</td>
</tr>
</tbody>
</table>

**Return value**
String. Returns the character whose code point value is \( n \).

**Examples**
This expression returns the escape character:

```
Char(27)
```

**See also**
- Asc

### CharA

**Description**
Converts an integer to an ASCII character.

**Syntax**

```
CharA (n)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>The integer you want to convert to a character</td>
</tr>
</tbody>
</table>

**Return value**
String. Returns the character whose ASCII value is \( n \).

**Examples**
This expression returns the escape character:

```
CharA(27)
```

**See also**
- AscA
**Cos**

Description  Calculates the cosine of an angle.

Syntax  \texttt{Cos ( n )}

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textit{n}</td>
<td>The angle (in radians) for which you want the cosine</td>
</tr>
</tbody>
</table>

Return value  Double. Returns the cosine of \textit{n}.

Examples  This expression returns 1:

\texttt{Cos (0)}

This expression returns .540302:

\texttt{Cos (1)}

This expression returns -1:

\texttt{Cos (Pi(1))}

See also  Pi

Sin

Tan

---

**Count**

Description  Calculates the total number of rows in the specified column.

Syntax  \texttt{Count ( column \{ FOR range \{ DISTINCT \{ expres1 \{, expres2 \{, ... \}\}\}\}\}\})}

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\textit{column}</td>
<td>The column for which you want the number of rows. \textit{Column} can be the column name or the column number preceded by a pound sign (#). \textit{Column} can also be an expression that includes a reference to the column.</td>
</tr>
</tbody>
</table>
Usage

If you specify range, Count determines the number of rows in column in range. If you specify DISTINCT, Count returns the number of the distinct rows displayed in column, or if you specify expresn, the number of rows displayed in column where the value of expresn is distinct.

For graphs and OLE objects, you do not select the range when you call the function. The range has already been determined by the Rows setting on the Data property page (the Range property), and the aggregation function uses that range.

Settings for Rows include the following:

- For the Graph presentation style, Rows is always All.
- For Graph controls, Rows can be All, Page, or Group.
- For OLE controls, Rows can be All, Current Row, Page, or Group. The available choices depend on the layer the control occupies.

Null values in the column are ignored and are not included in the count.
CrosstabAvg

Not in validation rules or filter expressions
You cannot use this or other aggregate functions in validation rules or filter expressions.

Using an aggregate function cancels the effect of setting Retrieve Rows As Needed in the painter. To do the aggregation, a DataWindow object always retrieves all rows.

Examples
This expression returns the number of rows in the column named emp_id that are not null:
\[
\text{Count}(\text{emp_id})
\]
This expression returns the number of rows in the column named emp_id of group 1 that are not null:
\[
\text{Count}(\text{emp_id for group 1})
\]
This expression returns the number of dept_ids that are distinct:
\[
\text{Count}(\text{dept_id for all DISTINCT})
\]
This expression returns the number of regions with distinct products:
\[
\text{Count}(\text{region_id for all DISTINCT Lower(product_id)})
\]
This expression returns the number of rows in column 3 on the page that are not null:
\[
\text{Count}(\#3\text{ for page})
\]

See also
“Example 1: counting null values in a column” on page 14

CrosstabAvg

Description
Calculates the average of the values returned by an expression in the values list of the crosstab. When the crosstab definition has more than one column, CrosstabAvg can also calculate averages of the expression’s values for groups of column values.

For crosstabs only
You can use this function only in a crosstab DataWindow object.
CHAPTER 2  DataWindow Expression Functions

Syntax

\[
\text{CrosstabAvg} \left( n, \{ \text{column}, \text{groupvalue} \} \right)
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(n)</td>
<td>The number of the crosstab-values expression for which you want the average of the returned values. The crosstab expression must be numeric.</td>
</tr>
<tr>
<td>(\text{column}) (optional)</td>
<td>The number of the crosstab column as it is listed in the Columns box of the Crosstab Definition dialog box for which you want intermediate calculations.</td>
</tr>
<tr>
<td>(\text{groupvalue}) (optional)</td>
<td>A string whose value controls the grouping for the calculation. (\text{Groupvalue}) is usually a value from another column in the crosstab. To specify the current column value in a dynamic crosstab, rather than a specific value, specify @ plus the column name as a quoted string.</td>
</tr>
</tbody>
</table>

Return value

Double. Returns the average of the crosstab values returned by expression \(n\) for all the column values or, optionally, for a subset of column values. To return a decimal datatype, use CrosstabAvgDec.

Usage

This function is meaningful only for the average of the values of the expression in a row in the crosstab. This means you can use it only in the detail band, not in a header, trailer, or summary band.

Null values are ignored and are not included in the average.

**How functions in a crosstab are used** When a crosstab is generated from your definition, the appropriate computed fields are automatically created using the Crosstab functions. To understand the functions, consider a crosstab with two columns (year and quarter), a row (product), and the values expression \(\text{Avg(amount for crosstab)}\).

The Crosstab Definition dialog box looks like this.

---

DataWindow Object Reference 37
When you define the crosstab described above, the painter automatically creates the appropriate computed fields. A computed field named `avg_amount` returns the average of the quarterly figures for each year. Its expression is:

\[
\text{CrosstabAvg}(1, 2, \"@\text{year}\")
\]

A second computed field named `grand_avg_amount` computes the average of all the amounts in the row. Its expression is:

\[
\text{CrosstabAvg}(1)
\]

Other computed fields in the summary band use the `Avg` function to display the average of the values in the amount column, the yearly averages, and the final average.

The crosstab in the Design view looks like this.

<table>
<thead>
<tr>
<th>Year</th>
<th>Quarter</th>
<th>1998 Avg</th>
<th>1999 Avg</th>
</tr>
</thead>
<tbody>
<tr>
<td>1997</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1998</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>1999</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Each row in the crosstab (after adjusting the column widths) has cells for the amounts in the quarters, a repeating cell for the yearly average, and a grand average. The crosstab also displays averages of the amounts for all the financial codes in the quarters in the summary band at the bottom.
What the function arguments mean When the crosstab definition has more than one column, you can specify column qualifiers for any of the Crosstab functions, so that the crosstab displays calculations for groups of column values. As illustrated previously, when year and quarter are the columns in the crosstab, the expression for the computed field is:

\[
\text{CrosstabAvg}(1, 2, \text{"@year"})
\]

The value 2 refers to the quarter column (the second column in the Crosstab Definition dialog) and "@year" specifies grouping values from the year column (meaning the function will average values for the quarters within each year). The value 1 refers to the crosstab-values expression that will be averaged. In the resulting crosstab, the computed field repeats in each row after the cells for the quarters within each year.

Tips for defining crosstabs When you define a crosstab with more than one column, the order of the columns in the Columns box of the Crosstab Definition dialog box governs the way the columns are grouped. To end up with the most effective expressions, make the column that contains the grouping values (for example, year or department) the first column in the Columns box and the column that contains the values to be grouped (for example, quarter or employee) second.

To display calculations for groups of rows, define groups as you would for other DataWindow presentation styles and define computed fields in the group header or footer using noncrosstab aggregation functions, such as Avg, Sum, or Max.

Examples

The first two examples use the crosstab expressions shown below:

\[
\text{Count(emp_id for crosstab), Sum(salary for crosstab)}
\]

This expression for a computed field in the crosstab returns the average of the employee counts (the first expression):

\[
\text{CrosstabAvg}(1)
\]

This expression for a computed field in the crosstab returns the average of the salary totals (the second expression):

\[
\text{CrosstabAvg}(2)
\]
Consider a crosstab that has two columns (region and city) and the values expression `Avg(sales for crosstab)`. This expression for a computed field in the detail band computes the average sales over all the cities in a region:

`CrosstabAvg(1, 2, "@region")`

This expression for another computed field in the same crosstab computes the grand average over all the cities:

`CrosstabAvg(1)`

See also
- CrosstabAvgDec
- CrosstabCount
- CrosstabMax
- CrosstabMin
- CrosstabSum

### CrosstabAvgDec

**Description**
Calculates the average of the values returned by an expression in the values list of the crosstab and returns a result with the decimal datatype. When the crosstab definition has more than one column, `CrosstabAvgDec` can also calculate averages of the expression’s values for groups of column values.

**For crosstabs only**
You can use this function *only* in a crosstab DataWindow object.

**Syntax**
`CrosstabAvgDec( n {, column, groupvalue } )`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>n</code></td>
<td>The number of the crosstab-values expression for which you want the average of the returned values. The crosstab expression must be numeric.</td>
</tr>
<tr>
<td><code>column</code> (optional)</td>
<td>The number of the crosstab column as it is listed in the Columns box of the Crosstab Definition dialog box for which you want intermediate calculations.</td>
</tr>
<tr>
<td><code>groupvalue</code> (optional)</td>
<td>A string whose value controls the grouping for the calculation. <code>Groupvalue</code> is usually a value from another column in the crosstab. To specify the current column value in a dynamic crosstab, rather than a specific value, specify @ plus the column name as a quoted string.</td>
</tr>
</tbody>
</table>
Return value

Decimal. Returns the average of the crosstab values returned by expression \( n \) for all the column values or, optionally, for a subset of column values.

Usage

Use this function instead of \texttt{CrosstabAvg} when you want to return a decimal datatype instead of a double datatype. For more information, see \texttt{CrosstabAvg}.

See also

\texttt{CrosstabMaxDec}
\texttt{CrosstabMinDec}
\texttt{CrosstabSumDec}

**CrosstabCount**

Description

Counts the number of values returned by an expression in the values list of the crosstab. When the crosstab definition has more than one column, \texttt{CrosstabCount} can also count the number of the expression’s values for groups of column values.

*For crosstabs only*  
You can use this function *only* in a crosstab DataWindow object.

Syntax

\[
\texttt{CrosstabCount ( n (, column, groupvalue ) )}
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n )</td>
<td>The number of the crosstab-values expression for which you want the total number of returned values.</td>
</tr>
<tr>
<td>\texttt{column} (optional)</td>
<td>The number of the crosstab column as it is listed in the Columns box of the Crosstab Definition dialog for which you want intermediate calculations.</td>
</tr>
<tr>
<td>\texttt{groupvalue} (optional)</td>
<td>A string whose value controls the grouping for the calculation. \texttt{Groupvalue} is usually a value from another column in the crosstab. To specify the current column value in a dynamic crosstab, rather than a specific value, specify @ plus the column name as a quoted string.</td>
</tr>
</tbody>
</table>

Return value

Long. Returns the number of values returned by expression \( n \) for all the column values or, optionally, for a subset of column values.

Usage

This function is meaningful *only* for the count of the values of the expression in a *row* in the crosstab. This means you can use it only in the detail band, not in a header, trailer, or summary band.

Null values are ignored and are not included in the count.
CrosstabCount

For more information about restricting the calculation to groups of values when the crosstab definition has more than one column, see Usage for CrosstabAvg.

Reviewing the expressions
To review the expressions defined for the crosstab values, open the Crosstab Definition dialog box (select Design>Crosstab from the menubar).

Examples
These examples all use the crosstab-values expressions shown below:

\[
\text{Count(emp\_id for crosstab), Sum(salary for crosstab)}
\]

This expression for a computed field in the crosstab returns the count of the employee counts (the first expression):

\[
\text{CrosstabCount}(1)
\]

This expression for a computed field in the crosstab returns the count of the salary totals (the second expression):

\[
\text{CrosstabCount}(2)
\]

The next two examples use a crosstab with two columns (year and quarter), a row (product), and the values expression Avg(sales for crosstab).

This expression for a computed field returns the count of the sales for each year:

\[
\text{CrosstabCount}(1, 2, "@year")
\]

This expression for a computed field returns the count of all the sales in the row:

\[
\text{CrosstabCount}(1)
\]

For an example illustrating how the painter automatically defines a crosstab by creating computed fields using the Crosstab functions, see CrosstabAvg.

See also
CrosstabAvg
CrosstabMax
CrosstabMin
CrosstabSum
CHAPTER 2  DataWindow Expression Functions

CrosstabMax

**Description**
Calculates the maximum value returned by an expression in the values list of the crosstab. When the crosstab definition has more than one column, CrosstabMax can also calculate the maximum of the expression’s values for groups of column values.

**For crosstabs only**
You can use this function only in a crosstab DataWindow object.

**Syntax**
```
CrosstabMax ( n {, column, groupvalue } )
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>The number of the crosstab-values expression for which you want the maximum returned value. The expression’s datatype must be numeric.</td>
</tr>
<tr>
<td>column</td>
<td>The number of the crosstab column as it is listed in the Columns box of the Crosstab Definition dialog box for which you want intermediate calculations.</td>
</tr>
<tr>
<td>groupvalue</td>
<td>A string whose value controls the grouping for the calculation. Groupvalue is usually a value from another column in the crosstab. To specify the current column value in a dynamic crosstab, rather than a specific value, specify @ plus the column name as a quoted string.</td>
</tr>
</tbody>
</table>

**Return value**
Double. Returns the maximum value returned by expression n for all the column values or, optionally, for a subset of column values. To return a decimal datatype, use CrosstabMaxDec.

**Usage**
This function is meaningful only for the maximum of the values of the expression in a row in the crosstab. This means you can use it only in the detail band, not in a header, trailer, or summary band.

Null values are ignored and are not included in the comparison.

For more information about restricting the calculation to groups of values when the crosstab definition has more than one column, see Usage for CrosstabAvg.

**Reviewing the expressions**
To review the expressions defined for the crosstab values, open the Crosstab Definition dialog box (select Design>Crosstab from the menubar).
These examples all use the crosstab-values expressions shown below:

\[
\text{Count}(\text{emp_id for crosstab}), \text{Sum}(\text{salary for crosstab})
\]

This expression for a computed field in the crosstab returns the maximum of the employee counts (the first expression):

\[
\text{CrosstabMax}(1)
\]

This expression for a computed field in the crosstab returns the maximum of the salary totals (the second expression):

\[
\text{CrosstabMax}(2)
\]

The next two examples use a crosstab with two columns (year and quarter), a row (product), and a values expression \(\text{Avg}(\text{sales for crosstab})\).

This expression for a computed field returns the largest of the quarterly average sales for each year:

\[
\text{CrosstabMax}(1, 2, "@year")
\]

This expression for a computed field returns the maximum of all the average sales in the row:

\[
\text{CrosstabMax}(1)
\]

For an example illustrating how the painter automatically defines a crosstab by creating computed fields using the \text{Crosstab} functions, see \text{CrosstabAvg}.

See also

\text{CrosstabAvg}
\text{CrosstabCount}
\text{CrosstabMaxDec}
\text{CrosstabMin}
\text{CrosstabSum}

### CrosstabMaxDec

**Description**

Calculates the maximum value returned by an expression in the values list of the crosstab and returns a result with the decimal datatype. When the crosstab definition has more than one column, \text{CrosstabMaxDec} can also calculate the maximum of the expression’s values for groups of column values.

**For crosstabs only**

You can use this function only in a crosstab DataWindow object.
CHAPTER 2  DataWindow Expression Functions

Syntax

**CrosstabMaxDec** ( \( n \{, \) column, groupvalue \} )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n )</td>
<td>The number of the crosstab-values expression for which you want</td>
</tr>
<tr>
<td></td>
<td>the maximum returned value. The expression’s datatype must be</td>
</tr>
<tr>
<td></td>
<td>numeric.</td>
</tr>
<tr>
<td>column</td>
<td>The number of the crosstab column as it is listed in the Columns box of</td>
</tr>
<tr>
<td>(optional)</td>
<td>the Crosstab Definition dialog box for which you want</td>
</tr>
<tr>
<td></td>
<td>intermediate calculations.</td>
</tr>
<tr>
<td>groupvalue</td>
<td>A string whose value controls the grouping for the calculation. Groupvalue</td>
</tr>
<tr>
<td>(optional)</td>
<td>is usually a value from another column in the crosstab. To specify the</td>
</tr>
<tr>
<td></td>
<td>current column value in a dynamic crosstab, rather</td>
</tr>
<tr>
<td></td>
<td>than a specific value, specify @ plus the column name as a quoted string.</td>
</tr>
</tbody>
</table>

Return value

Decimal. Returns the maximum value returned by expression \( n \) for all the column values or, optionally, for a subset of column values.

Usage

Use this function instead of **CrosstabMax** when you want to return a decimal datatype instead of a double datatype. For more information, see **CrosstabMax**.

See also

**CrosstabAvgDec**  
**CrosstabMinDec**  
**CrosstabSumDec**

---

**CrosstabMin**

Description

Calculates the minimum value returned by an expression in the values list of the crosstab. When the crosstab definition has more than one column, **CrosstabMin** can also calculate the minimum of the expression’s values for groups of column values.

For crosstabs only

You can use this function *only* in a crosstab DataWindow object.

Syntax

**CrosstabMin** ( \( n \{, \) column, groupvalue \} )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n )</td>
<td>The number of the crosstab-values expression for which you want</td>
</tr>
<tr>
<td></td>
<td>the minimum return value. The expression’s datatype must be numeric.</td>
</tr>
</tbody>
</table>
### CrosstabMin

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>column</td>
<td>The number of the crosstab column as it is listed in the Columns box of the Crosstab Definition dialog box for which you want intermediate calculations.</td>
</tr>
<tr>
<td>groupvalue</td>
<td>A string whose value controls the grouping for the calculation. Groupvalue is usually a value from another column in the crosstab. To specify the current column value in a dynamic crosstab, rather than a specific value, specify @ plus the column name as a quoted string.</td>
</tr>
</tbody>
</table>

**Return value**

Double. Returns the minimum value returned by expression \( n \) for all the column values or, optionally, for a subset of column values. To return a decimal datatype, use `CrosstabMinDec`.

**Usage**

This function is meaningful *only* for the minimum of the values of the expression in a row in the crosstab. This means you can use it only in the detail band, not in a header, trailer, or summary band.

Null values are ignored and are not included in the comparison.

For more information about restricting the calculation to groups of values when the crosstab definition has more than one column, see Usage for `CrosstabAvg`.

**Reviewing the expressions**

To review the expressions defined for the crosstab values, open the Crosstab Definition dialog box (select Design>Crosstab from the menubar).

**Examples**

These examples all use the crosstab-values expressions shown below:

```plaintext
Count(emp_id for crosstab),Sum(salary for crosstab)
```

This expression for a computed field in the crosstab returns the minimum of the employee counts (the first expression):

```plaintext
CrosstabMin(1)
```

This expression for a computed field in the crosstab returns the minimum of the salary totals (the second expression):

```plaintext
CrosstabMin(2)
```

The next two examples use a crosstab with two columns (year and quarter), a row (product), and the values expression `Avg(sales for crosstab)`.

This expression for a computed field returns the smallest of the quarterly average sales for each year:

```plaintext
CrosstabMin(1, 2, "@year")
```
This expression for a computed field returns the minimum of all the average sales in the row:

\[ \text{CrosstabMin}(1) \]

For an example illustrating how the painter automatically defines a crosstab by creating computed fields using the crosstab functions, see \text{CrosstabAvg}.

See also

\text{CrosstabAvg}  \\
\text{CrosstabCount}  \\
\text{CrosstabMax}  \\
\text{CrosstabMinDec}  \\
\text{CrosstabSum}

---

### CrosstabMinDec

**Description**

Calculates the minimum value returned by an expression in the values list of the crosstab and returns a result with the decimal datatype. When the crosstab definition has more than one column, \text{CrosstabMinDec} can also calculate the minimum of the expression’s values for groups of column values.

**For crosstabs only**

You can use this function only in a crosstab DataWindow object.

**Syntax**

\[ \text{CrosstabMinDec}(n\{(, \text{column}, \text{groupvalue}\}) ) \]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n )</td>
<td>The number of the crosstab-values expression for which you want the minimum return value. The expression’s datatype must be numeric.</td>
</tr>
<tr>
<td>( \text{column} ) (optional)</td>
<td>The number of the crosstab column as it is listed in the Columns box of the Crosstab Definition dialog box for which you want intermediate calculations.</td>
</tr>
<tr>
<td>( \text{groupvalue} ) (optional)</td>
<td>A string whose value controls the grouping for the calculation. ( \text{Groupvalue} ) is usually a value from another column in the crosstab. To specify the current column value in a dynamic crosstab, rather than a specific value, specify @ plus the column name as a quoted string.</td>
</tr>
</tbody>
</table>

**Return value**

Decimal. Returns the minimum value returned by expression \( n \) for all the column values or, optionally, for a subset of column values.
**Usage**

Use this function instead of `CrosstabMin` when you want to return a decimal datatype instead of a double datatype. For more information, see `CrosstabMin`.

**See also**

- `CrosstabAvgDec`
- `CrosstabMaxDec`
- `CrosstabSumDec`

---

**CrosstabSum**

**Description**

Calculates the sum of the values returned by an expression in the values list of the crosstab. When the crosstab definition has more than one column, `CrosstabSum` can also calculate the sum of the expression’s values for groups of column values.

**For crosstabs only**

You can use this function *only* in a crosstab DataWindow object.

**Syntax**

```
CrosstabSum ( n {, column, groupvalue } )
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>n</code></td>
<td>The number of the crosstab-values expression for which you want the sum of the returned values. The expression’s datatype must be numeric.</td>
</tr>
<tr>
<td><code>column</code> (optional)</td>
<td>The number of the crosstab column as it is listed in the Columns box of the Crosstab Definition dialog box for which you want intermediate calculations.</td>
</tr>
<tr>
<td><code>groupvalue</code> (optional)</td>
<td>A string whose value controls the grouping for the calculation. <code>Groupvalue</code> is usually a value from another column in the crosstab. To specify the current column value in a dynamic crosstab, rather than a specific value, specify @ plus the column name as a quoted string.</td>
</tr>
</tbody>
</table>

**Return value**

Double. Returns the total of the values returned by expression `n` for all the column values or, optionally, for a subset of column values. To return a decimal datatype, use `CrosstabSumDec`.

**Usage**

This function is meaningful *only* for the sum of the values of the expression in a *row* in the crosstab. This means you can use it only in the detail band, not in a header, trailer, or summary band.

Null values are ignored and are not included in the sum.
For more information about restricting the calculation to groups of values when the crosstab definition has more than one column, see Usage for CrosstabAvg.

**Reviewing the expressions**

To review the expressions defined for the crosstab values, open the Crosstab Definition dialog box (select Design>Crosstab from the menubar).

### Examples

These examples all use the crosstab-values expressions shown below:

\[
\text{Count(emp\_id for crosstab)}, \text{Sum(salary for crosstab)}
\]

This expression for a computed field in the crosstab returns the sum of the employee counts (the first expression):

\[
\text{CrosstabSum}(1)
\]

This expression for a computed field in the crosstab returns the sum of the salary totals (the second expression):

\[
\text{CrosstabSum}(2)
\]

The next two examples use a crosstab with two columns (year and quarter), a row (product), and the values expression Avg(sales for crosstab).

This expression for a computed field returns the sum of the quarterly average sales for each year:

\[
\text{CrosstabSum}(1, 2, \"@year\")
\]

This expression for a computed field returns the sum of all the average sales in the row:

\[
\text{CrosstabSum}(1)
\]

For an example illustrating how the painter automatically defines a crosstab by creating computed fields using the Crosstab functions, see CrosstabAvg.

### See also

- CrosstabAvg
- CrosstabCount
- CrosstabMax
- CrosstabMin
- CrosstabSumDec
CrosstabSumDec

Description
Calculates the sum of the values returned by an expression in the values list of the crosstab and returns a result with the decimal datatype. When the crosstab definition has more than one column, CrosstabSumDec can also calculate the sum of the expression’s values for groups of column values.

For crosstabs only
You can use this function only in a crosstab DataWindow object.

Syntax
CrosstabSumDec(n{, column, groupvalue})

Argument | Description
--- | ---
n | The number of the crosstab-values expression for which you want the sum of the returned values. The expression’s datatype must be numeric.
column (optional) | The number of the crosstab column as it is listed in the Columns box of the Crosstab Definition dialog box for which you want intermediate calculations.
groupvalue (optional) | A string whose value controls the grouping for the calculation. Groupvalue is usually a value from another column in the crosstab. To specify the current column value in a dynamic crosstab, rather than a specific value, specify @ plus the column name as a quoted string.

Return value
Decimal. Returns the total of the values returned by expression n for all the column values or, optionally, for a subset of column values.

Usage
Use this function instead of CrosstabSum when you want to return a decimal datatype instead of a double datatype. For more information, see CrosstabSum.

See also
CrosstabAvgDec
CrosstabMaxDec
CrosstabMinDec
**CumulativePercent**

**Description**
Calculates the total value of the rows up to and including the current row in the specified column as a percentage of the total value of the column (a running percentage).

**Syntax**

```
CumulativePercent( column { FOR range } )
```

**Argument** | **Description**
--- | ---
`column` | The column for which you want the cumulative value of the rows up to and including the current row as a percentage of the total value of the column for `range`. `Column` can be the column name or the column number preceded by a pound sign (#). `Column` can also be an expression that includes a reference to the column. The datatype of `column` must be numeric.

`FOR range` (optional) | The data that will be included in the cumulative percentage. For most presentation styles, values for `range` are:
- **ALL** – (Default) The cumulative percentage of all rows in `column`.
- **GROUP n** – The cumulative percentage of rows in `column` in the specified group. Specify the keyword GROUP followed by the group number: for example, GROUP 1.
- **PAGE** – The cumulative percentage of the rows in `column` on a page.

For Crosstabs, specify CROSSTAB for `range`:
- **CROSSTAB** – (Crosstabs only) The cumulative percentage of all rows in `column` in the crosstab.

For Graph and OLE objects, specify one of the following:
- **GRAPH** – (Graphs only) The cumulative percentage of values in `column` in the range specified for the Rows option.
- **OBJECT** – (OLE objects only) The cumulative percentage of values in `column` in the range specified for the Rows option.

**Return value**
Long. Returns the cumulative percentage value.

**Usage**
If you specify `range`, CumulativePercent restarts the accumulation at the start of the range.

For graphs and OLE objects, you do not select the range when you call the function. The range has already been determined by the Rows setting on the Data property page (the Range property), and the aggregation function uses that range.
Settings for Rows include the following:

- For the Graph presentation style, Rows is always All.
- For Graph controls, Rows can be All, Page, or Group.
- For OLE controls, Rows can be All, Current Row, Page, or Group. The available choices depend on the layer the control occupies.

In calculating the percentage, null values are ignored.

**Not in validation rules or filter expressions**

You cannot use this or other aggregate functions in validation rules or filter expressions.

Using an aggregate function cancels the effect of setting Retrieve Rows As Needed in the painter. To do the aggregation, a DataWindow object always retrieves all rows.

**Examples**

This expression returns the running percentage for the values that are not null in the column named salary:

```
CumulativePercent(salary)
```

This expression returns the running percentage for the column named salary for the values in group 1 that are not null:

```
CumulativePercent(salary for group 1)
```

This expression entered in the Value box on the Data property page for a graph returns the running percentage for the salary column for the values in the graph that are not null:

```
CumulativePercent(salary for graph)
```

This expression in a crosstab computed field returns the running percentage for the salary column for the values in the crosstab that are not null:

```
CumulativePercent(salary for crosstab)
```

**See also**

Percent
CumulativeSum
CHAPTER 2  DataWindow Expression Functions

**CumulativeSum**

**Description**  
Calculates the total value of the rows up to and including the current row in the specified column (a running total).

**Syntax**  
`CumulativeSum ( column { FOR range } )`

**Argument** | **Description**  
--- | ---  
`column` | The column for which you want the cumulative total value of the rows up to and including the current row for group. `Column` can be the column name or the column number preceded by a pound sign (#). `Column` can also be an expression that includes a reference to the column. The datatype of `column` must be numeric.  

**FOR range** (optional) | The data that will be included in the cumulative sum. For most presentation styles, values for `range` are:
- ALL – (Default) The cumulative sum of all values in `column`.
- GROUP `n` – The cumulative sum of values in `column` in the specified group. Specify the keyword GROUP followed by the group number: for example, GROUP 1.
- PAGE – The cumulative sum of the values in `column` on a page. For Crosstabs, specify CROSSTAB for `range`:
  - CROSSTAB – (Crosstabs only) The cumulative sum of all values in `column` in the crosstab.
  - CROSSTAB – (Crosstabs only) The cumulative sum of all values in `column` in the crosstab. For Graph and OLE objects, specify one of the following:
  - GRAPH – (Graphs only) The cumulative sum of values in `column` in the range specified for the Rows option.
  - OBJECT – (OLE objects only) The cumulative sum of values in `column` in the range specified for the Rows option.

**Return value** | The appropriate numeric datatype. Returns the cumulative total value of the rows.

**Usage**  
If you specify `range`, CumulativeSum restarts the accumulation at the start of the range.

For graphs and OLE objects, you do not select the range when you call the function. The range has already been determined by the Rows setting on the Data property page (the Range property), and the aggregation function uses that range. Settings for Rows include the following:
- For the Graph presentation style, Rows is always All.
- For Graph controls, Rows can be All, Page, or Group.
- For OLE controls, Rows can be All, Current Row, Page, or Group. The available choices depend on the layer the control occupies.
**CurrentRow**

In calculating the sum, null values are ignored.

**Examples**

This expression returns the running total for the values that are not null in the column named salary:

\[
\text{CumulativeSum}(\text{salary})
\]

This expression returns the running total for the values that are not null in the column named salary in group 1:

\[
\text{CumulativeSum}(\text{salary for group 1})
\]

This expression entered in the Value box on the Data property page for a graph returns the running total for the salary column for the values in the graph that are not null:

\[
\text{CumulativeSum}(\text{salary for graph})
\]

This expression in a crosstab computed field returns the running total for the salary column for the values in the crosstab that are not null:

\[
\text{CumulativeSum}(\text{salary for crosstab})
\]

**See also**

CumulativePercent

---

**CurrentRow**

**Description**

Reports the number of the current row (the row with focus).

**Syntax**

\[
\text{CurrentRow}()
\]

**Return value**

Long. Returns the number of the row if it succeeds and 0 if no row is current.

**What row is current**

The current row is not always a row displayed on the screen. For example, if the cursor is on row 7 column 2 and the user uses the scroll bar to scroll to row 50, the current row remains row 7 unless the user clicks row 50.

**Examples**

This expression in a computed field returns the number of the current row:

\[
\text{CurrentRow}()
\]

This expression for a computed control displays an arrow bitmap as an indicator for the row with focus and displays no bitmap for rows not having focus. As the user moves from row to row, an arrow marks where the user is:

\[
\text{Bitmap}(\text{If} (\text{CurrentRow}() = \text{GetRow}(), "arrow.bmp", ""))
\]
Alternatively, this expression for the Visible property of an arrow picture control makes the arrow bitmap visible for the row with focus and invisible for rows not having focus. As the user moves from row to row, an arrow marks where the user is:

\[
\text{If} (\text{CurrentRow()} = \text{GetRow()} , 1, 0)
\]

See also
“Example 3: creating a row indicator” on page 19
GetRow

### Date

**Description**
Converts a string whose value is a valid date to a value of datatype date.

**Syntax**
\[
\text{Date ( string )}
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string containing a valid date (such as Jan 1, 2004, or 12-31-99) that you want returned as a date</td>
</tr>
</tbody>
</table>

**Return value**
Date. Returns the date in string as a date. If string does not contain a valid date, Date returns null.

**Usage**
The value of the string must be a valid date.

**Valid dates**
Valid dates can include any combination of day (1–31), month (1–12 or the name or abbreviation of a month), and year (two or four digits). Leading zeros are optional for month and day. If the month is a name or an abbreviation, it can come before or after the day; if it is a number, it must be in the month location specified in the Windows control panel. A 4-digit number is assumed to be a year.

If the year is two digits, the assumption of century follows this rule: for years between 00 and 49, the first two digits are assumed to be 20; for years between 50 and 99, the first two digits are assumed to be 19. If your data includes dates before 1950, such as birth dates, always specify a four-digit year to ensure the correct interpretation.

The function handles years from 1000 to 3000 inclusive.
**DateTime**

An expression has a more limited set of datatypes than the functions that can be part of the expression. Although the Date function returns a date value, the whole expression is promoted to a DateTime value. Therefore, if your expression consists of a single Date function, it will appear that Date returns the wrong datatype. To display the date without the time, choose an appropriate display format. (See “Using DataWindow expression functions” on page 13.)

**Examples**

These expressions all return the date datatype for July 4, 2004 when the default location of the month in Regional Settings is center:

- `Date("2004/07/04")`
- `Date("2004 July 4")`
- `Date("July 4, 2004")`

**See also**

IsDate

---

**DateTime**

**Description**

Combines a date and a time value into a DateTime value.

**Syntax**

```
DateTime ( date [, time ] )
```

**Argument** | **Description**
--- | ---
*date* | A valid date (such as Jan 1, 2005, or 12-31-99) or a blob variable whose first value is a date that you want included in the value returned by DateTime.

*time* (optional) | A valid time (such as 8am or 10:25:23:456799) or a blob variable whose first value is a time that you want included in the value returned by DateTime. If you include a time, only the hour portion is required. If you omit the minutes, seconds, or microseconds, they are assumed to be zeros. If you omit am or pm, the hour is determined according to the 24-hour clock.

**Return value**

DateTime. Returns a DateTime value based on the values in *date* and optionally *time*. If time is omitted, DateTime uses 00:00:00.000000 (midnight).

**Usage**

To display microseconds in a time, the display format for the field must include microseconds.

For information on valid dates, see Date.

**Examples**

This expression returns the values in the order_date and order_time columns as a DateTime value that can be used to update the database:

```
DateTime(Order_Date, Order_Time)
```
Using this expression for a computed field displays 11/11/01 11:11:00:

```
DateTime(11/11/01, 11:11)
```

See also
- Date
- Time

## Day

**Description**

Obtains the day of the month in a date value.

**Syntax**

```
Day(date)
```

**Argument** | **Description**
--- | ---
`date` | The date for which you want the day

**Return value**

Integer. Returns an integer (1–31) representing the day of the month in `date`.

**Examples**

This expression returns 31:

```
Day(2005-01-31)
```

This expression returns the day of the month in the `start_date` column:

```
Day(start_date)
```

See also
- Date
- IsDate
- Month
- Year

## DayName

**Description**

Gets the day of the week in a date value and returns the weekday’s name.

**Syntax**

```
DayName(date)
```

**Argument** | **Description**
--- | ---
`date` | The date for which you want the name of the day

**Return value**

String. Returns a string whose value is the name of the weekday (Sunday, Monday, and so on) for `date`.
**DayNumber**

This expression for a computed field returns Okay if the day in date_signed is not Sunday:

\[ \text{If(DayName(date_signed) <> "Sunday", "Okay", "Invalid Date")} \]

To pass this validation rule, the day in date_signed must not be Sunday:

\[ \text{DayName(date_signed) <> "Sunday"} \]

**Examples**

This expression for a computed field returns Wrong Day if the date in start_date is not a Sunday or a Monday:

\[ \text{If(DayNumber(start_date) > 2, "Okay", "Wrong Day")} \]

This expression for a computed field returns Wrong Day if the date in end_date is not a Saturday or a Sunday:

\[ \text{If(DayNumber(end_date) > 1 and DayNumber(end_date) < 7, "Okay", "Wrong Day")} \]

This validation rule for the column end_date ensures that the day is not a Saturday or Sunday:

\[ \text{DayNumber(end_date) > 1 and DayNumber(end_date) < 7} \]

**See also**

Date  
Day  
DayNumber  
IsDate

---

**DayNumber**

**Description**

Gets the day of the week of a date value and returns the number of the weekday.

**Syntax**

\[
\text{DayNumber ( date )}
\]

**Argument** | **Description**  
--- | ---  
\( \text{date} \) | The date from which you want the number of the day of the week  

**Return value**

Integer. Returns an integer (1–7) representing the day of the week of \( \text{date} \). Sunday is day 1, Monday is day 2, and so on.

**Examples**

This expression for a computed field returns Wrong Day if the date in start_date is not a Sunday or a Monday:

\[ \text{If(DayNumber(start_date) > 2, "Okay", "Wrong Day")} \]

This expression for a computed field returns Wrong Day if the date in end_date is not a Saturday or a Sunday:

\[ \text{If(DayNumber(end_date) > 1 and DayNumber(end_date) < 7, "Okay", "Wrong Day")} \]

This validation rule for the column end_date ensures that the day is not a Saturday or Sunday:

\[ \text{DayNumber(end_date) > 1 and DayNumber(end_date) < 7} \]

**See also**

Date  
Day  
DayName  
IsDate
DaysAfter

Description
Gets the number of days one date occurs after another.

Syntax
DaysAfter ( date1, date2 )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date1</td>
<td>A date value that is the start date of the interval being measured</td>
</tr>
<tr>
<td>date2</td>
<td>A date value that is the end date of the interval</td>
</tr>
</tbody>
</table>

Return value
Long. Returns a long containing the number of days date2 occurs after date1. If date2 occurs before date1, DaysAfter returns a negative number.

Examples
This expression returns 4:
DaysAfter(2005-12-20, 2005-12-24)

This expression returns -4:
DaysAfter(2005-12-24, 2005-12-20)

This expression returns 0:
DaysAfter(2005-12-24, 2005-12-24)

This expression returns 5:
DaysAfter(2004-12-29, 2005-01-03)

See also
Date
SecondsAfter

Dec

Description
Converts the value of a string to a decimal.

Syntax
Dec ( string )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>The string you want returned as a decimal</td>
</tr>
</tbody>
</table>

Return value
Decimal. Returns the contents of string as a decimal if it succeeds and 0 if string is not a number.
**Describe**

**Usage**

The decimal datatype supports up to 28 digits.

You can also append the letter D in upper or lowercase to identify a number as a decimal constant in DataWindow expressions. For example, 2.0d and 123.456789012345678901D are treated as decimals.

**Examples**

This expression returns the string 24.3 as a decimal datatype:

\[
\text{Dec}("24.3")
\]

This expression for a computed field returns “Not a valid score” if the string in the score column does not contain a number. The expression checks whether the Dec function returns 0, which means it failed to convert the value:

\[
\text{If ( Dec(score) <> 0, score, "Not a valid score")}
\]

This expression returns 0:

\[
\text{Dec("3ABC") // 3ABC is not a number}
\]

This validation rule checks that the value in the column the user entered is greater than 1999.99:

\[
\text{Dec(GetText()) > 1999.99}
\]

This validation rule for the column named score insures that score contains a string:

\[
\text{Dec(score) <> 0}
\]

**Describe**

Reports the values of properties of a DataWindow object and controls within the object. Each column and graphic control in the DataWindow object has a set of properties. You specify one or more properties as a string and Describe returns the values of the properties.

**Syntax**

\[
\text{Describe ( propertylist )}
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>propertylist</td>
<td>A string whose value is a blank-separated list of properties or Evaluate functions</td>
</tr>
</tbody>
</table>

**Return value**

String. Returns a string that includes a value for each property or Evaluate function. A new line character (~n) separates the value of each item in propertylist.
If propertylist contains an invalid item, Describe returns an exclamation point (!) for that item and ignores the rest of propertylist. Describe returns a question mark (?) if there is no value for a property.

Usage
Specifying the values for propertylist can be complex. For information and examples, see the Describe method for the DataWindowControl.

Examples
This expression for a computed field in the header band of a DataWindow object displays the DataWindow object’s SELECT statement:

```
Describe("DataWindow.Table.Select")
```

### Exp

Description
Raises $e$ to the specified power.

Syntax
```
Exp ( n )
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>The power to which you want to raise $e$ (2.71828)</td>
</tr>
</tbody>
</table>

Return value
Double. Returns $e$ raised to the power $n$.

Examples
This expression returns 7.38905609893065:

```
Exp (2)
```

See also
Log
LogTen

### Fact

Description
Gets the factorial of a number.

Syntax
```
Fact ( n )
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>The number for which you want the factorial</td>
</tr>
</tbody>
</table>

Return value
Double. Returns the factorial of $n$. 
Fill

Description Builds a string of the specified length by repeating the specified characters until the result string is long enough.

Syntax

\[
\text{Fill}(\text{chars}, n)
\]

Return value String. Returns a string \(n\) characters long filled with repetitions of the characters in the argument \text{chars}. If the argument \text{chars} has more than \(n\) characters, the first \(n\) characters of \text{chars} are used to fill the return string. If the argument \text{chars} has fewer than \(n\) characters, the characters in \text{chars} are repeated until the return string has \(n\) characters.

Usage Fill is used to create a line or other special effect. For example, asterisks repeated in a printed report can fill an amount line, or hyphens can simulate a total line in a screen display.

Examples

This expression returns a string containing 35 asterisks:

\[
\text{Fill}(\text{"\*"}, 35)
\]

This expression returns the string -+-+--:

\[
\text{Fill}(\text{"-+"}, 7)
\]

This expression returns 10 tildes (-):

\[
\text{Fill}(\text{"-"}, 10)
\]

See also FillA
Space
CHAPTER 2  DataWindow Expression Functions

FillA

Description
Builds a string of the specified length in bytes by repeating the specified characters until the result string is long enough.

Syntax
FillA ( chars, n )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>chars</td>
<td>A string whose value will be repeated to fill the return string</td>
</tr>
<tr>
<td>n</td>
<td>A long whose value is the number of bytes in the string you want returned</td>
</tr>
</tbody>
</table>

Return value
String. Returns a string \( n \) bytes long filled with repetitions of the characters in the argument \( \text{chars} \). If the argument \( \text{chars} \) has more than \( n \) bytes, the first \( n \) bytes of \( \text{chars} \) are used to fill the return string. If the argument \( \text{chars} \) has fewer than \( n \) bytes, the characters in \( \text{chars} \) are repeated until the return string has \( n \) bytes.

Usage
In SBCS environments, Fill and FillA return the same results.

See also
Fill

First

Description
Reports the value in the first row in the specified column.

Syntax
First ( column ( FOR range ( DISTINCT ( expresn {, expres2 {, ... } } ) ) ) )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>column</td>
<td>The column for which you want the value of the first row. Column can be a column name or a column number preceded by a pound sign (#). Column can also be an expression that includes a reference to the column.</td>
</tr>
<tr>
<td>FOR range (optional)</td>
<td>The data that will be included when the value in the first row is found. Values for range depend on the presentation style. See the Usage section for more information.</td>
</tr>
<tr>
<td>DISTINCT (optional)</td>
<td>Causes First to consider only the distinct values in ( \text{column} ) when determining the first value. For a value of ( \text{column} ), the first row found with the value is used and other rows that have the same value are ignored.</td>
</tr>
<tr>
<td>expresn (optional)</td>
<td>One or more expressions that you want to evaluate to determine distinct rows. ( \text{Expresn} ) can be the name of a column, a function, or an expression.</td>
</tr>
</tbody>
</table>

Return value
The datatype of the column. Returns the value in the first row of \( \text{column} \). If you specify \( \text{range} \), First returns the value of the first row in \( \text{column} \) in \( \text{range} \).
Usage

If you specify `range`, `First` determines the value of the first row in `column` in `range`. If you specify DISTINCT, `First` returns the first distinct value in `column`, or if you specify `expresn`, the first distinct value in `column` where the value of `expresn` is distinct.

For most presentation styles, values for `range` are:

- **ALL** – (Default) The value in the first of all rows in `column`.
- **GROUP** `n` – The value in the first of rows in `column` in the specified group. Specify the keyword GROUP followed by the group number: for example, GROUP 1.
- **PAGE** – The value in the first of the rows in `column` on a page.

For Crosstabs, specify CROSSTAB for `range` to indicate the first of all rows in `column` in the crosstab.

For Graphs specify GRAPH and for OLE objects specify OBJECT for `range`, to indicate the value in the first row in `column` in the range specified for the Rows option.

For graphs and OLE objects, you do not select the range when you call the function. The range has already been determined by the Rows setting on the Data property page (the Range property), and the aggregation function uses that range. Settings for Rows include the following:

- For the Graph presentation style, Rows is always All.
- For Graph controls, Rows can be All, Page, or Group.
- For OLE controls, Rows can be All, Current Row, Page, or Group. The available choices depend on the layer the control occupies.

**Not in validation rules or filter expressions**

You cannot use this or other aggregate functions in validation rules or filter expressions.

Using an aggregate function cancels the effect of setting Retrieve Rows As Needed in the painter. To do the aggregation, a DataWindow object always retrieves all rows.
CHAPTER 2  DataWindow Expression Functions

Examples

This expression returns the first value in column 3 on the page:

\[ \text{First} (#3 \text{ for page}) \]

This expression returns the first distinct value in the column named dept_id in group 2:

\[ \text{First}(\text{dept_id for group 2 DISTINCT}) \]

This expression returns the first value in the column named dept_id in group 2:

\[ \text{First}(\text{dept_id for group 2}) \]

See also

Last

---

GetRow

Description

Reports the number of a row associated with a band in a DataWindow object.

Syntax

\[ \text{GetRow}() \]

Return value

Long. Returns the number of a row if it succeeds, 0 if no data has been retrieved or added, and -1 if an error occurs. Where you call GetRow determines what row it returns, as follows:

<table>
<thead>
<tr>
<th>If the control in the DataWindow object is in this band</th>
<th>GetRow returns</th>
</tr>
</thead>
<tbody>
<tr>
<td>Header</td>
<td>First row on the page</td>
</tr>
<tr>
<td>Group header</td>
<td>First row in the group</td>
</tr>
<tr>
<td>Detail</td>
<td>The row in which the expression occurs</td>
</tr>
<tr>
<td>Group trailer</td>
<td>Last row in the group</td>
</tr>
<tr>
<td>Summary</td>
<td>Last row in the DataWindow object</td>
</tr>
<tr>
<td>Footer</td>
<td>Last row on the page</td>
</tr>
</tbody>
</table>

Examples

This expression for a computed field in the detail band displays the number of each row:

\[ \text{GetRow}() \]

This expression for a computed field in the header band checks to see if there is data. It returns the number of the first row on the page if there is data, and otherwise returns No Data:

\[ \text{If}(\text{GetRow}() = 0, \text{"No Data"}, \text{String(GetRow())}) \]
**GetText**

**Description**
Obtains the text that a user has entered in a column.

**Syntax**
GetText()

**Return value**
String. Returns the text the user has entered in the current column.

**Usage**
Use GetText in validation rules to compare what the user has entered to application-defined criteria before it is accepted into the data buffer.

**Examples**
This validation rule checks that the value the user entered in the column is less than 100:

\[
\text{Integer(GetText())} < 100
\]

**Hour**

**Description**
Obtains the hour in a time value. The hour is based on a 24-hour clock.

**Syntax**
Hour(time)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>The time value from which you want the hour</td>
</tr>
</tbody>
</table>

**Return value**
Integer. Returns an integer (00–23) containing the hour portion of time.

**Examples**
This expression returns the current hour:

\[
\text{Hour(Now())}
\]

This expression returns 19:

\[
\text{Hour(19:01:31)}
\]

**See also**
Minute
Now
Second
CHAPTER 2  DataWindow Expression Functions

If

Description
Evaluates a condition and returns a value based on that condition.

Syntax
\[
\text{If ( boolean, truevalue, falsevalue )}
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>boolean</td>
<td>A boolean expression that evaluates to true or false.</td>
</tr>
<tr>
<td>truevalue</td>
<td>The value you want returned if the boolean expression is true. The value can be a string or numeric value.</td>
</tr>
<tr>
<td>falsevalue</td>
<td>The value you want returned if the boolean expression is false. The value can be a string or numeric value.</td>
</tr>
</tbody>
</table>

Return value
The datatype of truevalue or falsevalue. Returns truevalue if boolean is true and falsevalue if it is false. Returns null if an error occurs.

Examples
This expression returns Boss if salary is over $100,000 and Employee if salary is less than or equal to $100,000:

\[
\text{If(} \text{salary > 100000, "Boss", "Employee"})
\]

This expression returns Boss if salary is over $100,000, Supervisor if salary is between $12,000 and $100,000, and Clerk if salary is less than or equal to $12,000:

\[
\text{If(} \text{salary > 100000, "Boss", If(} \text{salary > 12000, "Supervisor", "Clerk"})\text{)}
\]

In this example of a validation rule, the value the user should enter in the commission column depends on the price. If price is greater than or equal to 1000, then the commission is between .10 and .20. If price is less than 1000, then the commission must be between .04 and .09. The validation rule is:

\[
(\text{Number(GetText()) } \geq \text{ If(} \text{price } \geq 1000, .10, .04) \text{)} \text{ AND } (\text{Number(GetText()) } \leq \text{ If(} \text{price } \geq 1000, .20, .09) \text{)}
\]

The accompanying error message expression might be:

"Price is " + \text{If(} \text{price } \geq 1000, \"greater than or equal to", \"less than\") + \" 1000. Commission must be between \" + \text{If(} \text{price } \geq 1000, \."10", \."04") + \" and \" + \text{If(} \text{price } \geq 1000, \."20.", \."09.")"

See also
“Example 1: counting null values in a column” on page 14
“Example 2: counting male and female employees” on page 16
“Example 3: creating a row indicator” on page 19
“Example 4: displaying all data when a column allows nulls” on page 21

DataWindow Object Reference 67
**Int**

Description: Gets the largest whole number less than or equal to a number.

Syntax: `Int(n)`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>n</code></td>
<td>The number for which you want the largest whole number that is less than or equal to it</td>
</tr>
</tbody>
</table>

Return value: The datatype of `n`. Returns the largest whole number less than or equal to `n`.

Examples:

These expressions return 3.0:

- `Int(3.2)`
- `Int(3.8)`

These expressions return -4.0:

- `Int(-3.2)`
- `Int(-3.8)`

See also: Ceiling, Integer, Round, Truncate

---

**Integer**

Description: Converts the value of a string to an integer.

Syntax: `Integer(string)`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>string</code></td>
<td>The string you want returned as an integer</td>
</tr>
</tbody>
</table>

Return value: Integer. Returns the contents of `string` as an integer if it succeeds and 0 if `string` is not a number.

Examples:

This expression converts the string 24 to an integer:

- `Integer("24")`
This expression for a computed field returns “Not a valid age” if age does not contain a number. The expression checks whether the Integer function returns 0, which means it failed to convert the value:

\[
\text{If (Integer(age) <> 0, age, "Not a valid age")}
\]

This expression returns 0:

\[
\text{Integer("3ABC") // 3ABC is not a number}
\]

This validation rule checks that the value in the column the user entered is less than 100:

\[
\text{Integer(GetText()) < 100}
\]

This validation rule for the column named age insures that age contains a string:

\[
\text{Integer(age) <> 0}
\]

See also
IsNumber

### IsDate

**Description**
Tests whether a string value is a valid date.

**Syntax**

\[
\text{IsDate (datevalue)}
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>datevalue</td>
<td>A string whose value you want to test to determine whether it is a valid date</td>
</tr>
</tbody>
</table>

**Return value**
Boolean. Returns true if `datevalue` is a valid date and false if it is not.

**Examples**
This expression returns true:

\[
\text{IsDate("Jan 1, 99")}
\]

This expression returns false:

\[
\text{IsDate("Jan 32, 2005")}
\]

This expression for a computed field returns a number or 0. If the date_received column contains a valid date, the expression returns the number of the day in date_received in the computed field, and otherwise returns 0:

\[
\text{If(IsDate(String(date_received)), DayNumber(date_received), 0)}
\]
**IsExpanded**

**Description**
Tests whether a node in a TreeView DataWindow with the specified TreeView level and that includes the specified row is expanded.

**Syntax**

\[
\text{IsExpanded}(\text{long } \text{row}, \text{long } \text{level})
\]

**Argument** | **Description**
--- | ---
\text{row} | The number of the row that belongs to the node
\text{level} | The TreeView level of the node

**Return value**
Returns true if the group is expanded and false otherwise.

**Usage**
A TreeView DataWindow has several TreeView level bands that can be expanded and collapsed. You can use the \text{IsExpanded} function to test whether or not a node in a TreeView DataWindow is expanded.

**Examples**
This expression returns true if the node that contains row 3 at TreeView level 2 is expanded:

\[
\text{IsExpanded}(3, 2)
\]

**IsNull**

**Description**
Reports whether the value of a column or expression is null.

**Syntax**

\[
\text{IsNull}(\text{any})
\]

**Argument** | **Description**
--- | ---
\text{any} | A column or expression that you want to test to determine whether its value is null

**Return value**
Boolean. Returns true if \text{any} is null and false if it is not.

**Usage**
Use \text{IsNull} to test whether a user-entered value or a value retrieved from the database is null.

**Examples**
This expression returns true if either a or b is null:

\[
\text{IsNull}(a + b)
\]

This expression returns true if the value in the salary column is null:

\[
\text{IsNull}(\text{salary})
\]

This expression returns true if the value the user has entered is null:

\[
\text{IsNull}(\text{GetText}())
\]
CHAPTER 2  DataWindow Expression Functions

See also  “Example 1: counting null values in a column” on page 14
          “Example 4: displaying all data when a column allows nulls” on page 21

IsNumber

Description  Reports whether the value of a string is a number.

Syntax  \texttt{IsNumber ( string )}

\begin{table}[h]
\centering
\begin{tabular}{|l|l|}
\hline
\textbf{Argument} & \textbf{Description} \\
\hline
\textit{string} & A string whose value you want to test to determine whether it is a valid number \\
\hline
\end{tabular}
\end{table}

Return value  Boolean. Returns true if \textit{string} is a valid number and false if it is not.

Examples  This expression returns true:

\begin{verbatim}
    IsNumber("32.65")
\end{verbatim}

This expression returns false:

\begin{verbatim}
    IsNumber("A16")
\end{verbatim}

This expression for a computed field returns “Not a valid age” if age does not contain a number:

\begin{verbatim}
    If(IsNumber(age), age, "Not a valid age")
\end{verbatim}

To pass this validation rule, \textit{Age_nbr} must be a number:

\begin{verbatim}
    IsNumber(Age_nbr) = true
\end{verbatim}

See also  Integer

IsRowModified

Description  Reports whether the row has been modified.

Syntax  \texttt{IsRowModified ()}

Return value  Boolean. Returns true if the row has been modified and false if it has not.

Usage  In a DataWindow object, when you use \texttt{IsRowModified} in bands other than the detail band, it reports on a row in the detail band. See \texttt{GetRow} for a table specifying which row is associated with each band for reporting purposes.
This expression in a computed field in the detail area displays \texttt{true} or \texttt{false} to indicate whether each row has been modified:

\begin{verbatim}
IsRowModified()
\end{verbatim}

This expression defined in the Properties view for the Color property of the computed field displays the text (\texttt{true}) in red if the user has modified any value in the row:

\begin{verbatim}
If(IsRowModified(), 255, 0)
\end{verbatim}

See also
\begin{verbatim}
GetRow
\end{verbatim}

### IsRowNew

Reports whether the row has been newly inserted.

**Syntax**

\begin{verbatim}
IsRowNew()
\end{verbatim}

**Return value**

Boolean. Returns \texttt{true} if the row is new and \texttt{false} if it was retrieved from the database.

**Usage**

In a DataWindow object, when you call \texttt{IsRowNew} in bands other than the detail band, it reports on a row in the detail band. See \texttt{GetRow} for a table specifying which row is associated with each band for reporting purposes.

**Examples**

This expression defined in the Properties view for the Protect property of a column prevents the user from modifying the column unless the row has been newly inserted:

\begin{verbatim}
If(IsRowNew(), 0, 1)
\end{verbatim}

See also
\begin{verbatim}
GetRow
\end{verbatim}

### IsSelected

Determines whether the row is selected. A selected row is highlighted using reverse video.

**Syntax**

\begin{verbatim}
IsSelected()
\end{verbatim}

**Return value**

Boolean. Returns \texttt{true} if the row is selected and \texttt{false} if it is not selected.
**Usage**

When you use IsSelected in bands other than the detail band, it reports on a row in the detail band. See GetRow for a table specifying which row is associated with each band for reporting purposes.

**Examples**

This expression for a computed field in the detail area displays a bitmap if the row is selected:

\[
\text{Bitmap(If(IsSelected(), "beach.bmp", "\"))}
\]

This example allows the DataWindow object to display a salary total for all the selected rows. The expression for a computed field in the detail band returns the salary only when the row is selected so that another computed field in the summary band can add up all the selected salaries.

The expression for cf_selected_salary (the computed field in the detail band) is:

\[
\text{If(IsSelected(), salary, 0)}
\]

The expression for the computed field in the summary band is:

\[
\text{Sum(cf_selected_salary for all)}
\]

**See also**

GetRow

---

**IsTime**

**Description**

Reports whether the value of a string is a valid time value.

**Syntax**

\[
\text{IsTime ( timevalue )}
\]

**Argument** | **Description**
--- | ---
\text{timevalue} | A string whose value you want to test to determine whether it is a valid time

**Return value**

Boolean. Returns true if \text{timevalue} is a valid time and false if it is not.

**Examples**

This expression returns true:

\[
\text{IsTime("8:00:00 am")}
\]

This expression returns false:

\[
\text{IsTime("25:00")}
\]

To pass this validation rule, the value in start_time must be a time:

\[
\text{IsTime(start_time)}
\]
**Large**

**Description**
Finds a large value at a specified ranking in a column (for example, third-largest, fifth-largest) and returns the value of another column or expression based on the result.

**Syntax**
```
Large ( returnexp, column, ntop { FOR range { DISTINCT { expres1 , expres2 {, ... } } } } )
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>returnexp</code></td>
<td>The value you want returned when the large value is found. <code>Returnexp</code> includes a reference to a column, but not necessarily the column that is being evaluated for the largest value, so that a value is returned from the same row that contains the large value.</td>
</tr>
<tr>
<td><code>column</code></td>
<td>The column that contains the large value you are searching for. <code>Column</code> can be a column name or a column number preceded by a pound sign (#). <code>Column</code> can also be an expression that includes a reference to the column. The datatype of <code>column</code> must be numeric.</td>
</tr>
<tr>
<td><code>ntop</code></td>
<td>The ranking of the large value in relation to the column’s largest value. For example, when <code>ntop</code> is 2, <code>Large</code> finds the second-largest value.</td>
</tr>
</tbody>
</table>
| `FOR range` | The data that will be included when the largest value is found. For most presentation styles, values for `range` are:  
  - ALL – (Default) The largest of all values in `column`.  
  - GROUP n – The largest of values in `column` in the specified group. Specify the keyword `GROUP` followed by the group number: for example, `GROUP 1`.  
  - PAGE – The largest of the values in `column` on a page. For Crosstabs, specify `CROSSTAB` for `range`:  
    - CROSSTAB – (Crosstabs only) The largest of all values in `column` in the crosstab. For Graph objects, specify one of the following:  
      - GRAPH – (Graphs only) The largest of values in `column` in the range specified for the Rows option.  
| `DISTINCT`  | Causes `Large` to consider only the distinct values in `column` when determining the large value. For a value of `column`, the first row found with the value is used and other rows that have the same value are ignored. |
| `expresn`  | One or more expressions that you need to evaluate to determine distinct rows. `Expresn` can be the name of a column, a function, or an expression. |

**Return value**
The datatype of `returnexp`. Returns the `ntop`-largest value if it succeeds and -1 if an error occurs.
CHAPTER 2  DataWindow Expression Functions

Usage

If you specify range, Large returns the value in returnexp when the value in column is the ntop-largest value in range. If you specify DISTINCT, Large returns returnexp when the value in column is the ntop-largest value of the distinct values in column, or if you specify expresn, the ntop-largest for each distinct value of expresn.

For graphs, you do not select the range when you call the function. The range has already been determined by the Rows setting on the Data property page (the Range property), and the aggregation function uses that range. Settings for Rows are as follows:

- For the Graph presentation style, Rows is always All
- For Graph controls, Rows can be All, Page, or Group

Max might be faster

If you do not need a return value from another column and you want to find the largest value (ntop = 1), use Max; it is faster.

You cannot use this or other aggregate functions in validation rules or filter expressions.

Using an aggregate function cancels the effect of setting Retrieve Rows As Needed in the painter. To do the aggregation, a DataWindow object always retrieves all rows.

Examples

These expressions return the names of the salespersons with the three largest sales (sum_sales is the sum of the sales for each salesperson) in group 2, which might be the salesregion group. Note that sum_sales contains the values being compared, but Large returns a value in the name column:

```
Large(name, sum_sales, 1 for group 2)
Large(name, sum_sales, 2 for group 2)
Large(name, sum_sales, 3 for group 2)
```

This example reports the salesperson with the third-largest sales, considering only the first entry for each person:

```
Large(name, sum_sales, 3 for all DISTINCT sum_sales)
```

See also

Small
Last

Description
Gets the value in the last row in the specified column.

Syntax
Last ( column { FOR range { DISTINCTT { expres1 {, expres2 {, ... } } } } } )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>column</td>
<td>The column for which you want the value of the last row. Column can be a column name or a column number preceded by a pound sign (#). Column can also be an expression that includes a reference to the column.</td>
</tr>
<tr>
<td>FOR range</td>
<td>The data that will be included when the value in the last row is found. For most presentation styles, values for range are:</td>
</tr>
<tr>
<td>(optional)</td>
<td>• ALL – (Default) The value in the last of all rows in column.</td>
</tr>
<tr>
<td></td>
<td>• GROUP n – The value in the last row in column in the specified group. Specify the keyword GROUP followed by the group number: for example, GROUP 1.</td>
</tr>
<tr>
<td></td>
<td>• PAGE – The value in the last row in column on a page.</td>
</tr>
<tr>
<td>DISTINCT</td>
<td>Causes Last to consider only the distinct values in column when determining the last value. For a value of column, the first row found with the value is used and other rows that have the same value are ignored.</td>
</tr>
<tr>
<td>(optional)</td>
<td>expresn (optional) One or more expressions that you want to evaluate to determine distinct rows. Expresn can be the name of a column, a function, or an expression.</td>
</tr>
</tbody>
</table>

Return value
The datatype of the column. Returns the value in the last row of column. If you specify range, Last returns the value of the last row in column in range.

Usage
If you specify range, Last determines the value of the last row in column in range. If you specify DISTINCT, Last returns the last distinct value in column, or if you specify expresn, the last distinct value in column where the value of expresn is distinct.
For graphs, you do not select the range when you call the function. The range has already been determined by the Rows setting on the Data property page (the Range property), and the aggregation function uses that range. Settings for Rows include the following:

- For the Graph presentation style, Rows is always All.
- For Graph controls, Rows can be All, Page, or Group.

**Not in validation rules or filter expressions**
You cannot use this or other aggregate functions in validation rules or filter expressions.

Using an aggregate function cancels the effect of setting Retrieve Rows As Needed in the painter. To do the aggregation, a DataWindow object always retrieves all rows.

**Examples**
This expression returns the last distinct value in the column named dept_id in group 2:

```
Last(dept_id for group 2 DISTINCT)
```

This expression returns the last value in the column named emp_id in group 2:

```
Last(emp_id for group 2)
```

**See also**
First

### LastPos

**Description**
Finds the last position of a target string in a source string.

**Syntax**
```
LastPos ( string1, string2, searchlength )
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string1</td>
<td>The string in which you want to find string2.</td>
</tr>
<tr>
<td>string2</td>
<td>The string you want to find in string1.</td>
</tr>
<tr>
<td>searchlength</td>
<td>A long that limits the search to the leftmost searchlength characters of the source string string1. The default is the entire string. (optional)</td>
</tr>
</tbody>
</table>

DataWindow Object Reference 77
Return value
Long. Returns a long whose value is the starting position of the last occurrence of `string2` in `string1` within the characters specified in `searchlength`. If `string2` is not found in `string1` or if `searchlength` is 0, `LastPos` returns 0. If any argument’s value is null, `LastPos` returns null.

Usage
The `LastPos` function is case sensitive. The entire target string must be found in the source string.

Examples
This statement returns 6, because the position of the last occurrence of RU is position 6:

```
LastPos("BABE RUTH", "RU")
```
This statement returns 3:

```
LastPos("BABE RUTH", "B")
```
This statement returns 0, because the case does not match:

```
LastPos("BABE RUTH", "be")
```
This statement searches the leftmost 4 characters and returns 0, because the only occurrence of RU is after position 4:

```
LastPos("BABE RUTH", "RU", 2)
```

See also
Pos

---

**Left**

Description
Obtains a specified number of characters from the beginning of a string.

Syntax
```
Left ( string, n )
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>string</code></td>
<td>The string containing the characters you want</td>
</tr>
<tr>
<td><code>n</code></td>
<td>A long specifying the number of characters you want</td>
</tr>
</tbody>
</table>

Return value
String. Returns the leftmost `n` characters in `string` if it succeeds and the empty string ("") if an error occurs.

If `n` is greater than or equal to the length of the string, `Left` returns the entire string. It does not add spaces to make the return value’s length equal to `n`. 
Examples

This expression returns BABE:

\[ \text{Left}(\text{"BABE RUTH"}, 4) \]

This expression returns BABE RUTH:

\[ \text{Left}(\text{"BABE RUTH"}, 40) \]

This expression for a computed field returns the first 40 characters of the text in the column home_address:

\[ \text{Left(\text{home_address}, 40)} \]

See also

LeftA
Mid
Pos
Right

**LeftA**

Description
Obtains a specified number of bytes from the beginning of a string.

Syntax

\[ \text{LeftA}(\text{string}, n) \]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>The string containing the characters you want</td>
</tr>
<tr>
<td>n</td>
<td>A long specifying the number of bytes you want</td>
</tr>
</tbody>
</table>

Return value
String. Returns the characters in the leftmost \( n \) bytes in \text{string} if it succeeds and the empty string (""") if an error occurs.

If \( n \) is greater than or equal to the length of the string, \text{LeftA} returns the entire string. It does not add spaces to make the return value’s length equal to \( n \).

Usage
In SBCS environments, \text{Left} and \text{LeftA} return the same results.

See also
MidA
PosA
RightA
**LeftTrim**

**Description**
Removes spaces from the beginning of a string.

**Syntax**

```
LeftTrim ( string )
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>The string you want returned with leading spaces deleted</td>
</tr>
</tbody>
</table>

**Return value**

String. Returns a copy of `string` with leading spaces deleted if it succeeds and the empty string ("") if an error occurs.

**Examples**

This expression returns RUTH:

```
LeftTrim(" RUTH")
```

This expression for a computed field deletes any leading blanks from the value in the column lname and returns the value preceded by the salutation specified in salut_emp:

```
salut_emp + " " + LeftTrim(lname)
```

**See also**

RightTrim
Trim

---

**Len**

**Description**
Reports the length of a string in characters.

**Syntax**

```
Len ( string )
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>The string for which you want the length</td>
</tr>
</tbody>
</table>

**Return value**

Long. Returns a long containing the length of `string` in characters if it succeeds and -1 if an error occurs.

**Examples**

This expression returns 0:

```
Len(" ")
```

This validation rule tests that the value the user entered is fewer than 20 characters:

```
Len(GetText()) < 20
```

**See also**

LenA
**LenA**

**Description**
Reports the length of a string in bytes.

**Syntax**
`LenA (string)`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>string</code></td>
<td>The string for which you want the length</td>
</tr>
</tbody>
</table>

**Return value**
Long. Returns a long containing the length of `string` in bytes if it succeeds and -1 if an error occurs.

**Usage**
In SBCS environments, `Len` and `LenA` return the same results.

**See also**
Len

---

**Log**

**Description**
Gets the natural logarithm of a number.

**Syntax**
`Log (n)`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>n</code></td>
<td>The number for which you want the natural logarithm (base e). The value of <code>n</code> must be greater than 0.</td>
</tr>
</tbody>
</table>

**Return value**
Double. Returns the natural logarithm of `n`. An execution error occurs if `n` is negative or zero.

**Inverse**
The inverse of the Log function is the Exp function.

**Examples**
This expression returns 2.302585092:

```
Log(10)
```

This expression returns -.693147 ...

```
Log(0.5)
```

Both these expressions result in an error at runtime:

```
Log(0)
Log(-2)
```

**See also**
Exp
LogTen

Description
Gets the base 10 logarithm of a number.

Syntax
```
LogTen(n)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>The number for which you want the base 10 logarithm. The value of n must not be negative.</td>
</tr>
</tbody>
</table>

Return value
Double. Returns the base 10 logarithm.

Obtaining a number
The expression $10^n$ is the inverse for `LogTen(n)`. To obtain $n$ given number $(nbr = \text{LogTen}(n))$, use $n = 10^{nbr}$.

Examples
This expression returns 1:
```
LogTen(10)
```
The following expressions both return 0:
```
LogTen(1)
LogTen(0)
```
This expression results in an execution error:
```
LogTen(-2)
```

See also
Log

Long

Description
Converts the value of a string to a long.

Syntax
```
Long(string)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>The string you want returned as a long</td>
</tr>
</tbody>
</table>
Return value | Long. Returns the contents of string as a long if it succeeds and 0 if string is not a valid number.

Examples | This expression returns 2167899876 as a long:

Long("2167899876")

---

**LookUpDisplay**

Description | Obtains the display value in the code table associated with the data value in the specified column.

Syntax | LookUpDisplay ( column )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>column</td>
<td>The column for which you want the code table display value</td>
</tr>
</tbody>
</table>

Return value | String. Returns the display value when it succeeds and the empty string ("") if an error occurs.

Usage | If a column has a code table, a buffer stores a value from the data column of the code table, but the user sees a value from the display column. Use LookUpDisplay to get the value the user sees.

**Code tables and data values and graphs**

When a column that is displayed in a graph has a code table, the graph displays the data values of the code table by default. To display the display values, call this function when you define the graph data.

Examples | This expression returns the display value for the column unit_measure:

LookUpDisplay(unit_measure)

Assume the column product_type has a code table and you want to use it as a category for a graph. To display the product type descriptions instead of the data values in the categories, enter this expression in the Category option on the Data page in the graph’s property sheet:

LookUpDisplay(product_type)
Lower

Description
Converts all the characters in a string to lowercase.

Syntax
Lower ( string )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>The string you want to convert to lowercase letters</td>
</tr>
</tbody>
</table>

Return value
String. Returns string with uppercase letters changed to lowercase if it succeeds and the empty string ("") if an error occurs.

Examples
This expression returns castle hill:

Lower("Castle Hill")

See also
Upper

Match

Description
Determines whether a string’s value contains a particular pattern of characters.

Syntax
Match ( string, textpattern )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>The string in which you want to look for a pattern of characters</td>
</tr>
<tr>
<td>textpattern</td>
<td>A string whose value is the text pattern</td>
</tr>
</tbody>
</table>

Return value
Boolean. Returns true if string matches textpattern and false if it does not. Match also returns false if either argument has not been assigned a value or the pattern is invalid.

Usage
Match enables you to evaluate whether a string contains a general pattern of characters. To find out whether a string contains a specific substring, use the Pos function.

Textpattern is similar to a regular expression. It consists of metacharacters, which have special meaning, and ordinary characters, which match themselves. You can specify that the string begin or end with one or more characters from a set, or that it contain any characters except those in a set.

A text pattern consists of metacharacters, which have special meaning in the match string, and nonmetacharacters, which match the characters themselves.
The following tables explain the meaning and use of these metacharacters:

<table>
<thead>
<tr>
<th>Metacharacter</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>caret (^)</td>
<td>Matches the beginning of a string</td>
<td>^C matches C at the beginning of a string.</td>
</tr>
<tr>
<td>dollar sign ($)</td>
<td>Matches the end of a string</td>
<td>s$ matches s at the end of a string.</td>
</tr>
<tr>
<td>period (.)</td>
<td>Matches any character</td>
<td>. . . matches three consecutive characters.</td>
</tr>
<tr>
<td>backslash ()</td>
<td>Removes the following metacharacter’s special characteristics so that it matches itself</td>
<td>$ matches $.</td>
</tr>
<tr>
<td>character class (a group of characters enclosed in square brackets [ ])</td>
<td>Matches any of the enclosed characters</td>
<td>[AEIOU] matches A, E, I, O, or U.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>You can use hyphens to abbreviate ranges of characters in a character class. For example, [A-Za-z] matches any letter.</td>
</tr>
<tr>
<td>complemented character class (first character inside the square brackets is a caret)</td>
<td>Matches any character not in the group following the caret</td>
<td>[^0-9] matches any character except a digit, and[^A-Za-z] matches any character except a letter.</td>
</tr>
</tbody>
</table>

The metacharacters asterisk (*), plus (+), and question mark (?) are unary operators that are used to specify repetitions in a regular expression:

<table>
<thead>
<tr>
<th>Metacharacter</th>
<th>Meaning</th>
<th>Example</th>
</tr>
</thead>
<tbody>
<tr>
<td>* (asterisk)</td>
<td>Indicates zero or more occurrences</td>
<td>A* matches zero or more As (no As, A, AA, AAA, and so on)</td>
</tr>
<tr>
<td>+ (plus)</td>
<td>Indicates one or more occurrences</td>
<td>A+ matches one A or more than one A (A, AAA, and so on)</td>
</tr>
<tr>
<td>? (question mark)</td>
<td>Indicates zero or one occurrence</td>
<td>A? matches an empty string (&quot;&quot;&quot;) or A</td>
</tr>
</tbody>
</table>
**Sample patterns**  The following table shows various text patterns and sample text that matches each pattern:

<table>
<thead>
<tr>
<th>This pattern</th>
<th>Matches</th>
</tr>
</thead>
<tbody>
<tr>
<td>AB</td>
<td>Any string that contains AB, such as ABA, DEABC, graphAB_one.</td>
</tr>
<tr>
<td>B*</td>
<td>Any string that contains 0 or more Bs, such as AC, B, BB, BBB, ABBBC, and so on. Since B* used alone matches any string, you would not use it alone, but notice its use in some the following examples.</td>
</tr>
<tr>
<td>AB*C</td>
<td>Any string containing the pattern AC or ABC or ABBC, and so on (0 or more Bs).</td>
</tr>
<tr>
<td>AB+C</td>
<td>Any string containing the pattern ABC or ABBC or ABBBC, and so on (1 or more Bs).</td>
</tr>
<tr>
<td>ABB*C</td>
<td>Any string containing the pattern ABC or ABBC or ABBBC, and so on (1 B plus 0 or more Bs).</td>
</tr>
<tr>
<td>^AB</td>
<td>Any string starting with AB.</td>
</tr>
<tr>
<td>AB?C</td>
<td>Any string containing the pattern AC or ABC (0 or 1 B).</td>
</tr>
<tr>
<td>^[ABC]</td>
<td>Any string starting with A, B, or C.</td>
</tr>
<tr>
<td>*[^ABC]</td>
<td>A string containing any characters other than A, B, or C.</td>
</tr>
<tr>
<td>*[^abc]</td>
<td>A string that begins with any character except a, b, or c.</td>
</tr>
<tr>
<td>*[a-z]$</td>
<td>Any single-character string that is not a lowercase letter (^ and $ indicate the beginning and end of the string).</td>
</tr>
<tr>
<td>[A-Z]+</td>
<td>Any string with one or more uppercase letters.</td>
</tr>
<tr>
<td>*[0-9]+$</td>
<td>Any string consisting only of digits.</td>
</tr>
<tr>
<td>*[0-9][0-9][0-9]$</td>
<td>Any string consisting of exactly three digits.</td>
</tr>
<tr>
<td>*[0-9][0-9][0-9]$</td>
<td>Any string consisting of exactly three digits enclosed in parentheses.</td>
</tr>
</tbody>
</table>

**Examples**

This validation rule checks that the value the user entered begins with an uppercase letter. If the value of the expression is false, the data fails validation:

```
Match(GetText(), "^[A-Z]")
```

**See also**

Pos
## Max

**Description**

Gets the maximum value in the specified column.

**Syntax**

```
Max ( column { FOR range { DISTINCT { expres1 {, expres2 {, ... } } } } } )
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>column</code></td>
<td>The column for which you want the maximum value. Column can be the column name or the column number preceded by a pound sign (#). Column can also be an expression that includes a reference to the column. The datatype of <code>column</code> must be numeric.</td>
</tr>
</tbody>
</table>
| `FOR range` (optional) | The data that will be included when the maximum value is found. For most presentation styles, values for `range` are:  
- ALL – (Default) The maximum value of all rows in `column`.  
- GROUP `n` – The maximum value of rows in `column` in the specified group. Specify the keyword GROUP followed by the group number: for example, GROUP 1.  
- PAGE – The maximum value of the rows in `column` on a page. For Crosstabs, specify CROSSTAB for `range`:  
  - CROSSTAB – (Crosstabs only) The maximum value of all rows in `column` in the crosstab.  
  - CROSSTAB FOR range:  
    - CROSSTAB – (Crosstabs only) The maximum value of all rows in `column` in the crosstab.  
  - CROSSTAB – (Crosstabs only) The maximum value in `column` in the range specified for the Rows option.  
| `DISTINCT` (optional) | Causes Max to consider only the distinct values in `column` when determining the largest value. For a value of `column`, the first row found with the value is used and other rows that have the same value are ignored. |
| `expresn` (optional) | One or more expressions that you want to evaluate to determine distinct rows. `Expresn` can be the name of a column, a function, or an expression. |

**Return value**

The datatype of the column. Returns the maximum value in the rows of `column`. If you specify `range`, Max returns the maximum value in `column` in `range`.

**Usage**

If you specify `range`, Max determines the maximum value in `column` in `range`. If you specify `DISTINCT`, Max returns the maximum distinct value in `column`, or if you specify `expresn`, the maximum distinct value in `column` where the value of `expresn` is distinct.
For graphs, you do not select the range when you call the function. The range has already been determined by the Rows setting on the Data property page (the Range property), and the aggregation function uses that range. Settings for Rows include the following:

- For the Graph presentation style, Rows is always All.
- For Graph controls, Rows can be All, Page, or Group.

Null values are ignored and are not considered in determining the maximum.

**Not in validation rules or filter expressions**
You cannot use this or other aggregate functions in validation rules or filter expressions.

Using an aggregate function cancels the effect of setting Retrieve Rows As Needed in the painter. To do the aggregation, a DataWindow object always retrieves all rows.

**Examples**

This expression returns the maximum of the values in the age column on the page:

```
Max(age for page)
```

This expression returns the maximum of the values in column 3 on the page:

```
Max(#3 for page)
```

This expression returns the maximum of the values in the column named age in group 1:

```
Max(age for group 1)
```

Assuming a DataWindow object displays the order number, amount, and line items for each order, this computed field returns the maximum of the order amount for the distinct order numbers:

```
Max(order_amt for all DISTINCT order_nbr)
```
Median

Calculates the median of the values of the column. The median is the middle value in the set of values, for which there is an equal number of values greater and smaller than it.

Syntax

```
Median ( column { FOR range { DISTINCT { expres1 {, expres2 {, ... } } } } } )
```

**Argument** | **Description**
--- | ---
`column` | The column for which you want the median of the data values. *Column* can be the column name or the column number preceded by a pound sign (#). *Column* can also be an expression that includes a reference to the column. The datatype of *column* must be numeric.

For range (optional) | The data that will be included in the median. For most presentation styles, values for *range* are:
- **ALL** – (Default) The median of all values in *column*.
- **GROUP n** – The median of values in *column* in the specified group. Specify the keyword GROUP followed by the group number: for example, GROUP 1.
- **PAGE** – The median of the values in *column* on a page.

For Crosstabs, specify CROSSTAB for *range*:
- **CROSSTAB** – (Crosstabs only) The median of all values in *column* in the crosstab.

For Graph objects, specify the following:
- **GRAPH** – (Graphs only) The median of values in *column* in the range specified for the Rows.

**DISTINCT** (optional) | Causes Median to consider only the distinct values in *column* when determining the median. For a value of *column*, the first row found with the value is used and other rows that have the same value are ignored.

**expresn** (optional) | One or more expressions that you want to evaluate to determine distinct rows. *Expresn* can be the name of a column, a function, or an expression.

**Return value** | The numeric datatype of the column. Returns the median of the values of the rows in *range* if it succeeds and -1 if an error occurs.

**Usage** | If you specify *range*, Median returns the median value of *column* in *range*. If you specify DISTINCT, Median returns the median value of the distinct values in *column*, or if you specify *expresn*, the median of *column* for each distinct value of *expresn*. 
**Median**

For graphs objects, you do not select the range when you call the function. The range has already been determined by the Rows setting on the Data property page (the Range property), and the aggregation function uses that range.

Settings for Rows include the following:

- For the Graph presentation style, Rows is always All.
- For Graph controls, Rows can be All, Page, or Group.

In calculating the median, null values are ignored.

---

**Not in validation rules or filter expressions**

You cannot use this or other aggregate functions in validation rules or filter expressions.

---

**Examples**

This expression returns the median of the values in the column named salary:

```
Median(salary)
```

This expression returns the median of the values in the column named salary of group 1:

```
Median(salary for group 1)
```

This expression returns the median of the values in column 5 on the current page:

```
Median(#5 for page)
```

This computed field returns Above Median if the median salary for the page is greater than the median for the report:

```
If(Median(salary for page) > Median(salary), "Above Median", "")
```

This expression for a graph value sets the data value to the median value of the sale_price column:

```
Median(sale_price)
```

This expression for a graph value entered on the Data page in the graph’s property sheet sets the data value to the median value of the sale_price column for the entire graph:

```
Median(sale_price for graph)
```
Assuming a DataWindow object displays the order number, amount, and line items for each order, this computed field returns the median of the order amount for the distinct order numbers:

\[ \text{Median} \left( \text{order}\_\text{amt} \text{ for all DISTINCT order}\_\text{nbr} \right) \]

See also

Avg
Mode

**Mid**

Description

Obtains a specified number of characters from a specified position in a string.

Syntax

\[ \text{Mid} \left( \text{string}, \text{start}, \{, \text{length} \} \right) \]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>The string from which you want characters returned.</td>
</tr>
<tr>
<td>start</td>
<td>A long specifying the position of the first character you want returned (the position of the first character of the string is 1).</td>
</tr>
<tr>
<td>length</td>
<td>(optional) A long whose value is the number of characters you want returned. If you do not enter length or if length is greater than the number of characters to the right of start, Mid returns the remaining characters in the string.</td>
</tr>
</tbody>
</table>

Return value

String. Returns characters specified in length of string starting at character start. If start is greater than the number of characters in string, the Mid function returns the empty string ("""). If length is greater than the number of characters remaining after the start character, Mid returns the remaining characters. The return string is not filled with spaces to make it the specified length.

Examples

This expression returns "":

\[ \text{Mid} \left( \text{"BABE RUTH"}, \ 40, \ 5 \right) \]

This expression returns BE RUTH:

\[ \text{Mid} \left( \text{"BABE RUTH"}, \ 3 \right) \]

This expression in a computed field returns ACCESS DENIED if the fourth character in the column password is not R:

\[ \text{If} \left( \text{Mid} \left( \text{password}, \ 4, \ 1 \right) = \text{"R"}, \ "\text{ENTER}" , \ "\text{ACCESS DENIED}" \right) \]
To pass this validation rule, the fourth character in the column password must be 6:

\[ \text{Mid}(\text{password}, 4, 1) = "6" \]

**MidA**

**Description**
Obtains a specified number of bytes from a specified position in a string.

**Syntax**

\[ \text{MidA} ( \text{string}, \text{start} \{, \text{length} \} ) \]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>The string from which you want characters returned.</td>
</tr>
<tr>
<td>start</td>
<td>A long specifying the position of the first byte you want returned (the position of the first byte of the string is 1).</td>
</tr>
<tr>
<td>length</td>
<td>(optional) A long whose value is the number of bytes you want returned. If you do not enter length or if length is greater than the number of bytes to the right of start, MidA returns the remaining bytes in the string.</td>
</tr>
</tbody>
</table>

**Return value**
String. Returns characters specified by the number of bytes in length of string starting at the byte specified by start. If start is greater than the number of bytes in string, the MidA function returns the empty string (""). If length is greater than the number of bytes remaining after the start byte, MidA returns the remaining bytes. The return string is not filled with spaces to make it the specified length.

**Usage**
In SBCS environments, Mid and MidA return the same results.

**See also**
Mid

**Min**

**Description**
Gets the minimum value in the specified column.

**Syntax**

\[ \text{Min} ( \text{column} \{ \text{FOR range} \{ \text{DISTINCT} \{ \text{expres1} \{, \text{expres2} \{, \ldots} \} \} \} \} ) \]

---

92

DataWindow .NET
CHAPTER 2  DataWindow Expression Functions

Return value

The datatype of the column. Returns the minimum value in the rows of column. If you specify range, Min returns the minimum value in the rows of column in range.

Usage

If you specify range, Min determines the minimum value in column in range. If you specify DISTINCT, Min returns the minimum distinct value in column, or if you specify expresn, the minimum distinct value in column where the value of expresn is distinct.

For graphs, you do not select the range when you call the function. The range has already been determined by the Rows setting on the Data property page (the Range property), and the aggregation function uses that range. Settings for Rows include:

- For the Graph presentation style, Rows is always All.
- For Graph controls, Rows can be All, Page, or Group.

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>column</td>
<td>The column for which you want the minimum value. Column can be the column name or the column number preceded by a pound sign (#). Column can also be an expression that includes a reference to the column. The datatype of column must be numeric.</td>
</tr>
</tbody>
</table>
| FOR range (optional) | The data that will be included in the minimum. For most presentation styles, values for range are:
  - ALL – (Default) The minimum of all values in column.
  - GROUP n – The minimum of values in column in the specified group. Specify the keyword GROUP followed by the group number: for example, GROUP 1.
  - PAGE – The minimum of the values in column on a page.
  For Crosstabs, specify CROSSTAB for range:
    - CROSSTAB – (Crosstabs only) The minimum of all values in column in the crosstab.
  For Graph objects, specify the following:
    - GRAPH – (Graphs only) The minimum of values in column in the range specified for the Rows option. |
| DISTINCT (optional) | Causes Min to consider only the distinct values in column when determining the minimum value. For a value of column, the first row found with the value is used and other rows that have the same value are ignored. |
| expresn (optional) | One or more expressions that you want to evaluate to determine distinct rows. Expresn can be the name of a column, a function, or an expression. |
Null values are ignored and are not considered in determining the minimum.

**Not in validation rules or filter expressions**
You cannot use this or other aggregate functions in validation rules or filter expressions.

Using an aggregate function cancels the effect of setting Retrieve Rows As Needed in the painter. To do the aggregation, a DataWindow object always retrieves all rows.

**Examples**

This expression returns the minimum value in the column named `age` in group 2:

\[
\text{Min}(\text{age for group 2})
\]

This expression returns the minimum of the values in column 3 on the page:

\[
\text{Min}(\#3 \text{ for page})
\]

Assuming a DataWindow object displays the order number, amount, and line items for each order, this computed field returns the minimum of the order amount for the distinct order numbers:

\[
\text{Min}(\text{order_amt for all DISTINCT order_nbr})
\]

See also

Max

---

**Minute**

**Description**
Obtains the number of minutes in the minutes portion of a time value.

**Syntax**

\[
\text{Minute}(\ time)\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>time</code></td>
<td>The time value from which you want the minutes</td>
</tr>
</tbody>
</table>

**Return value**
Integer. Returns the minutes portion of `time` (00 to 59).

**Examples**

This expression returns 1:

\[
\text{Minute}(19:01:31)
\]

See also

Hour
Second
**Mod**

Description: Obtains the remainder (modulus) of a division operation.

Syntax: \( \text{Mod} (x, y) \)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( x )</td>
<td>The number you want to divide by ( y )</td>
</tr>
<tr>
<td>( y )</td>
<td>The number you want to divide into ( x )</td>
</tr>
</tbody>
</table>

Return value: The datatype of \( x \) or \( y \), whichever datatype is more precise.

Examples:
- This expression returns 2:
  \[ \text{Mod}(20, 6) \]
- This expression returns 1.5:
  \[ \text{Mod}(25.5, 4) \]
- This expression returns 2.5:
  \[ \text{Mod}(25, 4.5) \]

**Mode**

Description: Calculates the mode of the values of the column. The mode is the most frequently occurring value.

Syntax: \( \text{Mode} (\text{column} \{ \text{FOR} \text{range} \{ \text{DISTINCT} \{ \text{expres1}, \text{expres2}, \ldots \} \} \}) \)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( \text{column} )</td>
<td>The column for which you want the mode of the data values. ( \text{Column} ) can be the column name or the column number preceded by a pound sign (#). ( \text{Column} ) can also be an expression that includes a reference to the column. The datatype of ( \text{column} ) must be numeric.</td>
</tr>
</tbody>
</table>
### Mode

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FOR range (optional)</td>
<td>The data that will be included in the mode. For most presentation styles, values for range are:</td>
</tr>
<tr>
<td></td>
<td>• ALL – (Default) The mode of all values in column.</td>
</tr>
<tr>
<td></td>
<td>• GROUP n – The mode of values in column in the specified group. Specify the keyword GROUP followed by the group number: for example, GROUP 1.</td>
</tr>
<tr>
<td></td>
<td>• PAGE – The mode of the values in column on a page.</td>
</tr>
<tr>
<td></td>
<td>For Crosstabs, specify CROSSTAB for range:</td>
</tr>
<tr>
<td></td>
<td>• CROSSTAB – (Crosstabs only) The mode of all values in column in the crosstab.</td>
</tr>
<tr>
<td></td>
<td>For Graph objects, specify:</td>
</tr>
<tr>
<td></td>
<td>• GRAPh – (Graphs only) The mode of values in column in the range specified for the Rows option.</td>
</tr>
<tr>
<td>DISTINCT (optional)</td>
<td>Causes Mode to consider only the distinct values in column when determining the mode. For a value of column, the first row found with the value is used and other rows that have the same value are ignored.</td>
</tr>
<tr>
<td>expresn (optional)</td>
<td>One or more expressions that you want to evaluate to determine distinct rows. Expresn can be the name of a column, a function, or an expression.</td>
</tr>
</tbody>
</table>

**Return value**

The numeric datatype of the column. Returns the mode of the values of the rows in range if it succeeds and -1 if an error occurs.

**Usage**

If you specify range, Mode returns the mode of column in range. If you specify DISTINCT, Mode returns the mode of the distinct values in column, or if you specify expresn, the mode of column for each distinct value of expresn.

For graphs objects, you do not select the range when you call the function. The range has already been determined by the Rows setting on the Data property page (the Range property), and the aggregation function uses that range. Settings for Rows include:

- For the Graph presentation style, Rows is always All.
- For Graph controls, Rows can be All, Page, or Group.

In calculating the mode, null values are ignored.

**Not in validation rules or filter expressions**

You cannot use this or other aggregate functions in validation rules or filter expressions.
Using an aggregate function cancels the effect of setting Retrieve Rows As Needed in the painter. To do the aggregation, a DataWindow object always retrieves all rows.

**Examples**

This expression returns the mode of the values in the column named salary:

```
Mode(salary)
```

This expression returns the mode of the values for group 1 in the column named salary:

```
Mode(salary for group 1)
```

This expression returns the mode of the values in column 5 on the current page:

```
Mode(#5 for page)
```

This computed field returns Above Mode if the mode of the salary for the page is greater than the mode for the report:

```
If(Mode(salary for page) > Mode(salary), "Above Mode","")
```

This expression for a graph value sets the data value to the mode of the sale_price column:

```
Mode(sale_price)
```

This expression for a graph value entered on the Data page in the graph’s property sheet sets the data value to the mode of the sale_price column for the entire graph:

```
Mode(sale_price for graph)
```

Assuming a DataWindow object displays the order number, amount, and line items for each order, this computed field returns the mode of the order amount for the distinct order numbers:

```
Mode(order_amt for all DISTINCT order_nbr)
```

**See also**

Avg
Median
Month

Description
Gets the month of a date value.

Syntax

\texttt{Month (date)}

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{date}</td>
<td>The date from which you want the month</td>
</tr>
</tbody>
</table>

Return value
Integer. Returns an integer (1 to 12) whose value is the month portion of \texttt{date}.

Examples
This expression returns 1:

\texttt{Month(2005-01-31)}

This expression for a computed column returns Wrong Month if the month in the column \texttt{expected\_grad\_date} is not 6:

\texttt{If(Month(expected\_grad\_date) = 6, "June", "Wrong Month")}

This validation rule expression checks that the value of the month in the date in the column \texttt{expected\_grad\_date} is 6:

\texttt{Month(expected\_grad\_date) = 6}

See also
Day
Date
Year

Now

Description
Obtains the current time based on the system time of the client machine.

Syntax

\texttt{Now ()}

Return value
Time. Returns the current time based on the system time of the client machine.

Usage
Use \texttt{Now} to compare a time to the system time or to display the system time on the screen. The timer interval specified for the DataWindow object determines the frequency at which the value of \texttt{Now} is updated. For example, if the timer interval is one second, it is updated every second. The default timer interval is one minute (60,000 milliseconds).
Examples

This expression returns the current system time:

\[ \text{Now}() \]

This expression sets the column value to 8:00 when the current system time is before 8:00 and to the current time if it is after 8:00:

\[ \text{If(Now}() < 08:00:00, '08:00:00', \text{String(Now()})} \]

The displayed time refreshes every time the specified time interval period elapses.

See also

If

Year

Number

Description

Converts a string to a number.

Syntax

\[ \text{Number}(\text{string}) \]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>The string you want returned as a number</td>
</tr>
</tbody>
</table>

Return value

A numeric datatype. Returns the contents of \text{string} as a number. If \text{string} is not a valid number, \text{Number} returns 0.

Examples

This expression converts the string 24 to a number:

\[ \text{Number}("24") \]

This expression for a computed field tests whether the value in the age column is greater than 55 and if so displays N/A; otherwise, it displays the value in age:

\[ \text{If(Number}(\text{age}) > 55, "N/A", \text{age}) \]

This validation rule checks that the number the user entered is between 25,000 and 50,000:

\[ \text{Number(GetText())} > 25000 \text{ AND Number(GetText())} < 50000 \]
Page

Description
Gets the number of the current page.

Syntax
Page ( )

Return value
Long. Returns the number of the current page.

Calculating the page count
The vertical size of the paper less the top and bottom margins is used to calculate the page count. When the print orientation is landscape, the vertical size of the paper is the shorter dimension.

Examples
This expression returns the number of the current page:

Page ( )

In the DataWindow object’s footer band, this expression for a computed field displays a string showing the current page number and the total number of pages in the report. The result has the format Page n of total:

'Page ' + Page ( ) + ' of ' + PageCount ( )

See also
PageAbs
PageAcross
PageCount
PageCountAcross

PageAbs

Description
Gets the absolute number of the current page.

Syntax
PageAbs ( )

Return value
Long. Returns the absolute number of the current page.

Usage
Use this function for group reports that have ResetPageCount = yes. It returns the absolute page number, ignoring the page reset count. This enables you to number the grouped pages, but also to obtain the absolute page when the user wants to print the current page, regardless of what that page number is in a grouped page report.

Examples
This expression returns the absolute number of the current page:

PageAbs ( )
This example obtains the absolute page number for the first row on the page in the string variable \textit{ret}:

\begin{verbatim}
string ret, row
ret = dw1.Describe("Evaluate('pageabs()', "+row+")")
\end{verbatim}

\section*{PageAcross}

\textbf{Description} \hspace{2cm} \begin{flushleft}
Gets the number of the current horizontal page. For example, if a report is twice the width of the print preview window and the window is scrolled horizontally to display the portion of the report that was outside the preview, PageAcross returns 2 because the current page is the second horizontal page.
\end{flushleft}

\textbf{Syntax} \hspace{2cm} \texttt{PageAcross ()}

\textbf{Return value} \hspace{2cm} Long. Returns the number of the current horizontal page if it succeeds and -1 if an error occurs.

\textbf{Examples} \hspace{2cm} This expression returns the number of the current horizontal page:

\begin{verbatim}
PageAcross()
\end{verbatim}

\textbf{See also} \hspace{2cm} Page \hspace{2cm} PageCount \hspace{2cm} PageCountAcross

\section*{PageCount}

\textbf{Description} \hspace{2cm} \begin{flushleft}
Gets the total number of pages when a DataWindow object is being viewed in Print Preview. This number is also the number of printed pages if the DataWindow object is not wider than the preview window. If the DataWindow object is wider than the preview window, the number of printed pages will be greater than the number PageCount gets.
\end{flushleft}

\textbf{Syntax} \hspace{2cm} \texttt{PageCount ()}

\textbf{Return value} \hspace{2cm} Long. Returns the total number of pages.
PageCountAcross

Usage
PageCount applies to Print Preview.

Calculating the page count
The vertical size of the paper less the top and bottom margins is used to calculate the page count. When the print orientation is landscape, the vertical size of the paper is the shorter dimension.

Examples
This expression returns the number of pages:

```
PageCount()
```

In the DataWindow object’s footer band, this expression for a computed field displays a string showing the current page number and the total number of pages in the report. The result has the format "Page n of total:

```
'Page ' + Page() + ' of ' + PageCount()
```

See also
Page
PageAcross
PageCountAcross

PageCountAcross

Description
Gets the total number of horizontal pages that are wider than the Print Preview window when a DataWindow object is viewed in Print preview.

Syntax
PageCountAcross ( )

Return value
Long. Returns the total number of horizontal pages if it succeeds and -1 if an error occurs.

Usage
PageCountAcross applies to Print Preview.

Examples
This expression returns the number of horizontal pages in the Print Preview window:

```
PageCountAcross()
```

See also
Page
PageAcross
PageCount
### Percent

**Description**
Gets the percentage that the current value represents of the total of the values in the column.

**Syntax**

```plaintext
Percent ( column { FOR range { DISTINCT { expres1, expres2, ... } } } )
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>column</code></td>
<td>The column for which you want the value of each row expressed as a percentage of the total of the values of the column. <code>Column</code> can be the column name or the column number preceded by a pound sign (#). <code>Column</code> can also be an expression that includes a reference to the column. The datatype of <code>column</code> must be numeric.</td>
</tr>
<tr>
<td><code>FOR range</code></td>
<td>The data to be included in the percentage. For most presentation styles, values for <code>range</code> are:</td>
</tr>
<tr>
<td></td>
<td>• ALL – (Default) The percentage that the current value represents of all rows in <code>column</code>.</td>
</tr>
<tr>
<td></td>
<td>• GROUP n – The percentage that the current value represents of rows in <code>column</code> in the specified group. Specify the keyword GROUP followed by the group number: for example, GROUP 1.</td>
</tr>
<tr>
<td></td>
<td>• PAGE – The percentage that the current value represents of the rows in <code>column</code> on a page.</td>
</tr>
<tr>
<td></td>
<td>For Crosstabs, specify CROSSTAB for <code>range</code>:</td>
</tr>
<tr>
<td></td>
<td>• CROSSTAB – (Crosstabs only) The percentage that the current value represents of all rows in <code>column</code> in the crosstab.</td>
</tr>
<tr>
<td></td>
<td>For Graph objects, specify one of the following:</td>
</tr>
<tr>
<td></td>
<td>• GRAPH – (Graphs only) The percentage that the current value represents of values in <code>column</code> in the range specified for the Rows option.</td>
</tr>
<tr>
<td><code>DISTINCT</code></td>
<td>Causes <code>Percent</code> to consider only the distinct values in <code>column</code> when determining the percentage. For a value of <code>column</code>, the first row found with the value is used and other rows that have the same value are ignored.</td>
</tr>
<tr>
<td><code>expresn</code></td>
<td>One or more expressions that you want to evaluate to determine distinct rows. <code>Expresn</code> can be the name of a column, a function, or an expression.</td>
</tr>
</tbody>
</table>

**Return value**
A numeric datatype (decimal, double, integer, long, or real). Returns the percentage the current row of `column` represents of the total value of the column.
Percent

Usage

Usually you use Percent in a column to display the percentage for each row. You can also use Percent in a header or trailer for a group. In the header, Percent displays the percentage for the first value in the group, and in the trailer, for the last value in the group.

If you specify range, Percent returns the percentage that the current row of column represents relative to the total value of range. For example, if column 5 is salary, Percent(#5 for group 1) is equivalent to salary/(Sum(Salary for group 1)).

If you specify DISTINCT, Percent returns the percent that a distinct value in column represents of the total value of column. If you specify expresn, Percent returns the percent that the value in column represents of the total for column in a row in which the value of expresn is distinct.

Formatting the percent value

The percentage is displayed as a decimal value unless you specify a format for the result. A display format can be part of the computed field’s definition.

For graphs, you do not select the range when you call the function. The range has already been determined by the Rows setting on the Data property page (the Range property), and the aggregation function uses that range. Settings for Rows include the following:

- For the Graph presentation style, Rows is always All.
- For Graph controls, Rows can be All, Page, or Group.

Null values are ignored and are not considered in the calculation.

Not in validation rules, filter expressions, or crosstabs

You cannot use Percent or other aggregate functions in validation rules or filter expressions. Percent does not work for crosstabs; specifying “for crosstab” as a range is not available for Percent.

Using an aggregate function cancels the effect of setting Retrieve Rows As Needed in the painter. To do the aggregation, a DataWindow object always retrieves all rows.

Examples

This expression returns the value of each row in the column named salary as a percentage of the total of salary:

Percent(salary)
This expression returns the value of each row in the column named cost as a percentage of the total of cost in group 2:

\[ \text{Percent}(\text{cost for group 2}) \]

This expression entered in the Value box on the Data tab page in the Graph Object property sheet returns the value of each row in the qty_ordered as a percentage of the total for the column in the graph:

\[ \text{Percent}(\text{qty\_ordered for graph}) \]

Assuming a DataWindow object displays the order number, amount, and line items for each order, this computed field returns the order amount as a percentage of the total order amount for the distinct order numbers:

\[ \text{Percent}(\text{order\_amt for all DISTINCT order\_nbr}) \]

See also CumulativePercent

---

**Pi**

**Description**

Multiplies pi by a specified number.

**Syntax**

\[ \text{Pi}(n) \]

**Argument** | **Description**
--- | ---
\( n \) | The number you want to multiply by pi (3.14159265358979323...)

**Return value**

Double. Returns the result of multiplying \( n \) by pi if it succeeds and -1 if an error occurs.

**Usage**

Use \( \text{Pi} \) to convert angles to and from radians.

**Examples**

This expression returns pi:

\[ \text{Pi}(1) \]

Both these expressions return the area of a circle with the radius Rad:

\[ \text{Pi}(1) \ast \text{Rad}^2 \]
\[ \text{Pi}(\text{Rad}^2) \]

This expression computes the cosine of a 45-degree angle:

\[ \text{Cos}(45.0 \ast (\text{Pi}(2)/360)) \]

See also Cos, Sin, Tan
**Pos**

Description | Finds one string within another string.
---|---
Syntax | `Pos (string1, string2, start)`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>string1</code></td>
<td>The string in which you want to find <code>string2</code>.</td>
</tr>
<tr>
<td><code>string2</code></td>
<td>The string you want to find in <code>string1</code>.</td>
</tr>
<tr>
<td><code>start</code> (optional)</td>
<td>A long indicating where the search will begin in <code>string</code>. The default is 1.</td>
</tr>
</tbody>
</table>

Return value | Long. Returns a long whose value is the starting position of the first occurrence of `string2` in `string1` after the position specified in `start`. If `string2` is not found in `string1` or if `start` is not within `string1`, `Pos` returns 0.

Usage | The `Pos` function is case sensitive.

Examples | This expression returns the position of the letter `a` in the value of the `last_name` column:

```
Pos(last_name, "a")
```

This expression returns 6:

```
Pos("BABE RUTH", "RU")
```

This expression returns 1:

```
Pos("BABE RUTH", "B")
```

This expression returns 0 (because the case does not match):

```
Pos("BABE RUTH", "be")
```

This expression returns 0 (because it starts searching at position 5, after the occurrence of `BE`):

```
Pos("BABE RUTH", "BE", 5)
```

See also | LastPos
| Left
| Mid
| PosA
| Right
CHAPTER 2  DataWindow Expression Functions

**PosA**

**Description**
Finds one string within another string.

**Syntax**
```
PosA ( string1, string2, start )
```

**Argument** | **Description**
---|---
`string1` | The string in which you want to find `string2`.  
`string2` | The string you want to find in `string1`.  
`start` (optional) | A long indicating the position in bytes where the search will begin in `string`. The default is 1.

**Return value**
Long. Returns a long whose value is the starting position of the first occurrence of `string2` in `string1 after` the position in bytes specified in `start`. If `string2` is not found in `string1` or if `start` is not within `string1`, PosA returns 0.

**Usage**
In SBCS environments, Pos and PosA return the same results.

**See also**
LastPos  
LeftA  
MidA  
Pos  
RightA

---

**ProfileInt**

**Description**
Obtains the integer value of a setting in the specified profile file.

**Syntax**
```
ProfileInt ( filename, section, key, default )
```

**Argument** | **Description**
---|---
`filename` | A string whose value is the name of the profile file. If you do not specify a full path, ProfileInt uses the operating system’s standard file search order to find the file.
`section` | A string whose value is the name of a group of related values in the profile file. In the file, section names are in square brackets. Do not include the brackets in `section`. `Section` is not case sensitive.
`key` | A string specifying the setting name in `section` whose value you want. The setting name is followed by an equal sign in the file. Do not include the equal sign in `key`. `Key` is not case sensitive.
`default` | An integer value that ProfileInt returns if `filename` is not found, if `section` or `key` does not exist in `filename`, or if the value of `key` cannot be converted to an integer.
ProfileInt

Return value

Integer. Returns default if filename is not found, section is not found in filename, key is not found in section, or the value of key is not an integer. Returns -1 if an error occurs.

Usage

Use ProfileInt and ProfileString to get configuration settings from a profile file you have designed for your application. ProfileInt and ProfileString can read files with ANSI or UTF16-LE encoding.

Examples

This example uses the following PROFILE.INI file:

```
[MyApp]
Maximized=1

[Security]
Class = 7
```

This expression tries to return the integer value of the keyword Minimized in section MyApp of file C:\PROFILE.INI. It returns 3 if there is no MyApp section or no Minimized keyword in the MyApp section. Based on the sample file above, it returns 3:

```
ProfileInt("C:\PROFILE.INI", "MyApp", "minimized", 3)
```

See also

ProfileString
ProfileString

Description
Obtains the string value of a setting in the specified profile file.

Syntax
`ProfileString ( filename, section, key, default )`

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>filename</td>
<td>A string whose value is the name of the profile file. If you do not specify a full path, <code>ProfileString</code> uses the operating system’s standard file search order to find the file.</td>
</tr>
<tr>
<td>section</td>
<td>A string whose value is the name of a group of related values in the profile file. In the file, section names are in square brackets. Do not include the brackets in <code>section</code>. <code>Section</code> is not case sensitive.</td>
</tr>
<tr>
<td>key</td>
<td>A string specifying the setting name in <code>section</code> whose value you want. The setting name is followed by an equal sign in the file. Do not include the equal sign in <code>key</code>. <code>Key</code> is not case sensitive.</td>
</tr>
<tr>
<td>default</td>
<td>A string value that <code>ProfileString</code> returns if <code>filename</code> is not found, if <code>section</code> or <code>key</code> does not exist in <code>filename</code>, or if the value of <code>key</code> cannot be converted to an integer.</td>
</tr>
</tbody>
</table>

Return value
String, with a maximum length of 4096 characters. Returns the string from `key` within `section` within `filename`. If `filename` is not found, `section` is not found in `filename`, or `key` is not found in `section`, `ProfileString` returns `default`. If an error occurs, it returns the empty string ("").

Usage
Use `ProfileInt` and `ProfileString` to get configuration settings from a profile file you have designed for your application. `ProfileInt` and `ProfileString` can read files with ANSI or UTF16-LE encoding.

Examples
This example uses the following section in the `PROFILE.INI` file:

```
[Employee]
Name="Smith"

[Dept]
Name="Marketing"
```

This expression returns the string for the keyword `Name` in section `Employee` in file `C:\PROFILE.INI`. It returns `None` if the section or keyword does not exist. In this case it returns `Smith`:

```
ProfileString("C:\PROFILE.INI", "Employee", "Name", "None")
```

See also
`ProfileInt`
Rand

Description
Obtains a random whole number between 1 and a specified upper limit.

Syntax
Rand(n)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>The upper limit of the range of random numbers you want returned. The lower limit is always 1. The upper limit cannot exceed 32,767.</td>
</tr>
</tbody>
</table>

Return value
A numeric datatype, the datatype of \( n \). Returns a random whole number between 1 and \( n \).

Usage
The sequence of numbers generated by repeated calls to the Rand function is a computer-generated pseudorandom sequence.

Examples
This expression returns a random whole number between 1 and 10:

\[
\text{Rand}(10)
\]

Real

Description
Converts a string value to a real datatype.

Syntax
Real(string)

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>The string whose value you want to convert to a real</td>
</tr>
</tbody>
</table>

Return value
Real. Returns the contents of a string as a real. If \( \text{string} \) is not a valid number, Real returns 0.

Examples
This expression converts 24 to a real:

\[
\text{Real}("24")
\]

This expression returns the value in the column temp_text as a real:

\[
\text{Real}(\text{temp_text})
\]
RelativeDate
Description
Obtains the date that occurs a specified number of days after or before another
date.
Syntax
\[
\text{RelativeDate} (\ date, \ n )
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>A date value</td>
</tr>
<tr>
<td>n</td>
<td>An integer indicating the number of days</td>
</tr>
</tbody>
</table>

Return value
Date. Returns the date that occurs \( n \) days after \( date \) if \( n \) is greater than 0.
Returns the date that occurs \( n \) days before \( date \) if \( n \) is less than 0.

Examples
This expression returns 2005-02-10:
\[\text{RelativeDate}(2005-01-31, 10)\]
This expression returns 2005-01-21:
\[\text{RelativeDate}(2005-01-31, -10)\]

See also
DaysAfter

RelativeTime
Description
Obtains a time that occurs a specified number of seconds after or before
another time within a 24-hour period.
Syntax
\[
\text{RelativeTime} (\ time, \ n )
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>A time value</td>
</tr>
<tr>
<td>n</td>
<td>A long number of seconds</td>
</tr>
</tbody>
</table>

Return value
Time. Returns the time that occurs \( n \) seconds after \( time \) if \( n \) is greater than 0.
Returns the time that occurs \( n \) seconds before \( time \) if \( n \) is less than 0. The
maximum return value is 23:59:59.

Examples
This expression returns 19:01:41:
\[\text{RelativeTime}(19:01:31, 10)\]
This expression returns 19:01:21:
\[\text{RelativeTime}(19:01:31, -10)\]

See also
SecondsAfter
Replace

Description
Replaces a portion of one string with another.

Syntax
Replace ( string1, start, n, string2 )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string1</td>
<td>The string in which you want to replace characters with string2.</td>
</tr>
<tr>
<td>start</td>
<td>A long whose value is the number of the first character you want replaced. (The first character in the string is number 1.)</td>
</tr>
<tr>
<td>n</td>
<td>A long whose value is the number of characters you want to replace.</td>
</tr>
<tr>
<td>string2</td>
<td>The string that replaces characters in string1. The number of characters in string2 can be greater than, equal to, or fewer than the number of characters you are replacing.</td>
</tr>
</tbody>
</table>

Return value
String. Returns the string with the characters replaced if it succeeds and the empty string ("") if it fails.

Usage
If the start position is beyond the end of the string, Replace appends string2 to string1. If there are fewer characters after the start position than specified in n, Replace replaces all the characters to the right of character start.

If n is zero, then in effect Replace inserts string2 into string1.

Examples
This expression changes the last two characters of the string David to e to make it Dave:

    Replace("David", 4, 2, "e")

This expression returns MY HOUSE:

    Replace("YOUR HOUSE", 1, 4, "MY")

This expression returns Closed for the Winter:

    Replace("Closed for Vacation", 12, 8, "the Winter")
### ReplaceA

**Description**
Replaces a portion of one string with another.

**Syntax**
```
ReplaceA ( string1, start, n, string2 )
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>string1</code></td>
<td>The string in which you want to replace bytes with <code>string2</code>.</td>
</tr>
<tr>
<td><code>start</code></td>
<td>A long whose value is the number of the first byte you want replaced. (The first byte in the string is number 1.)</td>
</tr>
<tr>
<td><code>n</code></td>
<td>A long whose value is the number of bytes you want to replace.</td>
</tr>
<tr>
<td><code>string2</code></td>
<td>The string that replaces bytes in <code>string1</code>. The number of bytes in <code>string2</code> can be greater than, equal to, or fewer than the number of bytes you are replacing.</td>
</tr>
</tbody>
</table>

**Return value**
String. Returns the string with the bytes replaced if it succeeds and the empty string (""") if it fails.

**Usage**
If the start position is beyond the end of the string, `ReplaceA` appends `string2` to `string1`. If there are fewer bytes after the start position than specified in `n`, `ReplaceA` replaces all the bytes to the right of character `start`.

If `n` is zero, then in effect `ReplaceA` inserts `string2` into `string1`.

In SBCS environments, `Replace` and `ReplaceA` return the same results.

**See also**
Replace

### RGB

**Description**
Calculates the long value that represents the color specified by numeric values for the red, green, and blue components of the color.

**Syntax**
```
RGB ( red, green, blue )
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>red</code></td>
<td>The integer value of the red component of the color</td>
</tr>
<tr>
<td><code>green</code></td>
<td>The integer value of the green component of the color</td>
</tr>
<tr>
<td><code>blue</code></td>
<td>The integer value of the blue component of the color</td>
</tr>
</tbody>
</table>

**Return value**
Long. Returns the long that represents the color created by combining the values specified in red, green, and blue. If an error occurs, `RGB` returns null.
Usage

The formula for combining the colors is:

\[
\text{Red} + (256 \times \text{Green}) + (65536 \times \text{Blue})
\]

Use RGB to obtain the long value required to set the color for text and drawing objects. You can also set an object’s color to the long value that represents the color. The RGB function provides an easy way to calculate that value.

**Determining color components**  The value of a component color is an integer between 0 and 255 that represents the amount of the component that is required to create the color you want. The lower the value, the darker the color; the higher the value, the lighter the color.

The following table lists red, green, and blue values for the 16 standard colors:

<table>
<thead>
<tr>
<th>Color</th>
<th>Red value</th>
<th>Green value</th>
<th>Blue value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Black</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>White</td>
<td>255</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>Light Gray</td>
<td>192</td>
<td>192</td>
<td>192</td>
</tr>
<tr>
<td>Dark Gray</td>
<td>128</td>
<td>128</td>
<td>128</td>
</tr>
<tr>
<td>Red</td>
<td>255</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Dark Red</td>
<td>128</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Green</td>
<td>0</td>
<td>255</td>
<td>0</td>
</tr>
<tr>
<td>Dark Green</td>
<td>0</td>
<td>128</td>
<td>0</td>
</tr>
<tr>
<td>Blue</td>
<td>0</td>
<td>0</td>
<td>255</td>
</tr>
<tr>
<td>Dark Blue</td>
<td>0</td>
<td>0</td>
<td>128</td>
</tr>
<tr>
<td>Magenta</td>
<td>255</td>
<td>0</td>
<td>255</td>
</tr>
<tr>
<td>Dark Magenta</td>
<td>128</td>
<td>0</td>
<td>128</td>
</tr>
<tr>
<td>Cyan</td>
<td>0</td>
<td>255</td>
<td>255</td>
</tr>
<tr>
<td>Dark Cyan</td>
<td>0</td>
<td>128</td>
<td>128</td>
</tr>
<tr>
<td>Yellow</td>
<td>255</td>
<td>255</td>
<td>0</td>
</tr>
<tr>
<td>Brown</td>
<td>128</td>
<td>128</td>
<td>0</td>
</tr>
</tbody>
</table>

**Examples**

This expression returns as a long 8421376, which represents dark cyan:

\[
\text{RGB}(0,128,128)
\]

This expression for the Background.Color property of a salary column returns a long that represents red if an employee’s salary is greater than $50,000 and white if salary is less than or equal to $50,000:

\[
\text{If(salary>50000, RGB(255,0,0), RGB(255,255,255))}
\]

See also

“Example 3: creating a row indicator” on page 19
**Right**

Description: Obtains a specified number of characters from the end of a string.

Syntax:  
```plaintext
Right ( string, n )
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>string</code></td>
<td>The string from which you want characters returned</td>
</tr>
<tr>
<td><code>n</code></td>
<td>A long whose value is the number of characters you want returned from the right end of <code>string</code></td>
</tr>
</tbody>
</table>

Return value: String. Returns the rightmost `n` characters in `string` if it succeeds and the empty string ("") if an error occurs.

If `n` is greater than or equal to the length of the string, `Right` returns the entire string. It does not add spaces to make the return value’s length equal to `n`.

Examples:

This expression returns HILL:
```plaintext
Right ("CASTLE HILL", 4)
```

This expression returns CASTLE HILL:
```plaintext
Right ("CASTLE HILL", 75)
```

See also: Left, Mid, Pos

---

**RightA**

Description: Obtains a specified number of characters from the end of a string.

Syntax:  
```plaintext
Right ( string, n )
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>string</code></td>
<td>The string from which you want characters returned</td>
</tr>
<tr>
<td><code>n</code></td>
<td>A long whose value is the number of characters you want returned from the right end of <code>string</code></td>
</tr>
</tbody>
</table>

Return value: String. Returns the rightmost `n` characters in `string` if it succeeds and the empty string ("") if an error occurs.

If `n` is greater than or equal to the length of the string, `RightA` returns the entire string. It does not add spaces to make the return value’s length equal to `n`.
RightTrim

Usage
In SBCS environments, Right and RightA return the same results.

See also
LeftA
MidA
PosA
Right

RightTrim

Description
Removes spaces from the end of a string.

Syntax
RightTrim ( string )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>The string you want returned with trailing blanks deleted</td>
</tr>
</tbody>
</table>

Return value
String. Returns a copy of string with trailing blanks deleted if it succeeds and
the empty string (""") if an error occurs.

Examples
This expression returns RUTH:

RightTrim("RUTH ")

See also
LeftTrim
Trim

Round

Description
Rounds a number to the specified number of decimal places.

Syntax
Round ( x , n )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>The number you want to round.</td>
</tr>
<tr>
<td>n</td>
<td>The number of decimal places to which you want to round x. Valid values are 0 through 28.</td>
</tr>
</tbody>
</table>

Return value
Decimal. If n is positive, Round returns x rounded to the specified number of
decimal places. If n is negative, it returns x rounded to (- n +1) places before
the decimal point. Returns -1 if it fails.
### DataWindow Expression Functions

#### Examples

This expression returns 9.62:

\[
\text{Round}(9.624, 2)
\]

This expression returns 9.63:

\[
\text{Round}(9.625, 2)
\]

This expression returns 9.600:

\[
\text{Round}(9.6, 3)
\]

This expression returns -9.63:

\[
\text{Round}(-9.625, 2)
\]

This expression returns -10:

\[
\text{Round}(-9.625, -1)
\]

**See also**
- Ceiling
- Int
- Truncate

#### RowCount

**Description**

Obtains the number of rows that are currently available in the primary buffer.

**Syntax**

\[
\text{RowCount}()
\]

**Return value**

Long. Returns the number of rows that are currently available, 0 if no rows are currently available, and -1 if an error occurs.

**Examples**

This expression in a computed field returns a warning if no data exists and the number of rows if there is data:

\[
\text{If(RowCount}() = 0, "No Data", \text{String(RowCount}()))
\]

#### RowHeight

**Description**

Reports the height of a row associated with a band in a DataWindow object.

**Syntax**

\[
\text{RowHeight}()
\]

**Return value**

Long. Returns the height of the row in the units specified for the DataWindow object if it succeeds, and -1 if an error occurs.
Second

Usage
When you call RowHeight in a band other than the detail band, it reports on a row in the detail band. See GetRow for a table specifying which row is associated with each band for reporting purposes.

When a band has Autosize Height set to true, you should avoid using the RowHeight DataWindow expression function to set the height of any element in the row. Doing so can result in a logical inconsistency between the height of the row and the height of the element. If you need to use RowHeight, you must set the Y coordinate of the element to 0 on the Position page in the Properties view, otherwise the bottom of the element might be clipped. You must do this for every element that uses such an expression. If you move any elements in the band, make sure that their Y coordinates are still set to 0.

You should not use an expression whose runtime value is greater than the value returned by RowHeight. For example, you should not set the height of a column to rowheight() + 30. Such an expression produces unpredictable results at runtime.

Examples
This expression for a computed field in the detail band displays the height of each row:

RowHeight()

See also
GetRow

Second

Description
Obtains the number of seconds in the seconds portion of a time value.

Syntax
Second( time )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>time</td>
<td>The time value from which you want the seconds</td>
</tr>
</tbody>
</table>

Return value
Integer. Returns the seconds portion of time (00 to 59).

Examples
This expression returns 31:

Second(19:01:31)

See also
Hour
Minute
CHAPTER 2  DataWindow Expression Functions

SecondsAfter

Description
Gets the number of seconds one time occurs after another.

Syntax
SecondsAfter ( time1, time2 )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>time1</td>
<td>A time value that is the start time of the interval being measured</td>
</tr>
<tr>
<td>time2</td>
<td>A time value that is the end time of the interval</td>
</tr>
</tbody>
</table>

Return value
Long. Returns the number of seconds time2 occurs after time1. If time2 occurs before time1, SecondsAfter returns a negative number.

Examples
This expression returns 15:

SecondsAfter(21:15:30, 21:15:45)

This expression returns -15:

SecondsAfter(21:15:45, 21:15:30)

This expression returns 0:

SecondsAfter(21:15:45, 21:15:45)

See also
DaysAfter

Sign

Description
Reports whether the number is negative, zero, or positive by checking its sign.

Syntax
Sign ( n )

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>n</td>
<td>The number for which you want to determine the sign</td>
</tr>
</tbody>
</table>

Return value
Integer. Returns a number (-1, 0, or 1) indicating the sign of n.

Examples
This expression returns 1 (the number is positive):

Sign(5)

This expression returns 0:

Sign(0)

This expression returns -1 (the number is negative):

Sign(-5)
**Sin**

**Description**
Calculates the sine of an angle.

**Syntax**
\[
\text{Sin}\ (n)
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( n )</td>
<td>The angle (in radians) for which you want the sine</td>
</tr>
</tbody>
</table>

**Return value**
Double. Returns the sine of \( n \) if it succeeds and -1 if an error occurs.

**Examples**
This expression returns .841470984078965:
\[
\text{Sin}(1)
\]
This expression returns 0:
\[
\text{Sin}(0)
\]
This expression returns 0:
\[
\text{Sin}(\pi(1))
\]

**See also**
Cos
Pi
Tan

---

**Small**

**Description**
Finds a small value at a specified ranking in a column (for example, third-smallest, fifth-smallest) and returns the value of another column or expression based on the result.

**Syntax**
\[
\text{Small}\ ( returnexp, column, nbottom } \{ \text{ FOR range } \{ \text{ DISTINCT } \{ \text{ expres1 } \{, \text{ expres2 } \{, ... \} \} \} \} \}
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>( returnexp )</td>
<td>The value you want returned when the small value is found. ( Returnexp ) includes a reference to a column, but not necessarily the column that is being evaluated for the small value, so that a value is returned from the same row that contains the small value.</td>
</tr>
<tr>
<td>( column )</td>
<td>The column that contains the small value you are searching for. ( Column ) can be a column name or a column number preceded by a pound sign (#). ( Column ) can also be an expression that includes a reference to the column. The datatype of ( column ) must be numeric.</td>
</tr>
</tbody>
</table>
## DataWindow Expression Functions

### CHAPTER 2  DataWindow Expression Functions

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>nbottom</code></td>
<td>The relationship of the small value to the column’s smallest value. For example, when <code>nbottom</code> is 2, Small finds the second-smallest value.</td>
</tr>
<tr>
<td><code>range</code> (optional)</td>
<td>The data that will be included when finding the small value. For most presentation styles, values for <code>range</code> are:</td>
</tr>
<tr>
<td></td>
<td>• ALL – (Default) The small value of all rows in column.</td>
</tr>
<tr>
<td></td>
<td>• GROUP n – The small value of rows in column in the specified group. Specify the keyword GROUP followed by the group number: for example, GROUP 1.</td>
</tr>
<tr>
<td></td>
<td>• PAGE – The small value of the rows in column on a page. For Crosstabs, specify CROSSTAB for <code>range</code>:</td>
</tr>
<tr>
<td></td>
<td>• CROSSTAB – (Crosstabs only) The small value of all rows in column in the crosstab. For Graph objects, specify the following:</td>
</tr>
<tr>
<td></td>
<td>• GRAPH – (Graphs only) The small value in column in the range specified for the Rows option.</td>
</tr>
<tr>
<td><code>DISTINCT</code> (optional)</td>
<td>Causes Small to consider only the distinct values in column when determining the small value. For a value of column, the first row found with the value is used and other rows that have the same value are ignored.</td>
</tr>
<tr>
<td><code>expresn</code> (optional)</td>
<td>One or more expressions that you want to evaluate to determine distinct rows. <code>Expresn</code> can be the name of a column, a function, or an expression.</td>
</tr>
</tbody>
</table>

#### Return value

The datatype of `returnexp`. Returns the `nbottom`-smallest value if it succeeds and -1 if an error occurs.

#### Usage

If you specify `range`, Small returns the value in `returnexp` when the value in `column` is the `nbottom`-smallest value in `range`. If you specify `DISTINCT`, Small returns `returnexp` when the value in `column` is the `nbottom`-smallest value of the distinct values in `column`, or if you specify `expresn`, the `nbottom`-smallest for each distinct value of `expresn`.

For graphs, you do not select the range when you call the function. The range has already been determined by the Rows setting on the Data property page (the Range property), and the aggregation function uses that range.

Settings for Rows include the following:

- For the Graph presentation style, Rows is always All.
- For Graph controls, Rows can be All, Page, or Group.
**Min might be faster**  If you do not need a return value from another column and you want to find the smallest value \( n_{bottom} = 1 \), use Min; it is faster.

**Not in validation rules or filter expressions**  You cannot use this or other aggregate functions in validation rules or filter expressions.

Using an aggregate function cancels the effect of setting Retrieve Rows As Needed in the painter. To do the aggregation, a DataWindow object always retrieves all rows.

**Examples**

These expressions return the names of the salespersons with the three smallest sales (sum_sales is the sum of the sales for each salesperson) in group 2, which might be the salesregion group. Note that sum_sales contains the values being compared, but Small returns a value in the name column:

- `Small(name, sum_sales, 1 for group 2)`
- `Small(name, sum_sales, 2 for group 2)`
- `Small(name, sum_sales, 3 for group 2)`

This example reports the salesperson with the third-smallest sales, considering only the first entry for each salesperson:

- `Small(name, sum_sales, 3 for all DISTINCT sum_sales)`

**See also**  Large

---

**Space**

**Description**
Builds a string of the specified length whose value consists of spaces.

**Syntax**

```
Space(n)
```

**Argument** | **Description**
--- | ---
\( n \) | A long whose value is the length of the string you want filled with spaces

**Return value**
String. Returns a string filled with \( n \) spaces if it succeeds and the empty string ("") if an error occurs.

**Examples**

This expression for a computed field returns 10 spaces in the computed field if the value of the rating column is Top Secret; otherwise, it returns the value in rating:

```
If(rating = "Top Secret", Space(10), rating)
```
**Sqrt**

Description: Calculates the square root of a number.

Syntax: 

\[
\text{Sqrt}(\ n) \\
\]

- **Argument**
  - \( n \)
- **Description**
  - The number for which you want the square root

Return value: Double. Returns the square root of \( n \).

Usage: \( \text{Sqrt}(\ n) \) is the same as \( n^{.5} \).

Taking the square root of a negative number causes an execution error.

Examples:

This expression returns 1.414213562373095:

\[
\text{Sqrt}(2) \\
\]

This expression results in an error at execution time:

\[
\text{Sqrt}(-2) \\
\]
StDev

Description

Calculates an estimate of the standard deviation for the specified column.
Standard deviation is a measurement of how widely values vary from average.

Syntax

\[
\text{StDev}(\text{column} \{ \text{FOR range} \{ \text{DISTINCT} \{ \text{expresn1}, \text{expresn2}, \ldots \} \} \})
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>column</td>
<td>The column for which you want an estimate for the standard deviation of the values in the rows. Column can be the column name or the column number preceded by a pound sign (#). Column can also be an expression that includes a reference to the column. The datatype of column must be numeric.</td>
</tr>
<tr>
<td>FOR range</td>
<td>The data to be included in the estimate of the standard deviation. For most presentation styles, values for range are: • ALL – (Default) The estimate of the standard deviation for all values in column. • GROUP n – The estimate of the standard deviation for values in column in the specified group. Specify the keyword GROUP followed by the group number: for example, GROUP 1. • PAGE – The estimate of the standard deviation for the values in column on a page. For Crosstabs, specify CROSSTAB for range to indicate the standard deviation for all values in column in the crosstab. For Graph objects specify GRAPH to indicate the standard deviation for values in column in the range specified for the Rows option.</td>
</tr>
<tr>
<td>DISTINCT</td>
<td>Causes StDev to consider only the distinct values in column when determining the standard deviation. For a value of column, the first row found with the value is used and other rows that have the same value are ignored.</td>
</tr>
<tr>
<td>expresn</td>
<td>One or more expressions that you want to evaluate to determine distinct rows. Expresn can be the name of a column, a function, or an expression.</td>
</tr>
</tbody>
</table>

Return value

Double. Returns an estimate of the standard deviation for column.

Usage

If you specify range, StDev returns an estimate for the standard deviation of column within range. If you specify DISTINCT, StDev returns an estimate of the standard deviation for the distinct values in column, or if you specify expresn, the estimate of the standard deviation of the rows in column where the value of expresn is distinct.
For graphs, you do not select the range when you call the function. The range has already been determined by the Rows setting on the Data tab page (the Range property), and the aggregation function uses that range. Settings for Rows include the following:

- For the Graph presentation style, Rows is always All.
- For Graph controls, Rows can be All, Page, or Group.

**Estimating or calculating actual standard deviation**  
StDev assumes that the values in column are a sample of the values in the rows in the column in the database table. If you selected all the rows in the column in the DataWindow object’s SELECT statement, use StDevP to compute the standard deviation of the population.

**Not in validation rules or filter expressions**  
You cannot use this or other aggregate functions in validation rules or filter expressions.

Using an aggregate function cancels the effect of setting Retrieve Rows As Needed in the painter. To do the aggregation, a DataWindow object always retrieves all rows.

**Examples**

These examples all assume that the SELECT statement did not retrieve all the rows in the database table. StDev is intended to work with a subset of rows, which is a sample of the full set of data.

This expression returns an estimate for standard deviation of the values in the column named salary:

\[
\text{StDev(salary)}
\]

This expression returns an estimate for standard deviation of the values in the column named salary in group 1:

\[
\text{StDev(salary for group 1)}
\]

This expression returns an estimate for standard deviation of the values in column 4 on the page:

\[
\text{StDev(#4 for page)}
\]

This expression entered in the Value box on the Data tab page in the graph’s property sheet returns an estimate for standard deviation of the values in the qty_used column in the graph:

\[
\text{StDev(qty_used for graph)}
\]
This expression for a computed field in a crosstab returns the estimate for standard deviation of the values in the qty_ordered column in the crosstab:

\[ \text{StDev(qty\_ordered for crosstab)} \]

Assuming a DataWindow object displays the order number, amount, and line items for each order, this computed field returns the estimated standard deviation of the order amount for the distinct order numbers:

\[ \text{StDev(order\_amt for all DISTINCT order\_nbr)} \]

**See also**

StDevP
Var

**StDevP**

**Description**

Calculates the standard deviation for the specified column. Standard deviation is a measurement of how widely values vary from average.

**Syntax**

\[
\text{StDevP( column \{ FOR range \{ DISTINCT \{ expres1 \{, expres2 \{, ... } \}} \} \}} \]
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>column</td>
<td>The column for which you want the standard deviation of the values in the rows. Column can be the column name or the column number preceded by a pound sign (#). Column can also be an expression that includes a reference to the column. The datatype of column must be numeric.</td>
</tr>
<tr>
<td>FOR range</td>
<td>The data to be included in the standard deviation. For most presentation styles, values for range are:</td>
</tr>
<tr>
<td>(optional)</td>
<td>• ALL – (Default) The standard deviation for all values in column.</td>
</tr>
<tr>
<td></td>
<td>• GROUP n – The standard deviation for values in column in the specified group. Specify the keyword GROUP followed by the group number: for example, GROUP 1.</td>
</tr>
<tr>
<td></td>
<td>• PAGE – The standard deviation for the values in column on a page.</td>
</tr>
<tr>
<td></td>
<td>For Crosstabs, specify CROSSTAB for range to indicate the standard deviation for all values in column in the crosstab.</td>
</tr>
<tr>
<td></td>
<td>For Graph objects specify GRAPH to indicate the standard deviation for values in column in the range specified for the Rows option.</td>
</tr>
</tbody>
</table>
## CHAPTER 2 DataWindow Expression Functions

### DataWindow Object Reference

**Return value**

Double. Returns the standard deviation for `column`.

**Usage**

If you specify `range`, `StDevP` returns the standard deviation for `column` within `range`. If you specify `DISTINCT`, `StDevP` returns an estimate of the standard deviation for the distinct values in `column`, or if you specify `expresn`, the estimate of the standard deviation of the rows in `column` where the value of `expresn` is distinct.

For graphs, you do not select the range when you call the function. The range has already been determined by the Rows setting on the Data tab page (the Range property), and the aggregation function uses that range. Settings for Rows include the following:

- For the Graph presentation style, Rows is always All.
- For Graph controls, Rows can be All, Page, or Group.

### Estimating or calculating actual standard deviation

`StDevP` assumes that the values in `column` are the values in all the rows in the column in the database table. If you did not select all rows in the column in the `SELECT` statement, use `StDev` to compute an estimate of the standard deviation of a sample.

### Not in validation rules or filter expressions

You cannot use this or other aggregate functions in validation rules or filter expressions.

Using an aggregate function cancels the effect of setting Retrieve Rows As Needed in the painter. To do the aggregation, a DataWindow object always retrieves all rows.

### Examples

These examples all assume that the `SELECT` statement retrieved all rows in the database table. `StDevP` is intended to work with a full set of data, not a subset.

This expression returns the standard deviation of the values in the column named `salary`:

\[
\text{StDevP}(\text{salary})
\]
This expression returns the standard deviation of the values in group 1 in the column named salary:

$$\text{StDevP}(\text{salary for group 1})$$

This expression returns the standard deviation of the values in column 4 on the page:

$$\text{StDevP}(\#4 \text{ for page})$$

This expression entered in the Value box on the Data tab page in the graph’s property sheet returns the standard deviation of the values in the qty_ordered column in the graph:

$$\text{StDevP}(\text{qty\_ordered for graph})$$

This expression for a computed field in a crosstab returns the standard deviation of the values in the qty_ordered column in the crosstab:

$$\text{StDevP}(\text{qty\_ordered for crosstab})$$

Assuming a DataWindow object displays the order number, amount, and line items for each order, this computed field returns the standard deviation of the order amount for the distinct order numbers:

$$\text{StDevP}(\text{order\_amt for all DISTINCT order\_nbr})$$

See also

StDev
VarP
String

Formats data as a string according to a specified display format mask. You can convert and format date, DateTime, numeric, and time data. You can also apply a display format to a string.

Syntax

```
String ( data [, format ] )
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>data</td>
<td>The data you want returned as a string with the specified formatting.</td>
</tr>
<tr>
<td></td>
<td><em>Data</em> can have a date, DateTime, numeric, time, or string datatype.</td>
</tr>
<tr>
<td>format</td>
<td>A string of the display masks you want to use to format the data.</td>
</tr>
<tr>
<td>(optional)</td>
<td>The masks consist of formatting information specific to the datatype of <em>data</em>. If <em>data</em> is type string, <em>format</em> is required. The format string can consist of more than one mask, depending on the datatype of <em>data</em>. Each mask is separated by a semicolon. See Usage for details on each datatype.</td>
</tr>
</tbody>
</table>

Return value

String. Returns `data` in the specified format if it succeeds and the empty string (“””) if the datatype of `data` does not match the type of display mask specified or `format` is not a valid mask.

Usage

For date, DateTime, numeric, and time data, the system’s default format is used for the returned string if you do not specify a format. For numeric data, the default format is the [General] format.

For string data, a display format mask is required. (Otherwise, the function would have nothing to do.)

The format can consist of one or more masks:

- Formats for date, DateTime, string, and time data can include one or two masks. The first mask is the format for the data; the second mask is the format for a null value.
- Formats for numeric data can have up to four masks. A format with a single mask handles both positive and negative data. If there are additional masks, the first mask is for positive values, and the additional masks are for negative, zero, and null values.

A format can include color specifications.

If the display format does not match the datatype, the attempt to apply the mask produces unpredictable results.

For information on specifying display formats, see the *User's Guide*. 

---

DataWindow Object Reference

129
Examples

This expression returns Jan 31, 2005:

```csharp
String(2005-01-31, "mmm dd, yyyy")
```

This expression returns Jan 31, 2005 6 hrs and 8 min:

```csharp
String(2005-01-31 06:08:00, "mmm dd, yyyy h "hrs and" m "min")
```

This expression:

```csharp
String(nbr, "0000;(000);****;empty")
```

returns:

0123 if `nbr` is 123
(123) if `nbr` is -123
**** if `nbr` is 0
empty if `nbr` is null

This expression returns A-B-C:

```csharp
String("ABC", "@-@-@")
```

This expression returns A*B:

```csharp
String("ABC", "@*@")
```

This expression returns ABC:

```csharp
String("ABC", "@@@")
```

This expression returns a space:

```csharp
String("ABC", " ")
```

This expression returns 6 hrs and 8 min:

```csharp
String(06:08:02, 'h "hrs and" m "min")
```

This expression returns 08:06:04 pm:

```csharp
String(20:06:04, "hh:mm:ss am/pm")
```

This expression returns 8:06:04 am:

```csharp
String(08:06:04, "h:mm:ss am/pm")
```

This expression returns 6:11:25.300000:

```csharp
String(6:11:25.30000, "h:mm:ss.ffffff")
```
CHAPTER 2  DataWindow Expression Functions

Sum

Description
Calculates the sum of the values in the specified column.

Syntax
Sum ( column { FOR range { DISTINCT { expres1 {, expres2 {, ...} } } } } )

Argument  Description
---  -----------------------------------------------------------------------------------------
column   The column for which you want the sum of the data values. Column can be the column name or the column number preceded by a pound sign (#). Column can also be an expression that includes a reference to the column. The datatype of column must be numeric.

FOR range (optional)  The data to be included in the sum. For most presentation styles, values for range are:

- ALL – (Default) The sum of all values in column.
- GROUP n – The sum of values in column in the specified group. Specify the keyword GROUP followed by the group number: for example, GROUP 1.
- PAGE – The sum of the values in column on a page.
For Crosstabs, specify CROSSTAB for range:
- CROSSTAB – (Crosstabs only) The sum of all values in column in the crosstab.
For Graph objects, specify the following:
- GRAPH – (Graphs only) The sum of values in column in the range specified for the Rows option of the graph.

DISTINCT (optional)  Causes Sum to consider only the distinct values in column when determining the sum. For a value of column, the first row found with the value is used and other rows that have the same value are ignored.

expresn (optional)  One or more expressions that you want to evaluate to determine distinct rows. Expresn can be the name of a column, a function, or an expression.

Return value
The appropriate numeric datatype. Returns the sum of the data values in column.

Usage
If you specify range, Sum returns the sum of the values in column within range. If you specify DISTINCT, Sum returns the sum of the distinct values in column, or if you specify expresn, the sum of the values of column where the value of expresn is distinct.
For graphs, you do not select the range when you call the function. The range has already been determined by the Rows setting on the Data property page (the Range property), and the aggregation function uses that range. Settings for Rows include the following:

- For the Graph presentation style, Rows is always All.
- For Graph controls, Rows can be All, Page, or Group.

Null values are ignored and are not included in the calculation.

**Not in validation rules or filter expressions**

You cannot use this or other aggregate functions in validation rules or filter expressions.

Using an aggregate function cancels the effect of setting Retrieve Rows As Needed in the painter. To do the aggregation, a DataWindow object always retrieves all rows.

**Examples**

This expression returns the sum of the values in group 1 in the column named salary:

\[
\text{Sum} \text{(salary for group 1)}
\]

This expression returns the sum of the values in column 4 on the page:

\[
\text{Sum} \text{(#4 for page)}
\]

Assuming a DataWindow object displays the order number, amount, and line items for each order, this computed field returns the sum of the order amount for the distinct order numbers:

\[
\text{Sum} \text{(order_amt for all DISTINCT order_nbr)}
\]

**See also**

“Example 1: counting null values in a column” on page 14
“Example 2: counting male and female employees” on page 16
## Tan

**Description**
Calculates the tangent of an angle.

**Syntax**

```plaintext
Tan(n)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>n</code></td>
<td>The angle (in radians) for which you want the tangent</td>
</tr>
</tbody>
</table>

**Return value**
Double. Returns the tangent of `n` if it succeeds and -1 if an error occurs.

**Examples**
Both these expressions return 0:

```
Tan(0)
Tan(Pi(1))
```

This expression returns 1.55741:

```
Tan(1)
```

**See also**
Cos
Pi
Sin

## Time

**Description**
Converts a string to a time datatype.

**Syntax**

```plaintext
Time(string)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>string</code></td>
<td>A string containing a valid time (such as 8 AM or 10:25) that you want returned as a time datatype. Only the hour is required; you do not have to include the minutes, seconds, or microseconds of the time or AM or PM. The default value for minutes and seconds is 00 and for microseconds is 000000. AM or PM is determined automatically.</td>
</tr>
</tbody>
</table>

**Return value**
Time. Returns the time in `string` as a time datatype. If `string` does not contain a valid time, `Time` returns 00:00:00.

**Examples**
This expression returns the time datatype for 45 seconds before midnight (23:59:15):

```
Time("23:59:15")
```
This expression for a computed field returns the value in the time_received column as a value of type time if time_received is not the empty string. Otherwise, it returns 00:00:00:

```
If(time_received = "" ,00:00:00,
    Time(time_received))
```

This example is similar to the previous one, except that it returns 00:00:00 if time_received contains a null value:

```
If(IsNull(time_received), 00:00:00,
    Time(time_received))
```

### Today

**Description**
Obtains the system date and time.

**Syntax**

```
Today()
```

**Return value**
DateTime. Returns the current system date and time.

**Usage**
To display both the date and the time, a computed field must have a display format that includes the time.

**Examples**

This expression for a computed field displays the date and time when the display format for the field is "mm/dd/yy hh:mm":

```
Today()
```

**See also**
Now

### Trim

**Description**
Removes leading and trailing spaces from a string.

**Syntax**

```
Trim(string)
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>The string you want returned with leading and trailing spaces deleted</td>
</tr>
</tbody>
</table>

**Return value**
String. Returns a copy of string with all leading and trailing spaces deleted if it succeeds and the empty string (""") if an error occurs.

**Usage**
Trim is useful for removing spaces that a user might have typed before or after newly entered data.
**Examples**

This expression returns BABE RUTH:

```
Trim(" BABE RUTH ")
```

**See also**

LeftTrim
RightTrim

---

**Truncate**

**Description**

Truncates a number to the specified number of decimal places.

**Syntax**

```
Truncate( x, n )
```

**Argument** | **Description**
--- | ---
`x` | The number you want to truncate.
`n` | The number of decimal places to which you want to truncate `x`. Valid values are 0 through 28.

**Return value**

The datatype of `x`. If `n` is positive, returns `x` truncated to the specified number of decimal places. If `n` is negative, returns `x` truncated to `-n +1` places before the decimal point. Returns -1 if it fails.

**Examples**

This expression returns 9.2:

```
Truncate(9.22, 1)
```

This expression returns 9.2:

```
Truncate(9.28, 1)
```

This expression returns 9:

```
Truncate(9.9, 0)
```

This expression returns -9.2:

```
Truncate(-9.29, 1)
```

This expression returns 0:

```
Truncate(9.2, -1)
```

This expression returns 50:

```
Truncate(54, -1)
```

**See also**

Ceiling
Int
Round
Upper

Description:
Converting all characters in a string to uppercase letters.

Syntax:
Upper (string)

Return value:
String. Returns string with lowercase letters changed to uppercase if it succeeds and the empty string ("") if an error occurs.

Examples:
This expression returns BABE RUTH:
Upper ("Babe Ruth")

See also:
Lower

Var

Description:
Calculates an estimate of the variance for the specified column. The variance is the square of the standard deviation.

Syntax:
Var (column { FOR range { DISTINCT { expres1 {, expres2 {, ... } } } } } )

Arguments:
<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>column</td>
<td>The column for which you want an estimate for the variance of the values in the rows. Column can be the column name or the column number preceded by a pound sign (#). Column can also be an expression that includes a reference to the column. The datatype of column must be numeric.</td>
</tr>
</tbody>
</table>
CHAPTER 2 DataWindow Expression Functions

Return value
Double or decimal if the arguments are decimal. Returns an estimate for the variance for \textit{column}. If you specify \textit{group}, \textit{Var} returns an estimate for the variance for \textit{column} within \textit{group}.

Usage
If you specify \textit{range}, \textit{Var} returns an estimate for the variance for \textit{column} within \textit{range}. If you specify \textit{DISTINCT}, \textit{Var} returns the variance for the distinct values in \textit{column}, or if you specify \textit{expresn}, the estimate for the variance of the rows in \textit{column} where the value of \textit{expresn} is distinct.

For graphs, you do not select the range when you call the function. The range has already been determined by the Rows setting on the Data property page (the Range property), and the aggregation function uses that range.

Settings for Rows include the following:

- For the Graph presentation style, Rows is always All.
- For Graph controls, Rows can be All, Page, or Group.
**Estimating variance or calculating actual variance**

Var assumes that the values in column are a sample of the values in rows in the column in the database table. If you select all rows in the column in the SELECT statement, use VarP to compute the variance of a population.

**Not in validation rules or filter expressions**

You cannot use this or other aggregate functions in validation rules or filter expressions.

Using an aggregate function cancels the effect of setting Retrieve Rows As Needed in the painter. To do the aggregation, a DataWindow object always retrieves all rows.

These examples all assume that the SELECT statement did not retrieve all of the rows in the database table. Var is intended to work with a subset of rows, which is a sample of the full set of data.

This expression returns an estimate for the variance of the values in the column named salary:

\[
\text{Var}(\text{salary})
\]

This expression returns an estimate for the variance of the values in the column named salary in group 1:

\[
\text{Var}(\text{salary for group 1})
\]

This expression entered in the Value box on the Data property page in the graph’s property sheet returns an estimate for the variance of the values in the quantity column in the graph:

\[
\text{Var}(\text{quantity for graph})
\]

This expression for a computed field in a crosstab returns an estimate for the variance of the values in the quantity column in the crosstab:

\[
\text{Var}(\text{quantity for crosstab})
\]

Assuming a DataWindow object displays the order number, amount, and line items for each order, this computed field returns the estimate for the variance of the order amount for the distinct order numbers:

\[
\text{Var}(\text{order_amt for all DISTINCT order_nbr})
\]

See also

StDev
VarP
VarP

Description
Calculates the variance for the specified column. The variance is the square of the standard deviation.

Syntax
\[ \text{VarP (} \text{column} \{ \text{FOR } \text{range} \{ \text{DISTINCT }\{ \text{expresn}_1 \{, \text{expresn}_2 \{, \ldots \}\}\}\}\}\)\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{column}</td>
<td>The column for which you want the variance of the values in the rows. \text{Column} can be the column name or the column number preceded by a pound sign (#). \text{Column} can also be an expression that includes a reference to the column. The datatype of \text{column} must be numeric.</td>
</tr>
<tr>
<td>\text{FOR range} (optional)</td>
<td>The data that will be included in the variance. For most presentation styles, values for \text{range} are:</td>
</tr>
<tr>
<td>• ALL – (Default) The variance for all rows in \text{column}.</td>
<td></td>
</tr>
<tr>
<td>• GROUP \text{n} – The variance for rows in \text{column} in the specified group. Specify the keyword GROUP followed by the group number: for example, GROUP 1.</td>
<td></td>
</tr>
<tr>
<td>• PAGE – The variance for the rows in \text{column} on a page.</td>
<td></td>
</tr>
<tr>
<td>For Crosstabs, specify CROSSTAB for \text{range}:</td>
<td></td>
</tr>
<tr>
<td>• CROSSTAB – (Crosstabs only) The variance for all rows in \text{column} in the crosstab.</td>
<td></td>
</tr>
<tr>
<td>For Graph objects, specify the following:</td>
<td></td>
</tr>
<tr>
<td>• GRAPH – (Graphs only) The variance for rows in \text{column} in the range specified for the Rows option.</td>
<td></td>
</tr>
<tr>
<td>\text{DISTINCT} (optional)</td>
<td>Causes VarP to consider only the distinct values in \text{column} when determining the variance. For a value of \text{column}, the first row found with the value is used and other rows that have the same value are ignored.</td>
</tr>
<tr>
<td>\text{expresn} (optional)</td>
<td>One or more expressions that you want to evaluate to determine distinct rows. \text{Expresn} can be the name of a column, a function, or an expression.</td>
</tr>
</tbody>
</table>

Return value
Double or decimal if the arguments are decimal. Returns the variance for \text{column}. If you specify \text{group}, Var returns the variance for \text{column} within \text{range}.

Usage
If you specify \text{range}, VarP returns the variance for \text{column} within \text{range}. If you specify \text{DISTINCT}, VarP returns the variance for the distinct values in \text{column}, or if you specify \text{expresn}, the variance of the rows in \text{column} where the value of \text{expresn} is distinct.
For graphs, you do not select the range when you call the function. The range has already been determined by the Rows setting on the Data property page (the Range property), and the aggregation function uses that range. Settings for Rows include the following:

- For the Graph presentation style, Rows is always All.
- For Graph controls, Rows can be All, Page, or Group.

**Estimating variance or calculating actual variance**

VarP assumes that the values in column are the values in all rows in the column in the database table. If you did not select all the rows in the column in the SELECT statement, use Var to compute an estimate of the variance of a sample.

**Not in validation rules or filter expressions**

You cannot use this or other aggregate functions in validation rules or filter expressions.

Using an aggregate function cancels the effect of setting Retrieve Rows As Needed in the painter. To do the aggregation, a DataWindow object always retrieves all rows.

**Examples**

These examples all assume that the SELECT statement retrieved all rows in the database table. VarP is intended to work with a full set of data, not a subset.

This expression returns the variance of the values in the column named salary:

\[\text{VarP}(\text{salary})\]

This expression returns the variance of the values in group 1 in the column named salary:

\[\text{VarP}(\text{salary for group 1})\]

This expression returns the variance of the values in column 4 on the page:

\[\text{VarP}(\#4 \text{ for page})\]

This expression entered in the Value box on the Data property page in the graph's property sheet returns the variance of the values in the quantity column in the graph:

\[\text{VarP}(\text{quantity for graph})\]

This expression for a computed field in a crosstab returns the variance of the values in the quantity column in the crosstab:

\[\text{VarP}(\text{quantity for crosstab})\]
Assuming a DataWindow object displays the order number, amount, and line items for each order, this computed field returns the variance of the order amount for the distinct order numbers:

\[ \text{VarP}\left(\text{order\_amt\ for\ all\ DISTINCT\ order\_nbr}\right) \]

See also

- StDevP
- Var

WordCap

**Description**
Sets the first letter of each word in a string to a capital letter and all other letters to lowercase (for example, ROBERT E. LEE would be Robert E. Lee).

**Syntax**

\[
\text{WordCap}\left(\text{string}\right)
\]

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>string</td>
<td>A string or expression that evaluates to a string that you want to display with initial capital letters (for example, Monday Morning)</td>
</tr>
</tbody>
</table>

**Return value**

String. Returns string with the first letter of each word set to uppercase and the remaining letters lowercase if it succeeds, and null if an error occurs.

**Examples**

This expression returns Boston, Massachusetts:

\[ \text{WordCap}\left(\text{"boston, MASSACHUSETTS"}\right) \]

This expression concatenates the characters in the emp_fname and emp_lname columns and makes the first letter of each word uppercase:

\[ \text{WordCap}\left(\text{emp\_fname} + \ "\ + \text{emp\_lname}\right) \]
**Year**

**Description**

Gets the year of a date value.

**Syntax**

```
Year ( date )
```

<table>
<thead>
<tr>
<th>Argument</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>date</td>
<td>The date value from which you want the year</td>
</tr>
</tbody>
</table>

**Return value**

Integer. Returns an integer whose value is a 4-digit year adapted from the year portion of `date` if it succeeds and 1900 if an error occurs.

If the year is two digits, then the century is set as follows. If the year is between 00 to 49, the first two digits are 20; if the year is between 50 and 99, the first two digits are 19.

**Usage**

Obtains the year portion of `date`. Years from 1000 to 3000 inclusive are handled.

If your data includes dates before 1950, such as birth dates, always specify a 4-digit year so that `Year` (and other functions, such as `Sort`) interpret the date as intended.

**Regional settings**

To make sure you get correct return values for the year, you must verify that `yyyy` is the Short Date Style for year in the Regional Settings of the user’s Control Panel.

**Examples**

This expression returns 2005:

```
Year (2005-01-31)
```

**See also**

Day

Month
CHAPTER 3  DataWindow Object Properties

About this chapter
This chapter describes the properties that control the appearance and behavior of a DataWindow object.

Contents

<table>
<thead>
<tr>
<th>Topic</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Overview of DataWindow object properties</td>
<td>143</td>
</tr>
<tr>
<td>Controls in a DataWindow and their properties</td>
<td>145</td>
</tr>
<tr>
<td>Alphabetical list of DataWindow object properties</td>
<td>167</td>
</tr>
</tbody>
</table>

Overview of DataWindow object properties

DataWindow object properties apply to the DataWindow object itself, not to the DataWindow control or DataStore that contains it. There are several ways you can affect the values of DataWindow object properties at runtime:

- Use the general-purpose Describe and Modify methods to get and set property values.
- Use the GetProperty and SetProperty methods.
- Use properties of classes derived from Sybase.DataWindow.GraphicObject.
- Use descendants of the ExpressionBasedProperty class. Each descendant has two properties: Expression and Value. The Expression property is a String. The Value property is of the datatype required in the expression.
- For many properties, enter expressions in the painter that set properties conditionally at runtime.
- Use the DataWindowSyntaxFromSql method to generate DataWindow source code that sets some DataWindow properties. You can use the generated code in the Create method to create new DataWindows.
Overview of DataWindow object properties

Summary tables in the first part of this chapter

The tables in “Controls in a DataWindow and their properties” on page 145 list the properties for each control within a DataWindow object, with short descriptions. There are also tables for DataWindowSyntaxFromSql object keywords. After the first table of DataWindow properties, the tables are alphabetical by control and keyword name.

The tables include check mark columns that identify whether you can use that property with Modify (M) or DataWindowSyntaxFromSql (S). When (exp) is included in the description, you can specify a DataWindow expression as the value for that property. A DataWindow expression lets you specify conditions for determining the property value.

Alphabetical reference list in the second part of this chapter

The second half of this chapter is an alphabetical list of properties with descriptions, syntax, and examples. When you find a property you want to use in the first part, look up the property in the alphabetical list to find the specific syntax you need to use. In the tables that describe the property values, (exp) again indicates that you can use a DataWindow expression for the value.

You can get the value of all properties in all tables

At runtime, you can use Describe or GetProperty to get the value of all properties listed in all tables.
Controls in a DataWindow and their properties

The tables in this section list the properties that apply to DataWindow objects and DataWindowSyntaxFromSql (Group, Style, and Title) keywords.

### Topic for DataWindow objects and keywords

<table>
<thead>
<tr>
<th>Topic for DataWindow objects and keywords</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>Properties for the DataWindow object</td>
<td>145</td>
</tr>
<tr>
<td>Properties for Button controls in DataWindow objects</td>
<td>154</td>
</tr>
<tr>
<td>Properties for Column controls in DataWindow objects</td>
<td>155</td>
</tr>
<tr>
<td>Properties for Computed Field controls in DataWindow objects</td>
<td>156</td>
</tr>
<tr>
<td>Properties for Graph controls in DataWindow objects</td>
<td>157</td>
</tr>
<tr>
<td>Properties for GroupBox controls in DataWindow objects</td>
<td>159</td>
</tr>
<tr>
<td>Properties for the Group keyword</td>
<td>160</td>
</tr>
<tr>
<td>Properties for InkPicture controls in DataWindow objects</td>
<td>160</td>
</tr>
<tr>
<td>Properties for Line controls in DataWindow objects</td>
<td>161</td>
</tr>
<tr>
<td>Properties for OLE Object controls in DataWindow objects</td>
<td>161</td>
</tr>
<tr>
<td>Properties for Oval, Rectangle, and RoundRectangle controls in DataWindow objects</td>
<td>162</td>
</tr>
<tr>
<td>Additional properties for RoundRectangle controls in DataWindow objects</td>
<td>163</td>
</tr>
<tr>
<td>Properties for Picture controls in DataWindow objects</td>
<td>163</td>
</tr>
<tr>
<td>Properties for Report controls in DataWindow objects</td>
<td>164</td>
</tr>
<tr>
<td>Properties for the Style keyword</td>
<td>165</td>
</tr>
<tr>
<td>Properties for TableBlob controls in DataWindow objects</td>
<td>165</td>
</tr>
<tr>
<td>Properties for Text controls in DataWindow objects</td>
<td>166</td>
</tr>
<tr>
<td>Title keyword</td>
<td>167</td>
</tr>
</tbody>
</table>

### Properties for the DataWindow object

An x in the M (Modify) column means you can change the property. An x in the S column means you can use the property with the DataWindowSyntaxFromSql method. When \( \text{exp} \) is included in the description, you can specify a DataWindow expression as the value for that property.

<table>
<thead>
<tr>
<th>Property for the DataWindow</th>
<th>M</th>
<th>S</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
<td></td>
<td></td>
<td>All general properties.</td>
</tr>
<tr>
<td>Bands</td>
<td></td>
<td></td>
<td>List of bands.</td>
</tr>
<tr>
<td>Bandname.property</td>
<td>x</td>
<td></td>
<td>Color, height, and so on for a band, where bandname is Detail, Footer, Header, Summary, Trailer, or TreeView.Level.</td>
</tr>
</tbody>
</table>
Controls in a *DataWindow* and their properties

<table>
<thead>
<tr>
<th>Property for the DataWindow</th>
<th>M</th>
<th>S</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bandname.Text</td>
<td>x</td>
<td></td>
<td>Rich text content where <em>bandname</em> is Detail, Footer, or Header.</td>
</tr>
<tr>
<td>Color</td>
<td>x</td>
<td>x</td>
<td>Background color.</td>
</tr>
<tr>
<td>Column.Count</td>
<td></td>
<td></td>
<td>Number of columns.</td>
</tr>
<tr>
<td>Crosstab.property</td>
<td>x</td>
<td></td>
<td>Settings for a crosstab DataWindow.</td>
</tr>
<tr>
<td>CSSGen.property</td>
<td>x</td>
<td></td>
<td>Settings that specify the physical path to which a generated CSS style sheet is published and the URL where the style sheet is located.</td>
</tr>
<tr>
<td>Data</td>
<td></td>
<td></td>
<td>Description of data.</td>
</tr>
<tr>
<td>Data.HTML</td>
<td></td>
<td></td>
<td>Description of the data and format of the DataWindow in HTML format.</td>
</tr>
<tr>
<td>Data.HTMLTable</td>
<td></td>
<td></td>
<td>Description of the data in the DataWindow in HTML table format.</td>
</tr>
<tr>
<td>Data.XHTML</td>
<td></td>
<td></td>
<td>A string containing the row data content of the DataWindow object in XHTML format.</td>
</tr>
<tr>
<td>Data.XML</td>
<td></td>
<td></td>
<td>A string containing the row data content of the DataWindow object in XML format.</td>
</tr>
<tr>
<td>Data.XMLDTD</td>
<td></td>
<td></td>
<td>A string containing the full document type definition (DTD) of the XML output for a DataWindow object.</td>
</tr>
<tr>
<td>Data.XMLSchema</td>
<td></td>
<td></td>
<td>A string containing the full schema of the XML output of a DataWindow object.</td>
</tr>
<tr>
<td>Data.XSLFO</td>
<td></td>
<td></td>
<td>A string containing XSL Formatting Objects (XSL-FO) that represents the data and presentation of the DataWindow object.</td>
</tr>
<tr>
<td>Detail.property</td>
<td>x</td>
<td></td>
<td>Color, height, and so on for the detail band.</td>
</tr>
<tr>
<td>EditMask.property</td>
<td>x</td>
<td></td>
<td>Settings for EditMask edit style.</td>
</tr>
<tr>
<td>Export.PDF.Distill.CustomPostScript</td>
<td>x</td>
<td></td>
<td>Setting that enables you to specify the PostScript printer driver settings used when data is exported to PDF using the Distill! method.</td>
</tr>
<tr>
<td>Export.PDF.Method</td>
<td></td>
<td></td>
<td>Setting that determines whether data is exported to PDF from a DataWindow object by printing to a PostScript file and distilling to PDF, or by saving in XSL Formatting Objects (XSL-FO) format and processing to PDF.</td>
</tr>
<tr>
<td>Export.PDF.XSLFOP.Print</td>
<td>x</td>
<td></td>
<td>Setting that enables you to send a DataWindow object directly to a printer using platform-independent Java printing when using the XSL-FO method to export to PDF. This is an option of the Apache FOP processor.</td>
</tr>
<tr>
<td>Export.XHTML.TemplateCount</td>
<td></td>
<td></td>
<td>The number of XHTML export templates associated with a DataWindow object.</td>
</tr>
<tr>
<td>Export.XHTML.Template[ ].Name</td>
<td></td>
<td></td>
<td>The name of an XHTML export template associated with a DataWindow object.</td>
</tr>
<tr>
<td>Property for the DataWindow</td>
<td>M</td>
<td>S</td>
<td>Description</td>
</tr>
<tr>
<td>-------------------------------------</td>
<td>---</td>
<td>---</td>
<td>------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Bandname.Text</td>
<td>x</td>
<td></td>
<td>Rich text content where bandname is Detail, Footer, or Header.</td>
</tr>
<tr>
<td>Color</td>
<td>x</td>
<td>x</td>
<td>Background color.</td>
</tr>
<tr>
<td>Column.Count</td>
<td></td>
<td></td>
<td>Number of columns.</td>
</tr>
<tr>
<td>Crosstab.property</td>
<td>x</td>
<td></td>
<td>Settings for a crosstab DataWindow.</td>
</tr>
<tr>
<td>CSSGen.property</td>
<td>x</td>
<td></td>
<td>Settings that specify the physical path to which a generated CSS style sheet is published and the URL where the style sheet is located.</td>
</tr>
<tr>
<td>Data</td>
<td></td>
<td></td>
<td>Description of data.</td>
</tr>
<tr>
<td>Data.HTML</td>
<td></td>
<td></td>
<td>Description of the data and format of the DataWindow in HTML format.</td>
</tr>
<tr>
<td>Data.HTMLTable</td>
<td></td>
<td></td>
<td>Description of the data in the DataWindow in HTML table format.</td>
</tr>
<tr>
<td>Data.XHTML</td>
<td></td>
<td></td>
<td>A string containing the row data content of the DataWindow object in XHTML format.</td>
</tr>
<tr>
<td>Data.XML</td>
<td></td>
<td></td>
<td>A string containing the row data content of the DataWindow object in XML format.</td>
</tr>
<tr>
<td>Data.XMIXMLDTD</td>
<td></td>
<td></td>
<td>A string containing the full document type definition (DTD) of the XML output for a DataWindow object.</td>
</tr>
<tr>
<td>Data.XMLSchema</td>
<td></td>
<td></td>
<td>A string containing the full schema of the XML output of a DataWindow object.</td>
</tr>
<tr>
<td>Data.XMIXMLWeb</td>
<td></td>
<td></td>
<td>A string containing browser-specific JavaScript that performs the XSLT transformation on the browser after the XML Web DataWindow generator generates all necessary components.</td>
</tr>
<tr>
<td>Data.XMIXSLFO</td>
<td></td>
<td></td>
<td>A string containing XSL Formatting Objects (XSL-FO) that represents the data and presentation of the DataWindow object.</td>
</tr>
<tr>
<td>Detail.property</td>
<td>x</td>
<td></td>
<td>Color, height, and so on for the detail band.</td>
</tr>
<tr>
<td>EditMask.property</td>
<td>x</td>
<td></td>
<td>Settings for EditMask edit style.</td>
</tr>
<tr>
<td>Export.PDF.Distill.CustomPostScript</td>
<td>x</td>
<td></td>
<td>Setting that enables you to specify the PostScript printer driver settings used when data is exported to PDF using the Distill! method.</td>
</tr>
<tr>
<td>Export.PDF.Method</td>
<td></td>
<td></td>
<td>Setting that determines whether data is exported to PDF from a DataWindow object by printing to a PostScript file and distilling to PDF, or by saving in XSL Formatting Objects (XSL-FO) format and processing to PDF.</td>
</tr>
<tr>
<td>Export.PDF.XMIXSLFOP.Print</td>
<td>x</td>
<td></td>
<td>Setting that enables you to send a DataWindow object directly to a printer using platform-independent Java printing when using the XSL-FO method to export to PDF. This is an option of the Apache FOP processor.</td>
</tr>
<tr>
<td>Export.XMIXHTML.TemplateCount</td>
<td></td>
<td></td>
<td>The number of XHTML export templates associated with a DataWindow object.</td>
</tr>
<tr>
<td>Export.XMIXHTML.Template[] .Name</td>
<td></td>
<td></td>
<td>The name of an XHTML export template associated with a DataWindow object.</td>
</tr>
<tr>
<td>Property for the DataWindow</td>
<td>M</td>
<td>S</td>
<td>Description</td>
</tr>
<tr>
<td>--------------------------------------</td>
<td>---</td>
<td>---</td>
<td>--------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Export.XHTML.UseTemplate</td>
<td>x</td>
<td></td>
<td>Setting that optionally controls the logical structure of the XHTML generated by a DataWindow object from a DataWindow data expression using dot notation.</td>
</tr>
<tr>
<td>Export.XML.HeadGroups</td>
<td>x</td>
<td></td>
<td>Setting that causes elements, attributes, and all other items above the Detail Start element in an XML export template for a group DataWindow to be iterated for each group in the exported XML.</td>
</tr>
<tr>
<td>Export.XML.IncludeWhitespace</td>
<td>x</td>
<td></td>
<td>Setting that determines whether the XML document is formatted by inserting whitespace characters (carriage returns, linefeeds, tabs, and spacebar spaces).</td>
</tr>
<tr>
<td>Export.XML.MetaDataType</td>
<td>x</td>
<td></td>
<td>Setting that controls the type of metadata generated with the XML exported from a DataWindow object using the SaveAs method or a .Data.XML expression.</td>
</tr>
<tr>
<td>Export.XML.SaveMetaData</td>
<td>x</td>
<td></td>
<td>Setting that controls the storage format for the metadata generated with the XML exported from a DataWindow object using the SaveAs method or a .Data.XML expression.</td>
</tr>
<tr>
<td>Export.XML.TemplateCount</td>
<td></td>
<td></td>
<td>The number of XML export templates associated with a DataWindow object.</td>
</tr>
<tr>
<td>Export.XML.Template[ ].Name</td>
<td></td>
<td></td>
<td>The name of an XML export template associated with a DataWindow object.</td>
</tr>
<tr>
<td>Export.XML.UseTemplate</td>
<td>x</td>
<td></td>
<td>Setting that optionally controls the logical structure of the XML exported from a DataWindow object using the SaveAs method or the .Data.XML property.</td>
</tr>
<tr>
<td>FirstRowOnPage</td>
<td></td>
<td></td>
<td>The row number of the first displayed row.</td>
</tr>
<tr>
<td>Font.Bias</td>
<td>x</td>
<td></td>
<td>Treat fonts as display or printer.</td>
</tr>
<tr>
<td>Footer.property</td>
<td>x</td>
<td></td>
<td>Color, height, and so on for the footer band (see Bandname.property in this table).</td>
</tr>
<tr>
<td>Grid.ColumnMove</td>
<td>x</td>
<td></td>
<td>Whether the user can drag to reposition columns.</td>
</tr>
<tr>
<td>Grid.Lines</td>
<td>x</td>
<td></td>
<td>Options for lines in grid DataWindow and crosstab.</td>
</tr>
<tr>
<td>Header.#.property</td>
<td>x</td>
<td></td>
<td>Color, height, and so on for a group's header band.</td>
</tr>
<tr>
<td>Header.property</td>
<td>x</td>
<td></td>
<td>Color, height, and so on for the header band.</td>
</tr>
<tr>
<td>Help.property</td>
<td>x</td>
<td></td>
<td>Help settings for DataWindow actions.</td>
</tr>
<tr>
<td>HideGrayLine</td>
<td>x</td>
<td></td>
<td>Whether a gray line displays at page boundaries.</td>
</tr>
<tr>
<td>HorizontalScrollMaximum</td>
<td></td>
<td></td>
<td>Width of scroll box in the horizontal scroll bar.</td>
</tr>
<tr>
<td>HorizontalScrollMaximum2</td>
<td></td>
<td></td>
<td>Width of second scroll box when scroll bar is split.</td>
</tr>
<tr>
<td>HorizontalScrollPosition</td>
<td>x</td>
<td></td>
<td>Position of the scroll box in the scroll bar.</td>
</tr>
<tr>
<td>HorizontalScrollPosition2</td>
<td>x</td>
<td></td>
<td>Position of scroll box in second split scroll bar.</td>
</tr>
<tr>
<td>HorizontalScrollSplit</td>
<td>x</td>
<td></td>
<td>The position of the split in the scroll bar.</td>
</tr>
<tr>
<td>HTMLDW</td>
<td>x</td>
<td></td>
<td>(exp) Whether HTML for the DataWindow is interactive and coordinated with a server component for retrievals and updates.</td>
</tr>
<tr>
<td>HTMLGen.property</td>
<td>x</td>
<td></td>
<td>(exp) Settings for HTML generation.</td>
</tr>
<tr>
<td>HTMLTable.property</td>
<td>x</td>
<td></td>
<td>Settings for the display of DataWindow data when displayed in HTML table format.</td>
</tr>
<tr>
<td>Property for the DataWindow</td>
<td>M</td>
<td>S</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------</td>
<td>---</td>
<td>---</td>
<td>-------------</td>
</tr>
<tr>
<td>Import.XML.Trace</td>
<td>x</td>
<td></td>
<td>Setting that determines whether import trace information is written to a log file.</td>
</tr>
<tr>
<td>Import.XML.TraceFile</td>
<td>x</td>
<td></td>
<td>Specifies the name and location of an import trace file.</td>
</tr>
<tr>
<td>Import.XML.UseTemplate</td>
<td>x</td>
<td></td>
<td>Setting that optionally controls the logical structure of the XML imported from an XML file to a DataWindow object using the ImportFile method.</td>
</tr>
<tr>
<td>JSGen.property</td>
<td>x</td>
<td></td>
<td>Settings that specify the physical path to which generated JavaScript is published and the URL indicating the location of the generated JavaScript.</td>
</tr>
<tr>
<td>Label.property</td>
<td>x</td>
<td>x</td>
<td>Settings for the Label presentation style.</td>
</tr>
<tr>
<td>LastRowOnPage</td>
<td></td>
<td></td>
<td>The last visible row on the page.</td>
</tr>
<tr>
<td>Message.Title</td>
<td>x</td>
<td>x</td>
<td>The title of the dialog box that displays errors.</td>
</tr>
<tr>
<td>Nested</td>
<td></td>
<td></td>
<td>Whether the DataWindow has nested reports.</td>
</tr>
<tr>
<td>NoUserPrompt</td>
<td>x</td>
<td></td>
<td>Determines whether an error message is displayed to the user.</td>
</tr>
<tr>
<td>Objects</td>
<td></td>
<td></td>
<td>The controls in the DataWindow.</td>
</tr>
<tr>
<td>OLE.Client.property</td>
<td>x</td>
<td></td>
<td>Settings for the DataWindow as OLE client.</td>
</tr>
<tr>
<td>Pointer</td>
<td>x</td>
<td></td>
<td>(exp) The pointer when over the DataWindow.</td>
</tr>
<tr>
<td>Print.Preview.property</td>
<td>x</td>
<td></td>
<td>Various settings for print preview.</td>
</tr>
<tr>
<td>Print.property</td>
<td>x</td>
<td>x</td>
<td>Various settings for printing.</td>
</tr>
<tr>
<td>Printer</td>
<td>x</td>
<td></td>
<td>The currently selected printer.</td>
</tr>
<tr>
<td>Processing</td>
<td></td>
<td></td>
<td>Processing required by the presentation style.</td>
</tr>
<tr>
<td>QueryMode</td>
<td>x</td>
<td></td>
<td>Whether the DataWindow is in query mode.</td>
</tr>
<tr>
<td>QuerySort</td>
<td>x</td>
<td></td>
<td>Whether to sort the result set from the query.</td>
</tr>
<tr>
<td>ReadOnly</td>
<td>x</td>
<td></td>
<td>Whether the DataWindow is read-only.</td>
</tr>
<tr>
<td>Retrieve.AsNeeded</td>
<td>x</td>
<td></td>
<td>Whether to retrieve data only as needed.</td>
</tr>
<tr>
<td>RichText.property</td>
<td>x</td>
<td></td>
<td>Settings for a RichText DataWindow.</td>
</tr>
<tr>
<td>Row.Resize</td>
<td>x</td>
<td></td>
<td>Whether user can change the height of rows.</td>
</tr>
<tr>
<td>Rows_Per_Detail</td>
<td>x</td>
<td></td>
<td>Number of rows in each column of N-Up style.</td>
</tr>
<tr>
<td>Selected</td>
<td>x</td>
<td></td>
<td>List of selected controls.</td>
</tr>
<tr>
<td>Selected.Data</td>
<td></td>
<td></td>
<td>List of selected data.</td>
</tr>
<tr>
<td>Selected.Mouse</td>
<td>x</td>
<td></td>
<td>Whether user can use the mouse to select.</td>
</tr>
<tr>
<td>ShowBackColorOnXP</td>
<td>x</td>
<td></td>
<td>Whether the background color that you select for a button displays on Windows XP.</td>
</tr>
<tr>
<td>ShowDefinition</td>
<td>x</td>
<td></td>
<td>(exp) Display column names instead of data.</td>
</tr>
<tr>
<td>Sparse</td>
<td>x</td>
<td></td>
<td>(exp) The repeating columns to be suppressed.</td>
</tr>
<tr>
<td>Storage</td>
<td></td>
<td></td>
<td>The amount of storage used by DataWindow.</td>
</tr>
<tr>
<td>StoragePageSize</td>
<td></td>
<td></td>
<td>The default page size for DataWindow storage.</td>
</tr>
<tr>
<td>Summary.property</td>
<td>x</td>
<td></td>
<td>Color, height, and so on for the summary band.</td>
</tr>
<tr>
<td>Syntax</td>
<td></td>
<td></td>
<td>The syntax of the DataWindow.</td>
</tr>
<tr>
<td>Syntax.Data</td>
<td></td>
<td></td>
<td>The data of the DataWindow in parse format.</td>
</tr>
</tbody>
</table>
## Controls in a DataWindow and their properties

<table>
<thead>
<tr>
<th>Property for the DataWindow</th>
<th>M</th>
<th>S</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Syntax.Modified</td>
<td>x</td>
<td></td>
<td>Whether the syntax has been modified.</td>
</tr>
<tr>
<td>Table.property</td>
<td>x</td>
<td></td>
<td>Various settings for the database.</td>
</tr>
<tr>
<td>Table.sqlaction.property</td>
<td>x</td>
<td></td>
<td>Stored procedures for update activity.</td>
</tr>
<tr>
<td>Timer.Interval</td>
<td>x</td>
<td>x</td>
<td>The milliseconds between timer events.</td>
</tr>
<tr>
<td>Trailer.#.property</td>
<td>x</td>
<td></td>
<td>Color, height, and so on for a group’s trailer band.</td>
</tr>
<tr>
<td>Tree.property</td>
<td>x</td>
<td></td>
<td>Settings for a TreeView DataWindow.</td>
</tr>
<tr>
<td>Tree.Leaf.TreeNodeIconName</td>
<td>x</td>
<td></td>
<td>The file name of the tree node icon in the detail band of a TreeView DataWindow.</td>
</tr>
<tr>
<td>Tree.Level.#.property</td>
<td>x</td>
<td></td>
<td>The file name of the icon for a TreeView node in a TreeView level band when the icon is in either the expanded or collapsed state.</td>
</tr>
<tr>
<td>Units</td>
<td>x</td>
<td></td>
<td>The unit of measure for the DataWindow.</td>
</tr>
<tr>
<td>VerticalScrollMaximum</td>
<td></td>
<td></td>
<td>The height of the scroll box in the scroll bar.</td>
</tr>
<tr>
<td>VerticalScrollPosition</td>
<td>x</td>
<td></td>
<td>The position of the scroll box in the scroll bar.</td>
</tr>
<tr>
<td>XHTMLGen.Browser</td>
<td>x</td>
<td></td>
<td>A string that identifies the browser in which XHTML generated within an XSLT style sheet is displayed.</td>
</tr>
<tr>
<td>XMLGen.property</td>
<td>x</td>
<td></td>
<td>Settings that specify the physical path to which XML is published and the URL referenced by the JavaScript that transforms the XML to XHTML.</td>
</tr>
<tr>
<td>XSLTGen.property</td>
<td>x</td>
<td></td>
<td>Settings that specify the physical path to which the generated XSLT style sheet is published and the URL referenced by the JavaScript that transforms the XML to XHTML.</td>
</tr>
<tr>
<td>Zoom</td>
<td>x</td>
<td></td>
<td>The scaling percentage of the DataWindow.</td>
</tr>
</tbody>
</table>

### Property for the DataWindow

<table>
<thead>
<tr>
<th>Property for the DataWindow</th>
<th>M</th>
<th>S</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
<td></td>
<td></td>
<td>All general properties.</td>
</tr>
<tr>
<td>Bands</td>
<td></td>
<td></td>
<td>List of bands.</td>
</tr>
<tr>
<td>Bandname.property</td>
<td>x</td>
<td></td>
<td>Color, height, and so on for a band, where bandname is Detail, Footer, Header, Summary, or Trailer.</td>
</tr>
<tr>
<td>Color</td>
<td>x</td>
<td>x</td>
<td>Background color.</td>
</tr>
<tr>
<td>Column.Count</td>
<td></td>
<td></td>
<td>Number of columns.</td>
</tr>
<tr>
<td>Crosstab.property</td>
<td>x</td>
<td></td>
<td>Settings for a crosstab DataWindow.</td>
</tr>
<tr>
<td>CSSGen.property</td>
<td>x</td>
<td></td>
<td>Settings that specify the physical path to which a generated CSS style sheet is published and the URL where the style sheet is located.</td>
</tr>
<tr>
<td>Data</td>
<td></td>
<td></td>
<td>Description of data.</td>
</tr>
<tr>
<td>Data.HTML</td>
<td></td>
<td></td>
<td>Description of the data and format of the DataWindow in HTML format.</td>
</tr>
<tr>
<td>Data.HTMLTable</td>
<td></td>
<td></td>
<td>Description of the data in the DataWindow in HTML table format.</td>
</tr>
</tbody>
</table>
### Property for the DataWindow

<table>
<thead>
<tr>
<th>Property for the DataWindow</th>
<th>M</th>
<th>S</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data.XHTML</td>
<td></td>
<td></td>
<td>A string containing the row data content of the DataWindow object in XHTML format.</td>
</tr>
<tr>
<td>Data.XML</td>
<td></td>
<td></td>
<td>A string containing the row data content of the DataWindow object in XML format.</td>
</tr>
<tr>
<td>Data.XMLDTD</td>
<td></td>
<td></td>
<td>A string containing the full document type definition (DTD) of the XML output for a DataWindow object.</td>
</tr>
<tr>
<td>Data.XMLSchema</td>
<td></td>
<td></td>
<td>A string containing the full schema of the XML output of a DataWindow object.</td>
</tr>
<tr>
<td>Data.XMLWeb</td>
<td></td>
<td></td>
<td>A string containing browser-specific JavaScript that performs the XSLT transformation on the browser after the XML Web DataWindow generator generates all necessary components.</td>
</tr>
<tr>
<td>Data.XSLFO</td>
<td></td>
<td></td>
<td>A string containing XSL Formatting Objects (XSL-FO) that represents the data and presentation of the DataWindow object.</td>
</tr>
<tr>
<td>Detail.property</td>
<td>x</td>
<td></td>
<td>Color, height, and so on for the detail band.</td>
</tr>
<tr>
<td>EditMask.property</td>
<td>x</td>
<td></td>
<td>Settings for EditMask edit style.</td>
</tr>
<tr>
<td>Export.PDF.Distill.CustomPostScript</td>
<td>x</td>
<td></td>
<td>Setting that enables you to specify the PostScript printer driver settings used when data is exported to PDF using the Distill! method.</td>
</tr>
<tr>
<td>Export.PDF.Method</td>
<td></td>
<td></td>
<td>Setting that determines whether data is exported to PDF from a DataWindow object by printing to a PostScript file and distilling to PDF, or by saving in XSL Formatting Objects (XSL-FO) format and processing to PDF.</td>
</tr>
<tr>
<td>Export.PDF.XSLFOP.Print</td>
<td>x</td>
<td></td>
<td>Setting that enables you to send a DataWindow object directly to a printer using platform-independent Java printing when using the XSL-FO method to export to PDF. This is an option of the Apache FOP processor.</td>
</tr>
<tr>
<td>Export.XHTML.TemplateCount</td>
<td></td>
<td></td>
<td>The number of XHTML export templates associated with a DataWindow object.</td>
</tr>
<tr>
<td>Export.XHTML.Template[ ].Name</td>
<td></td>
<td></td>
<td>The name of an XHTML export template associated with a DataWindow object.</td>
</tr>
<tr>
<td>Export.XHTML.UseTemplate</td>
<td>x</td>
<td></td>
<td>Setting that optionally controls the logical structure of the XHTML generated by a DataWindow object from a DataWindow data expression.</td>
</tr>
<tr>
<td>Export.XML.HeadGroups</td>
<td>x</td>
<td></td>
<td>Setting that causes elements, attributes, and all other items above the Detail Start element in an XML export template for a group DataWindow to be iterated for each group in the exported XML.</td>
</tr>
<tr>
<td>Export.XML.IncludeWhitespace</td>
<td>x</td>
<td></td>
<td>Setting that determines whether the XML document is formatted by inserting white space characters (carriage returns, linefeeds, tabs, and spacebar spaces).</td>
</tr>
</tbody>
</table>
## Controls in a DataWindow and their properties

<table>
<thead>
<tr>
<th>Property for the DataWindow</th>
<th>M</th>
<th>S</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Export.XML.MetaDataType</td>
<td>x</td>
<td></td>
<td>Setting that controls the type of metadata generated with the XML exported from a DataWindow object using the <code>SaveAs</code> method or a <code>.Data.XML</code> expression.</td>
</tr>
<tr>
<td>Export.XML.SaveMetaData</td>
<td>x</td>
<td></td>
<td>Setting that controls the storage format for the metadata generated with the XML exported from a DataWindow object using the <code>SaveAs</code> method or a <code>.Data.XML</code> expression.</td>
</tr>
<tr>
<td>Export.XML.TemplateCount</td>
<td></td>
<td></td>
<td>The number of XML export templates associated with a DataWindow object.</td>
</tr>
<tr>
<td>Export.XML.Template[i].Name</td>
<td></td>
<td></td>
<td>The name of an XML export template associated with a DataWindow object.</td>
</tr>
<tr>
<td>Export.XML.UseTemplate</td>
<td>x</td>
<td></td>
<td>Setting that optionally controls the logical structure of the XML exported from a DataWindow object using the <code>SaveAs</code> method or the <code>.Data.XML</code> property.</td>
</tr>
<tr>
<td>FirstRowOnPage</td>
<td></td>
<td></td>
<td>The row number of the first displayed row.</td>
</tr>
<tr>
<td>Font.Bias</td>
<td>x</td>
<td></td>
<td>Treat fonts as display or printer.</td>
</tr>
<tr>
<td>Footer.property</td>
<td>x</td>
<td></td>
<td>Color, height, and so on for the footer band (see <code>Bandname.property</code> in this table).</td>
</tr>
<tr>
<td>GraphType</td>
<td>x</td>
<td></td>
<td>(exp) The type of graph (pie, bar, and so on) for a Graph DataWindow. For additional Graph properties, see “Properties for Graph controls in DataWindow objects” on page 157.</td>
</tr>
<tr>
<td>Grid.ColumnMove</td>
<td>x</td>
<td></td>
<td>Whether the user can drag to reposition columns.</td>
</tr>
<tr>
<td>Grid.Lines</td>
<td>x</td>
<td></td>
<td>Options for lines in grid DataWindow and crosstab.</td>
</tr>
<tr>
<td>Header.#.property</td>
<td>x</td>
<td></td>
<td>Color, height, and so on for a group’s header band.</td>
</tr>
<tr>
<td>Header.property</td>
<td>x</td>
<td></td>
<td>Color, height, and so on for the header band.</td>
</tr>
<tr>
<td>Help.property</td>
<td>x</td>
<td></td>
<td>Help settings for DataWindow actions.</td>
</tr>
<tr>
<td>HideGrayLine</td>
<td>x</td>
<td></td>
<td>Whether a gray line displays at page boundaries.</td>
</tr>
<tr>
<td>HorizontalScrollMaximum</td>
<td></td>
<td></td>
<td>Width of scroll box in the horizontal scroll bar.</td>
</tr>
<tr>
<td>HorizontalScrollMaximum2</td>
<td></td>
<td></td>
<td>Width of second scroll box when scroll bar is split.</td>
</tr>
<tr>
<td>HorizontalScrollPosition</td>
<td>x</td>
<td></td>
<td>Position of the scroll box in the scroll bar.</td>
</tr>
<tr>
<td>HorizontalScrollPosition2</td>
<td>x</td>
<td></td>
<td>Position of scroll box in second split scroll bar.</td>
</tr>
<tr>
<td>HorizontalScrollSplit</td>
<td>x</td>
<td></td>
<td>The position of the split in the scroll bar.</td>
</tr>
<tr>
<td>HTMLDW</td>
<td>x</td>
<td></td>
<td>(exp) Whether HTML for the DataWindow is interactive and coordinated with a server component for retrievals and updates.</td>
</tr>
<tr>
<td>HTMLGen.property</td>
<td>x</td>
<td></td>
<td>(exp) Settings for HTML generation.</td>
</tr>
<tr>
<td>HTMLTable.property</td>
<td>x</td>
<td></td>
<td>Settings for the display of DataWindow data when displayed in HTML table format.</td>
</tr>
<tr>
<td>Import.XML.Trace</td>
<td>x</td>
<td></td>
<td>Setting that determines whether import trace information is written to a log file.</td>
</tr>
<tr>
<td>Import.XML.TraceFile</td>
<td>x</td>
<td></td>
<td>Specifies the name and location of an import trace file.</td>
</tr>
<tr>
<td>Property for the DataWindow</td>
<td>M</td>
<td>S</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------------------------------------------</td>
<td>---</td>
<td>---</td>
<td>----------------------------------------------------------------------------------------------------------------------------------------------</td>
</tr>
<tr>
<td>Import.XML.UseTemplate</td>
<td>x</td>
<td></td>
<td>Setting that optionally controls the logical structure of the XML imported from an XML file to a DataWindow object using the ImportFile method.</td>
</tr>
<tr>
<td>JSGen.property</td>
<td>x</td>
<td></td>
<td>Settings that specify the physical path to which generated JavaScript is published and the URL indicating the location of the generated JavaScript.</td>
</tr>
<tr>
<td>Label.property</td>
<td>x</td>
<td>x</td>
<td>Settings for the Label presentation style.</td>
</tr>
<tr>
<td>LastRowOnPage</td>
<td></td>
<td></td>
<td>The last visible row on the page.</td>
</tr>
<tr>
<td>Message.Title</td>
<td>x</td>
<td>x</td>
<td>The title of the dialog box that displays errors.</td>
</tr>
<tr>
<td>Nested</td>
<td></td>
<td></td>
<td>Whether the DataWindow has nested reports.</td>
</tr>
<tr>
<td>NoUserPrompt</td>
<td>x</td>
<td></td>
<td>Determines whether an error message is displayed to the user.</td>
</tr>
<tr>
<td>Objects</td>
<td></td>
<td></td>
<td>The controls in the DataWindow.</td>
</tr>
<tr>
<td>OLE.Client.property</td>
<td>x</td>
<td></td>
<td>Settings for the DataWindow as OLE client.</td>
</tr>
<tr>
<td>Pointer</td>
<td>x</td>
<td></td>
<td>(exp) The pointer when over the DataWindow.</td>
</tr>
<tr>
<td>Print.Preview.property</td>
<td>x</td>
<td></td>
<td>Various settings for print preview.</td>
</tr>
<tr>
<td>Print.property</td>
<td>x</td>
<td>x</td>
<td>Various settings for printing.</td>
</tr>
<tr>
<td>Printer</td>
<td>x</td>
<td></td>
<td>The currently selected printer.</td>
</tr>
<tr>
<td>Processing</td>
<td></td>
<td></td>
<td>Processing required by the presentation style.</td>
</tr>
<tr>
<td>QueryMode</td>
<td>x</td>
<td></td>
<td>Whether the DataWindow is in query mode.</td>
</tr>
<tr>
<td>QuerySort</td>
<td>x</td>
<td></td>
<td>Whether to sort the result set from the query.</td>
</tr>
<tr>
<td>ReadOnly</td>
<td>x</td>
<td></td>
<td>Whether the DataWindow is read-only.</td>
</tr>
<tr>
<td>Retrieve.AsNeeded</td>
<td>x</td>
<td></td>
<td>Whether to retrieve data only as needed.</td>
</tr>
<tr>
<td>Row.Resize</td>
<td>x</td>
<td></td>
<td>Whether user can change the height of rows.</td>
</tr>
<tr>
<td>Rows_Per_Detail</td>
<td>x</td>
<td></td>
<td>Number of rows in each column of N-Up style.</td>
</tr>
<tr>
<td>Selected</td>
<td>x</td>
<td></td>
<td>List of selected controls.</td>
</tr>
<tr>
<td>Selected.Data</td>
<td></td>
<td></td>
<td>List of selected data.</td>
</tr>
<tr>
<td>Selected.Mouse</td>
<td>x</td>
<td></td>
<td>Whether user can use the mouse to select.</td>
</tr>
<tr>
<td>ShowDefinition</td>
<td>x</td>
<td></td>
<td>(exp) Display column names instead of data.</td>
</tr>
<tr>
<td>Sparse</td>
<td>x</td>
<td></td>
<td>(exp) The repeating columns to be suppressed.</td>
</tr>
<tr>
<td>Storage</td>
<td></td>
<td></td>
<td>The amount of storage used by DataWindow.</td>
</tr>
<tr>
<td>StoragePageSize</td>
<td></td>
<td></td>
<td>The default page size for DataWindow storage.</td>
</tr>
<tr>
<td>Summary.property</td>
<td>x</td>
<td></td>
<td>Color, height, and so on for the summary band.</td>
</tr>
<tr>
<td>Syntax</td>
<td></td>
<td></td>
<td>The syntax of the DataWindow.</td>
</tr>
<tr>
<td>Syntax.Data</td>
<td></td>
<td></td>
<td>The data of the DataWindow in parse format.</td>
</tr>
<tr>
<td>Syntax.Modified</td>
<td>x</td>
<td></td>
<td>Whether the syntax has been modified.</td>
</tr>
<tr>
<td>Table.property</td>
<td>x</td>
<td></td>
<td>Various settings for the database.</td>
</tr>
<tr>
<td>Table.sqlaction.property</td>
<td>x</td>
<td></td>
<td>Stored procedures for update activity.</td>
</tr>
<tr>
<td>Timer_Interval</td>
<td>x</td>
<td>x</td>
<td>The milliseconds between timer events.</td>
</tr>
<tr>
<td>Trailer.#.property</td>
<td>x</td>
<td></td>
<td>Color, height, and so on for a group’s trailer band.</td>
</tr>
</tbody>
</table>
Controls in a DataWindow and their properties

<table>
<thead>
<tr>
<th>Property for the DataWindow</th>
<th>M</th>
<th>S</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Units</td>
<td>x</td>
<td></td>
<td>The unit of measure for the DataWindow.</td>
</tr>
<tr>
<td>VerticalScrollMaximum</td>
<td>x</td>
<td></td>
<td>The height of the scroll box in the scroll bar.</td>
</tr>
<tr>
<td>VerticalScrollPosition</td>
<td>x</td>
<td></td>
<td>The position of the scroll box in the scroll bar.</td>
</tr>
<tr>
<td>XHTMLGen.Browser</td>
<td>x</td>
<td></td>
<td>A string that identifies the browser in which XHTML generated within an XSLT style sheet is displayed.</td>
</tr>
<tr>
<td>XMLGen.property</td>
<td>x</td>
<td></td>
<td>Settings that specify the physical path to which XML is published and the URL referenced by the JavaScript that transforms the XML to XHTML.</td>
</tr>
<tr>
<td>XSLTGen.property</td>
<td>x</td>
<td></td>
<td>Settings that specify the physical path to which the generated XSLT style sheet is published and the URL referenced by the JavaScript that transforms the XML to XHTML.</td>
</tr>
<tr>
<td>Zoom</td>
<td>x</td>
<td></td>
<td>The scaling percentage of the DataWindow.</td>
</tr>
</tbody>
</table>

Properties for Button controls in DataWindow objects

<table>
<thead>
<tr>
<th>Property for a Button</th>
<th>M</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccessibleDescription</td>
<td>x</td>
<td>A description of the control for use by assistive technology tools.</td>
</tr>
<tr>
<td>AccessibleName</td>
<td>x</td>
<td>A descriptive label for the control.</td>
</tr>
<tr>
<td>AccessibleRole</td>
<td></td>
<td>A description of the kind of user-interface element that the control is.</td>
</tr>
<tr>
<td>Background.property</td>
<td>x</td>
<td>Background settings for the button.</td>
</tr>
<tr>
<td>Color</td>
<td>x</td>
<td>(exp) The text color.</td>
</tr>
<tr>
<td>DefaultPicture</td>
<td>x</td>
<td>Whether or not the action’s default picture is to be used on the button (user-defined action has no default picture).</td>
</tr>
<tr>
<td>Enabled</td>
<td>x</td>
<td>Determines whether a button control on a DataWindow is enabled.</td>
</tr>
<tr>
<td>Filename</td>
<td>x</td>
<td>(exp) Name of the file containing the picture to be used on the button (if not specified, just the text is used).</td>
</tr>
<tr>
<td>Font.property</td>
<td>x</td>
<td>(exp) Font settings for the text.</td>
</tr>
<tr>
<td>HTextAlign</td>
<td>x</td>
<td>(exp) How the text in the button is horizontally aligned. Values are: 0 (center), 1 (left), 2 (right).</td>
</tr>
<tr>
<td>Height</td>
<td>x</td>
<td>(exp) The height of the button control.</td>
</tr>
<tr>
<td>HideSnaked</td>
<td>x</td>
<td>Whether the button control appears once per page when printing newspaper columns.</td>
</tr>
<tr>
<td>Moveable</td>
<td>x</td>
<td>Whether the user can move the button control.</td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td>The name of the button control.</td>
</tr>
<tr>
<td>Pointer</td>
<td>x</td>
<td>(exp) The pointer image when it is over the button control.</td>
</tr>
<tr>
<td>Resizeable</td>
<td>x</td>
<td>Whether the user can resize the button control.</td>
</tr>
<tr>
<td>SlideLeft</td>
<td>x</td>
<td>(exp) Whether the button control moves left to fill in empty space.</td>
</tr>
<tr>
<td>SlideUp</td>
<td>x</td>
<td>(exp) How the button control moves up to fill in empty space.</td>
</tr>
</tbody>
</table>
### Property for a Button

<table>
<thead>
<tr>
<th><strong>Property</strong></th>
<th><strong>M</strong></th>
<th><strong>S</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>SuppressEventProcessing</td>
<td>x</td>
<td></td>
<td>Whether or not ButtonClicked and ButtonClicking events are fired for this particular button.</td>
</tr>
<tr>
<td>Tag</td>
<td>x</td>
<td></td>
<td><em>(exp)</em> The tag text for the button control.</td>
</tr>
<tr>
<td>Text</td>
<td>x</td>
<td></td>
<td><em>(exp)</em> The displayed text.</td>
</tr>
<tr>
<td>Type</td>
<td></td>
<td></td>
<td>The control's type, which is button.</td>
</tr>
<tr>
<td>VTextAlign</td>
<td>x</td>
<td></td>
<td><em>(exp)</em> How the text in the button is vertically aligned. Values are: 0 (center), 1 (top), 2 (bottom), 3 (multiline).</td>
</tr>
<tr>
<td>Visible</td>
<td>x</td>
<td></td>
<td><em>(exp)</em> Whether the button control is visible.</td>
</tr>
<tr>
<td>Width</td>
<td>x</td>
<td></td>
<td><em>(exp)</em> The width of the button control.</td>
</tr>
<tr>
<td>X</td>
<td>x</td>
<td></td>
<td><em>(exp)</em> The x coordinate of the button control.</td>
</tr>
<tr>
<td>Y</td>
<td>x</td>
<td></td>
<td><em>(exp)</em> The y coordinate of the button control.</td>
</tr>
</tbody>
</table>

### Properties for Column controls in DataWindow objects

<table>
<thead>
<tr>
<th><strong>Property for a Column</strong></th>
<th><strong>M</strong></th>
<th><strong>S</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>AccessibleDescription</td>
<td>x</td>
<td></td>
<td>A description of the control for use by assistive technology tools.</td>
</tr>
<tr>
<td>AccessibleName</td>
<td>x</td>
<td></td>
<td>A descriptive label for the control.</td>
</tr>
<tr>
<td>AccessibleRole</td>
<td>x</td>
<td></td>
<td>A description of the kind of user-interface element that the control is.</td>
</tr>
<tr>
<td>Accelerator</td>
<td>x</td>
<td></td>
<td><em>(exp)</em> The accelerator key for the column.</td>
</tr>
<tr>
<td>Alignment</td>
<td>x</td>
<td></td>
<td><em>(exp)</em> The alignment of the column's text.</td>
</tr>
<tr>
<td>Attributes</td>
<td>x</td>
<td></td>
<td>A list of the properties of the column.</td>
</tr>
<tr>
<td>Background.property</td>
<td>x</td>
<td>x</td>
<td><em>(exp)</em> Background settings for the column.</td>
</tr>
<tr>
<td>Band</td>
<td></td>
<td></td>
<td>The band containing the column.</td>
</tr>
<tr>
<td>BitmapName</td>
<td></td>
<td></td>
<td>Whether the column's content names a picture that will be displayed instead of the text.</td>
</tr>
<tr>
<td>Border</td>
<td>x</td>
<td>x</td>
<td><em>(exp)</em> The type of border around the column.</td>
</tr>
<tr>
<td>CheckBox.property</td>
<td>x</td>
<td></td>
<td>Settings for CheckBox edit style.</td>
</tr>
<tr>
<td>Color</td>
<td>x</td>
<td>x</td>
<td><em>(exp)</em> The text color.</td>
</tr>
<tr>
<td>ColType</td>
<td></td>
<td></td>
<td>The column's datatype.</td>
</tr>
<tr>
<td>Criteria.property</td>
<td>x</td>
<td></td>
<td>Settings for column in Prompt for Criteria dialog box.</td>
</tr>
<tr>
<td>dbAlias</td>
<td>x</td>
<td></td>
<td>An alias for the name of the database column.</td>
</tr>
<tr>
<td>dbName</td>
<td>x</td>
<td></td>
<td>The name of the database column.</td>
</tr>
<tr>
<td>dddw.property</td>
<td>x</td>
<td></td>
<td>Settings for DropDownDataWindow edit style.</td>
</tr>
<tr>
<td>dddl.property</td>
<td>x</td>
<td></td>
<td>Settings for DropDownListBox edit style.</td>
</tr>
<tr>
<td>Edit.property</td>
<td>x</td>
<td>x</td>
<td>Settings for Edit edit style.</td>
</tr>
<tr>
<td>EditMask.property</td>
<td>x</td>
<td></td>
<td>Settings for EditMask edit style.</td>
</tr>
<tr>
<td>Font.property</td>
<td>x</td>
<td>x</td>
<td><em>(exp)</em> Font settings for the column text.</td>
</tr>
</tbody>
</table>
### Controls in a DataWindow and their properties

<table>
<thead>
<tr>
<th>Property for a Column</th>
<th>M</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Format</td>
<td>x</td>
<td><em>(exp)</em> The column’s display format.</td>
</tr>
<tr>
<td>Height</td>
<td>x</td>
<td><em>(exp)</em> The height of the column.</td>
</tr>
<tr>
<td>Height.AutoSize</td>
<td>x</td>
<td>Whether column height is adjusted to fit the data.</td>
</tr>
<tr>
<td>HideSnaked</td>
<td>x</td>
<td>Whether the control appears once per page when printing newspaper columns.</td>
</tr>
<tr>
<td>HTML.property</td>
<td>x</td>
<td><em>(exp)</em> Settings for creating hyperlinks for column data.</td>
</tr>
<tr>
<td>Identity</td>
<td>x</td>
<td>Whether the DBMS sets the column’s value.</td>
</tr>
<tr>
<td>ID</td>
<td></td>
<td>The number of the column.</td>
</tr>
<tr>
<td>Ink.property</td>
<td>x</td>
<td>Settings for Ink attributes of the InkEdit edit style.</td>
</tr>
<tr>
<td>InkEdit.property</td>
<td>x</td>
<td>Settings for InkEdit edit style.</td>
</tr>
<tr>
<td>Initial</td>
<td>x</td>
<td>The initial value in the column for a new row.</td>
</tr>
<tr>
<td>Key</td>
<td>x</td>
<td>Whether column is part of the table’s primary key.</td>
</tr>
<tr>
<td>Moveable</td>
<td>x</td>
<td>Whether the user can move the column.</td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td>The name of the column.</td>
</tr>
<tr>
<td>Pointer</td>
<td>x</td>
<td><em>(exp)</em> The pointer’s image when it is over the column.</td>
</tr>
<tr>
<td>Protect</td>
<td>x</td>
<td><em>(exp)</em> Whether the column is protected from changes.</td>
</tr>
<tr>
<td>RadioButtons.property</td>
<td>x</td>
<td>Settings for RadioButton edit style.</td>
</tr>
<tr>
<td>Resizeable</td>
<td>x</td>
<td>Whether the user can resize the column.</td>
</tr>
<tr>
<td>SlideLeft</td>
<td>x</td>
<td><em>(exp)</em> Whether the column moves left to fill in space.</td>
</tr>
<tr>
<td>SlideUp</td>
<td>x</td>
<td><em>(exp)</em> How the column moves up to fill in space.</td>
</tr>
<tr>
<td>TabSequence</td>
<td>x</td>
<td>The position of the column in the tab order.</td>
</tr>
<tr>
<td>Tag</td>
<td>x</td>
<td><em>(exp)</em> The tag text for the column.</td>
</tr>
<tr>
<td>Type</td>
<td></td>
<td>The control’s type, which is Column.</td>
</tr>
<tr>
<td>Update</td>
<td>x</td>
<td>Whether the column is updatable.</td>
</tr>
<tr>
<td>Validation</td>
<td>x</td>
<td><em>(exp)</em> The validation expression for the column.</td>
</tr>
<tr>
<td>ValidationMsg</td>
<td>x</td>
<td><em>(exp)</em> The message displayed when validation fails.</td>
</tr>
<tr>
<td>Values (for columns)</td>
<td>x</td>
<td>The values in the column’s code table.</td>
</tr>
<tr>
<td>Visible</td>
<td>x</td>
<td><em>(exp)</em> Whether the column control is visible.</td>
</tr>
<tr>
<td>Width</td>
<td>x</td>
<td><em>(exp)</em> The width of the column.</td>
</tr>
<tr>
<td>X</td>
<td>x</td>
<td><em>(exp)</em> The x coordinate of the column.</td>
</tr>
<tr>
<td>Y</td>
<td>x</td>
<td><em>(exp)</em> The y coordinate of the column.</td>
</tr>
</tbody>
</table>

### Properties for Computed Field controls in DataWindow objects

<table>
<thead>
<tr>
<th>Property for a computed field</th>
<th>M</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccessibleDescription</td>
<td>x</td>
<td>A description of the control for use by assistive technology tools.</td>
</tr>
<tr>
<td>AccessibleName</td>
<td>x</td>
<td>A descriptive label for the control.</td>
</tr>
</tbody>
</table>
### Property for a computed field

<table>
<thead>
<tr>
<th>Property</th>
<th>M</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccessibleRole</td>
<td></td>
<td>A description of the kind of user-interface element that the control is.</td>
</tr>
<tr>
<td>Alignment</td>
<td>x</td>
<td>(expr) The alignment of the computed field’s text.</td>
</tr>
<tr>
<td>Attributes</td>
<td></td>
<td>A list of the properties of the computed field.</td>
</tr>
<tr>
<td>Background.property</td>
<td>x</td>
<td>(expr) Background settings for the computed field.</td>
</tr>
<tr>
<td>Band</td>
<td></td>
<td>The band containing the computed field.</td>
</tr>
<tr>
<td>Border</td>
<td>x</td>
<td>(expr) The type of border around the computed field.</td>
</tr>
<tr>
<td>Color</td>
<td>x</td>
<td>(expr) The text color.</td>
</tr>
<tr>
<td>ColType</td>
<td></td>
<td>The column’s datatype.</td>
</tr>
<tr>
<td>Expression</td>
<td>x</td>
<td>The expression for the computed field.</td>
</tr>
<tr>
<td>Font.property</td>
<td>x</td>
<td>(expr) Font settings for the computed field.</td>
</tr>
<tr>
<td>Format</td>
<td>x</td>
<td>(expr) The computed field’s display format.</td>
</tr>
<tr>
<td>Height</td>
<td>x</td>
<td>(expr) The height of the computed field.</td>
</tr>
<tr>
<td>Height.AutoSize</td>
<td>x</td>
<td>Whether the computed field’s height is adjusted to fit the data.edom</td>
</tr>
<tr>
<td>HideSnaked</td>
<td>x</td>
<td>Whether the control appears once per page when printing newspaper columns.</td>
</tr>
<tr>
<td>HTML.property</td>
<td>x</td>
<td>(expr) Settings for creating hyperlinks for the computed field.</td>
</tr>
<tr>
<td>Moveable</td>
<td>x</td>
<td>Whether the user can move the computed field.</td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td>The name of the computed field.</td>
</tr>
<tr>
<td>Pointer</td>
<td>x</td>
<td>(expr) The pointer image when it is over the computed field.</td>
</tr>
<tr>
<td>Resizeable</td>
<td>x</td>
<td>Whether the user can resize the computed field.</td>
</tr>
<tr>
<td>SlideLeft</td>
<td>x</td>
<td>(expr) Whether the computed field moves left to fill in space.</td>
</tr>
<tr>
<td>SlideUp</td>
<td>x</td>
<td>(expr) How the computed field moves up to fill in empty space.</td>
</tr>
<tr>
<td>Tag</td>
<td>x</td>
<td>(expr) The tag text for the computed field.</td>
</tr>
<tr>
<td>Type</td>
<td></td>
<td>The control’s type, which is Compute.</td>
</tr>
<tr>
<td>Visible</td>
<td>x</td>
<td>(expr) Whether the computed field control is visible.</td>
</tr>
<tr>
<td>Width</td>
<td>x</td>
<td>(expr) The width of the computed field.</td>
</tr>
<tr>
<td>X</td>
<td>x</td>
<td>(expr) The x coordinate of the computed field.</td>
</tr>
<tr>
<td>Y</td>
<td>x</td>
<td>(expr) The y coordinate of the computed field.</td>
</tr>
</tbody>
</table>

### Properties for Graph controls in DataWindow objects

<table>
<thead>
<tr>
<th>Property</th>
<th>M</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccessibleDescription</td>
<td>x</td>
<td>A description of the control for use by assistive technology tools.</td>
</tr>
<tr>
<td>AccessibleName</td>
<td>x</td>
<td>A descriptive label for the control.</td>
</tr>
<tr>
<td>AccessibleRole</td>
<td>x</td>
<td>A description of the kind of user-interface element that the control is.</td>
</tr>
<tr>
<td>Attributes</td>
<td></td>
<td>A list of the properties of the graph.</td>
</tr>
</tbody>
</table>
# Controls in a DataWindow and their properties

<table>
<thead>
<tr>
<th>Property for a Graph</th>
<th>M</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Axis</td>
<td></td>
<td>(exp) List of items (categories, series, or values) for the axis.</td>
</tr>
<tr>
<td>Axis.property</td>
<td></td>
<td>(exp) Properties for a graph axis.</td>
</tr>
<tr>
<td>Axis.DispAttr</td>
<td></td>
<td>(exp) Display properties for an axis (see DispAttr.fontproperty in this table).</td>
</tr>
<tr>
<td>BackColor</td>
<td></td>
<td>(exp) The background color of the graph.</td>
</tr>
<tr>
<td>Band</td>
<td></td>
<td>The band containing the graph.</td>
</tr>
<tr>
<td>Border</td>
<td></td>
<td>(exp) The type of border around the graph.</td>
</tr>
<tr>
<td>Category</td>
<td></td>
<td>(exp) List of categories for the axis (see Axis in this table).</td>
</tr>
<tr>
<td>Category.property</td>
<td></td>
<td>(exp) Properties for the Category axis (see Axis.property in this table).</td>
</tr>
<tr>
<td>Category.DispAttr</td>
<td></td>
<td>(exp) Display properties for the Category axis (see DispAttr.fontproperty in this table).</td>
</tr>
<tr>
<td>Color</td>
<td></td>
<td>(exp) The text color.</td>
</tr>
<tr>
<td>Depth</td>
<td></td>
<td>(exp) The depth of a 3D graph.</td>
</tr>
<tr>
<td>DispAttr.fontproperty</td>
<td></td>
<td>Font settings for various components of the graph.</td>
</tr>
<tr>
<td>Elevation</td>
<td></td>
<td>(exp) The elevation of a 3D graph.</td>
</tr>
<tr>
<td>GraphType</td>
<td></td>
<td>(exp) The type of graph (pie, bar, and so on).</td>
</tr>
<tr>
<td>Height</td>
<td></td>
<td>(exp) The height of the graph.</td>
</tr>
<tr>
<td>HideSnaked</td>
<td></td>
<td>Whether the control appears once per page when printing newspaper columns.</td>
</tr>
<tr>
<td>Legend</td>
<td></td>
<td>(exp) The location of the legend.</td>
</tr>
<tr>
<td>Legend.DispAttr.fontproperty</td>
<td></td>
<td>(exp) Display properties for the legend.</td>
</tr>
<tr>
<td>Moveable</td>
<td></td>
<td>Whether the user can move the graph.</td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td>The name of the graph control.</td>
</tr>
<tr>
<td>OverlapPercent</td>
<td></td>
<td>(exp) The overlap between data markers in different series.</td>
</tr>
<tr>
<td>Perspective</td>
<td></td>
<td>(exp) The distance of the graph from the front of the window.</td>
</tr>
<tr>
<td>Pie.DispAttr.fontproperty</td>
<td></td>
<td>(exp) Display properties for the pie slice labels.</td>
</tr>
<tr>
<td>PlotNullData</td>
<td></td>
<td>Whether a continuous line is drawn in a line graph when there is no data.</td>
</tr>
<tr>
<td>Pointer</td>
<td></td>
<td>(exp) The pointer image when it is over the graph.</td>
</tr>
<tr>
<td>Range</td>
<td></td>
<td>The rows in the DataWindow that are included in the graph.</td>
</tr>
<tr>
<td>Resizeable</td>
<td></td>
<td>Whether the user can resize the graph.</td>
</tr>
<tr>
<td>Rotation</td>
<td></td>
<td>(exp) The left-to-right rotation of a 3D graph.</td>
</tr>
<tr>
<td>Series</td>
<td></td>
<td>(exp) List of series for the axis (see Axis in the table).</td>
</tr>
<tr>
<td>Series.property</td>
<td></td>
<td>(exp) Properties for the Series axis (see Axis.property in this table).</td>
</tr>
<tr>
<td>Series.DispAttr</td>
<td></td>
<td>(exp) Display properties for the Series axis (see DispAttr.fontproperty in this table).</td>
</tr>
<tr>
<td>ShadeColor</td>
<td></td>
<td>(exp) The color of the back edge for 3D data markers.</td>
</tr>
<tr>
<td>SizeToDisplay</td>
<td></td>
<td>(exp) Whether to size the graph to the display area.</td>
</tr>
<tr>
<td>SlideLeft</td>
<td></td>
<td>(exp) Whether the graph moves left to fill in empty space.</td>
</tr>
</tbody>
</table>
CHAPTER 3  DataWindow Object Properties

<table>
<thead>
<tr>
<th>Property for a Graph</th>
<th>M</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SlideUp</td>
<td>x</td>
<td>(exp) How the graph moves up to fill in empty space.</td>
</tr>
<tr>
<td>Spacing</td>
<td>x</td>
<td>(exp) The gap between categories.</td>
</tr>
<tr>
<td>Tag</td>
<td>x</td>
<td>(exp) The tag text for the graph.</td>
</tr>
<tr>
<td>Title</td>
<td>x</td>
<td>(exp) The graph’s title.</td>
</tr>
<tr>
<td>Title.DispAttr.fontproperty</td>
<td>x</td>
<td>(exp) Display properties for the title.</td>
</tr>
<tr>
<td>Type</td>
<td></td>
<td>The control’s type, which is graph.</td>
</tr>
<tr>
<td>Values</td>
<td>x</td>
<td>(exp) List of values for the axis (see Axis in the table).</td>
</tr>
<tr>
<td>Values.property</td>
<td>x</td>
<td>(exp) Properties for the Values axis (see Axis.property in the table).</td>
</tr>
<tr>
<td>Values.DispAttr</td>
<td>x</td>
<td>(exp) Display properties for the Values axis (see DispAttr.fontproperty in the table).</td>
</tr>
<tr>
<td>Visible</td>
<td>x</td>
<td>(exp) Whether the graph control is visible.</td>
</tr>
<tr>
<td>Width</td>
<td>x</td>
<td>(exp) The width of the graph.</td>
</tr>
<tr>
<td>X</td>
<td>x</td>
<td>(exp) The x coordinate of the graph.</td>
</tr>
<tr>
<td>Y</td>
<td>x</td>
<td>(exp) The y coordinate of the graph.</td>
</tr>
</tbody>
</table>

Properties for GroupBox controls in DataWindow objects

<table>
<thead>
<tr>
<th>Property for a GroupBox</th>
<th>M</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccessibleDescription</td>
<td>x</td>
<td>A description of the control for use by assistive technology tools.</td>
</tr>
<tr>
<td>AccessibleName</td>
<td>x</td>
<td>A descriptive label for the control.</td>
</tr>
<tr>
<td>AccessibleRole</td>
<td>x</td>
<td>A description of the kind of user-interface element that the control is.</td>
</tr>
<tr>
<td>Attributes</td>
<td></td>
<td>A list of the properties of the GroupBox control.</td>
</tr>
<tr>
<td>Background.property</td>
<td>x</td>
<td>(exp) Background settings for the GroupBox control.</td>
</tr>
<tr>
<td>Band</td>
<td></td>
<td>The band containing the GroupBox control.</td>
</tr>
<tr>
<td>Border</td>
<td>x</td>
<td>(exp) Border style: 2 (box), 5 (3D lowered), 6 (3D raised).</td>
</tr>
<tr>
<td>Color</td>
<td>x</td>
<td>(exp) The text color.</td>
</tr>
<tr>
<td>Font.property</td>
<td>x</td>
<td>(exp) Font settings for the text.</td>
</tr>
<tr>
<td>Height</td>
<td>x</td>
<td>(exp) The height of the GroupBox control.</td>
</tr>
<tr>
<td>HideSnaked</td>
<td>x</td>
<td>Whether the GroupBox control appears once per page when printing newspaper columns.</td>
</tr>
<tr>
<td>Moveable</td>
<td>x</td>
<td>Whether the user can move the GroupBox control.</td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td>The name of the GroupBox control.</td>
</tr>
<tr>
<td>Pointer</td>
<td>x</td>
<td>(exp) The pointer image when it is over the GroupBox control.</td>
</tr>
<tr>
<td>Resizeable</td>
<td>x</td>
<td>Whether the user can resize the GroupBox control.</td>
</tr>
<tr>
<td>SlideLeft</td>
<td>x</td>
<td>(exp) Whether the GroupBox control moves left to fill in empty space.</td>
</tr>
<tr>
<td>SlideUp</td>
<td>x</td>
<td>(exp) How the GroupBox control moves up to fill in empty space.</td>
</tr>
<tr>
<td>Tag</td>
<td>x</td>
<td>(exp) The tag text for the GroupBox control.</td>
</tr>
</tbody>
</table>
**Controls in a DataWindow and their properties**

<table>
<thead>
<tr>
<th>Property for a GroupBox</th>
<th>M</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Text</td>
<td>x</td>
<td>(exp) The displayed text.</td>
</tr>
<tr>
<td>Type</td>
<td></td>
<td>The control’s type, which is GroupBox.</td>
</tr>
<tr>
<td>Visible</td>
<td>x</td>
<td>(exp) Whether the GroupBox control is visible.</td>
</tr>
<tr>
<td>Width</td>
<td>x</td>
<td>(exp) The width of the GroupBox control.</td>
</tr>
<tr>
<td>X</td>
<td>x</td>
<td>(exp) The x coordinate of the GroupBox control.</td>
</tr>
<tr>
<td>Y</td>
<td>x</td>
<td>(exp) The y coordinate of the GroupBox control.</td>
</tr>
</tbody>
</table>

**Properties for the Group keyword**

You use these properties when generating DataWindow source code with the DataWindowSyntaxFromSql method.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>NewPage (Group keywords)</td>
<td>Whether a change in a group column’s value causes a page break.</td>
</tr>
<tr>
<td>ResetPageCount</td>
<td>Whether a new value in a group column restarts page numbering.</td>
</tr>
</tbody>
</table>

**Properties for InkPicture controls in DataWindow objects**

<table>
<thead>
<tr>
<th>Property for an InkPicture</th>
<th>M</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>BackImage</td>
<td></td>
<td>The column containing the background image for the InkPicture.</td>
</tr>
<tr>
<td>Band</td>
<td></td>
<td>The band containing the InkPicture.</td>
</tr>
<tr>
<td>Border</td>
<td>x</td>
<td>(exp) The type of border around the InkPicture.</td>
</tr>
<tr>
<td>Enabled</td>
<td>x</td>
<td>(exp) Whether the control is enabled.</td>
</tr>
<tr>
<td>Height</td>
<td>x</td>
<td>(exp) The height of the InkPicture.</td>
</tr>
<tr>
<td>Ink.property</td>
<td>x</td>
<td>(exp) Attributes of the ink in the InkPicture.</td>
</tr>
<tr>
<td>InkPic.property</td>
<td>x</td>
<td>(exp) Properties that specify the behavior of the InkPicture.</td>
</tr>
<tr>
<td>KeyClause</td>
<td>x</td>
<td>(exp) The key clause used when retrieving the blob.</td>
</tr>
<tr>
<td>Moveable</td>
<td>x</td>
<td>Whether the user can move the InkPicture.</td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td>The name of the InkPicture control.</td>
</tr>
<tr>
<td>Pointer</td>
<td>x</td>
<td>(exp) The pointer image when it is over the InkPicture.</td>
</tr>
<tr>
<td>Resizeable</td>
<td>x</td>
<td>Whether the user can resize the InkPicture.</td>
</tr>
<tr>
<td>SlideLeft</td>
<td>x</td>
<td>(exp) Whether the InkPicture moves left to fill in empty space.</td>
</tr>
<tr>
<td>Table (for InkPicture and TableBlobs)</td>
<td>x</td>
<td>(exp) The table that contains large binary columns used in the control.</td>
</tr>
<tr>
<td>Tag</td>
<td>x</td>
<td>(exp) The tag text for the InkPicture.</td>
</tr>
<tr>
<td>Visible</td>
<td>x</td>
<td>(exp) Whether the InkPicture control is visible.</td>
</tr>
<tr>
<td>Width</td>
<td>x</td>
<td>(exp) The width of the InkPicture.</td>
</tr>
</tbody>
</table>
CHAPTER 3  DataWindow Object Properties

### Properties for Line controls in DataWindow objects

<table>
<thead>
<tr>
<th>Property for a Line</th>
<th>M</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
<td></td>
<td>A list of the properties of the line.</td>
</tr>
<tr>
<td>Background.property</td>
<td>x</td>
<td>(exp) Background settings for the line.</td>
</tr>
<tr>
<td>Band</td>
<td></td>
<td>The band containing the line.</td>
</tr>
<tr>
<td>HideSnaked</td>
<td>x</td>
<td>Whether the control appears once per page when printing newspaper columns.</td>
</tr>
<tr>
<td>Moveable</td>
<td>x</td>
<td>Whether the user can move the line.</td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td>The name of the line control.</td>
</tr>
<tr>
<td>Pen.property</td>
<td>x</td>
<td>(exp) Appearance settings of the line.</td>
</tr>
<tr>
<td>Pointer</td>
<td>x</td>
<td>(exp) The pointer image when it is over the line.</td>
</tr>
<tr>
<td>Resizeable</td>
<td>x</td>
<td>Whether the user can resize the line.</td>
</tr>
<tr>
<td>SlideLeft</td>
<td>x</td>
<td>(exp) Whether the line moves left to fill empty space.</td>
</tr>
<tr>
<td>SlideUp</td>
<td>x</td>
<td>(exp) How the line moves up to fill empty space.</td>
</tr>
<tr>
<td>Tag</td>
<td>x</td>
<td>(exp) The tag text for the line.</td>
</tr>
<tr>
<td>Type</td>
<td></td>
<td>The control’s type, which is Line.</td>
</tr>
<tr>
<td>Visible</td>
<td>x</td>
<td>(exp) Whether the Line control is visible.</td>
</tr>
<tr>
<td>X1, X2</td>
<td>x</td>
<td>(exp) The x coordinate of each end of the line.</td>
</tr>
<tr>
<td>Y1, Y2</td>
<td>x</td>
<td>(exp) The y coordinate of each end of the line.</td>
</tr>
</tbody>
</table>

### Properties for OLE Object controls in DataWindow objects

<table>
<thead>
<tr>
<th>Property for OLE Object control</th>
<th>M</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Activation</td>
<td>x</td>
<td>The way the OLE Object control is activated.</td>
</tr>
<tr>
<td>Attributes</td>
<td></td>
<td>A list of the properties of the OLE Object control.</td>
</tr>
<tr>
<td>Band</td>
<td></td>
<td>The band containing the OLE Object control.</td>
</tr>
<tr>
<td>BinaryIndex</td>
<td></td>
<td>An internal pointer.</td>
</tr>
<tr>
<td>Border</td>
<td>x</td>
<td>(exp) The type of border around the OLE Object control.</td>
</tr>
<tr>
<td>ClientName</td>
<td>x</td>
<td>The name of the OLE client in the server window.</td>
</tr>
<tr>
<td>ContentsAllowed</td>
<td>x</td>
<td>Whether the control can be embedded, linked, or both.</td>
</tr>
<tr>
<td>DisplayType</td>
<td>x</td>
<td>Whether the control displays an icon or contents.</td>
</tr>
<tr>
<td>GroupBy</td>
<td>x</td>
<td>(exp) The grouping columns for the transferred data.</td>
</tr>
<tr>
<td>Height</td>
<td>x</td>
<td>(exp) The height of the OLE Object control.</td>
</tr>
</tbody>
</table>
Controls in a DataWindow and their properties

<table>
<thead>
<tr>
<th>Property for OLE Object control</th>
<th>M</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>HideSnaked</td>
<td>x</td>
<td>Whether the control appears once per page when printing newspaper columns.</td>
</tr>
<tr>
<td>LinkUpdateOptions</td>
<td>x</td>
<td>How a linked control is updated.</td>
</tr>
<tr>
<td>Moveable</td>
<td>x</td>
<td>Whether the user can move the OLE Object control.</td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td>The name of the OLE Object control.</td>
</tr>
<tr>
<td>Pointer</td>
<td>x</td>
<td>(exp) The pointer image when it is over the control.</td>
</tr>
<tr>
<td>Range</td>
<td></td>
<td>Method for choosing the rows transferred to the OLE control.</td>
</tr>
<tr>
<td>Resizeable</td>
<td>x</td>
<td>Whether the user can resize the OLE Object control.</td>
</tr>
<tr>
<td>SizeToDisplay</td>
<td>x</td>
<td>(exp) Whether the OLE Object control is automatically sized to the display area.</td>
</tr>
<tr>
<td>SlideLeft</td>
<td>x</td>
<td>(exp) Whether the control moves left to fill in space.</td>
</tr>
<tr>
<td>SlideUp</td>
<td>x</td>
<td>(exp) How the control moves up to fill in space.</td>
</tr>
<tr>
<td>Tag</td>
<td>x</td>
<td>(exp) The tag text for the control.</td>
</tr>
<tr>
<td>Target</td>
<td>x</td>
<td>(exp) The columns or expressions whose data you want to transfer to the OLE Object control.</td>
</tr>
<tr>
<td>Type</td>
<td></td>
<td>The control’s type, which is OLE.</td>
</tr>
<tr>
<td>Visible</td>
<td>x</td>
<td>(exp) Whether the control is visible.</td>
</tr>
<tr>
<td>Width</td>
<td>x</td>
<td>(exp) The width of the control.</td>
</tr>
<tr>
<td>X</td>
<td>x</td>
<td>(exp) The x coordinate of the control.</td>
</tr>
<tr>
<td>Y</td>
<td>x</td>
<td>(exp) The y coordinate of the control.</td>
</tr>
</tbody>
</table>

Properties for Oval, Rectangle, and RoundRectangle controls in DataWindow objects

<table>
<thead>
<tr>
<th>Property</th>
<th>M</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
<td></td>
<td>A list of the properties of the control.</td>
</tr>
<tr>
<td>Background.property</td>
<td>x</td>
<td>(exp) Background settings for the control.</td>
</tr>
<tr>
<td>Band</td>
<td></td>
<td>The band containing the control.</td>
</tr>
<tr>
<td>Brush.property</td>
<td>x</td>
<td>(exp) Settings for fill pattern and color.</td>
</tr>
<tr>
<td>Height</td>
<td>x</td>
<td>(exp) The height of the control.</td>
</tr>
<tr>
<td>HideSnaked</td>
<td>x</td>
<td>Whether the control appears once per page when printing newspaper columns.</td>
</tr>
<tr>
<td>Moveable</td>
<td>x</td>
<td>Whether the user can move the control.</td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td>The name of the control.</td>
</tr>
<tr>
<td>Pen.property</td>
<td>x</td>
<td>(exp) Appearance settings of the control.</td>
</tr>
<tr>
<td>Pointer</td>
<td>x</td>
<td>(exp) The pointer image when it is over the control.</td>
</tr>
<tr>
<td>Resizeable</td>
<td>x</td>
<td>Whether the user can resize the control.</td>
</tr>
<tr>
<td>SlideLeft</td>
<td>x</td>
<td>(exp) Whether the control moves left to fill empty space.</td>
</tr>
<tr>
<td>SlideUp</td>
<td>x</td>
<td>(exp) How the control moves up to fill empty space.</td>
</tr>
</tbody>
</table>
CHAPTER 3  DataWindow Object Properties

Additional properties for RoundRectangle controls in DataWindow objects

Properties for Oval, Rectangle, and RoundRectangle controls in DataWindow objects also apply to RoundRectangle controls.

<table>
<thead>
<tr>
<th>Property</th>
<th>M</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>EllipseHeight</td>
<td>x</td>
<td>(exp) The radius of the vertical part of the rounded corner.</td>
</tr>
<tr>
<td>EllipseWidth</td>
<td>x</td>
<td>(exp) The radius of the horizontal part of the rounded corner.</td>
</tr>
</tbody>
</table>

Properties for Picture controls in DataWindow objects

<table>
<thead>
<tr>
<th>Property for a Picture</th>
<th>M</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccessibleDescription</td>
<td>x</td>
<td>A description of the control for use by assistive technology tools.</td>
</tr>
<tr>
<td>AccessibleName</td>
<td>x</td>
<td>A descriptive label for the control.</td>
</tr>
<tr>
<td>AccessibleRole</td>
<td>x</td>
<td>A description of the kind of user-interface element that the control is.</td>
</tr>
<tr>
<td>Attributes</td>
<td>x</td>
<td>A list of the properties of the picture.</td>
</tr>
<tr>
<td>Band</td>
<td>x</td>
<td>The band containing the picture.</td>
</tr>
<tr>
<td>Border</td>
<td>x</td>
<td>(exp) The type of border around the picture.</td>
</tr>
<tr>
<td>Filename</td>
<td>x</td>
<td>(exp) The file containing the picture.</td>
</tr>
<tr>
<td>Height</td>
<td>x</td>
<td>(exp) The height of the picture.</td>
</tr>
<tr>
<td>HideSnaked</td>
<td>x</td>
<td>Whether the control appears once per page when printing newspaper columns.</td>
</tr>
<tr>
<td>HTML..property</td>
<td>x</td>
<td>(exp) Settings for creating a hyperlink for the picture.</td>
</tr>
<tr>
<td>Invert</td>
<td>x</td>
<td>(exp) Whether the colors are displayed inverted.</td>
</tr>
<tr>
<td>Moveable</td>
<td>x</td>
<td>Whether the user can move the picture.</td>
</tr>
<tr>
<td>Name</td>
<td>x</td>
<td>The name of the picture control.</td>
</tr>
<tr>
<td>Pointer</td>
<td>x</td>
<td>(exp) The pointer image when it is over the picture.</td>
</tr>
<tr>
<td>Resizeable</td>
<td>x</td>
<td>Whether the user can resize the picture.</td>
</tr>
<tr>
<td>SlideLeft</td>
<td>x</td>
<td>(exp) Whether the picture moves left to fill in empty space.</td>
</tr>
<tr>
<td>SlideUp</td>
<td>x</td>
<td>(exp) How the picture moves up to fill in empty space.</td>
</tr>
<tr>
<td>Tag</td>
<td>x</td>
<td>(exp) The tag text for the picture.</td>
</tr>
<tr>
<td>Type</td>
<td></td>
<td>The control’s type, which is picture.</td>
</tr>
</tbody>
</table>
### Controls in a DataWindow and their properties

<table>
<thead>
<tr>
<th>Property for a Picture</th>
<th>M</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visible</td>
<td>x</td>
<td>(exp) Whether the picture control is visible.</td>
</tr>
<tr>
<td>Width</td>
<td>x</td>
<td>(exp) The width of the picture.</td>
</tr>
<tr>
<td>X</td>
<td>x</td>
<td>(exp) The x coordinate of the picture.</td>
</tr>
<tr>
<td>Y</td>
<td>x</td>
<td>(exp) The y coordinate of the picture.</td>
</tr>
</tbody>
</table>

### Properties for Report controls in DataWindow objects

<table>
<thead>
<tr>
<th>Property for a Report</th>
<th>M</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
<td></td>
<td>A list of the properties of the report.</td>
</tr>
<tr>
<td>Band</td>
<td></td>
<td>The band containing the report.</td>
</tr>
<tr>
<td>Border</td>
<td>x</td>
<td>(exp) The type of border around the report.</td>
</tr>
<tr>
<td>Criteria</td>
<td>x</td>
<td>The search condition of the WHERE clause that relates the report to the main DataWindow.</td>
</tr>
<tr>
<td>DataObject</td>
<td>x</td>
<td>The name of the DataWindow that is the nested report.</td>
</tr>
<tr>
<td>Height</td>
<td></td>
<td>(exp) The height of the report.</td>
</tr>
<tr>
<td>Height.AutoScale</td>
<td>x</td>
<td>Whether the height of the control will be adjusted to display all the data.</td>
</tr>
<tr>
<td>HideSnaked</td>
<td>x</td>
<td>Whether the control appears once per page when printing newspaper columns.</td>
</tr>
<tr>
<td>Moveable</td>
<td>x</td>
<td>Whether the user can move the report.</td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td>The name of the Report control.</td>
</tr>
<tr>
<td>Nest_Arguments</td>
<td>x</td>
<td>Retrieval arguments for the report.</td>
</tr>
<tr>
<td>NewPage (Report controls)</td>
<td>x</td>
<td>Whether to start the report on a new page (composite only).</td>
</tr>
<tr>
<td>Pointer</td>
<td>x</td>
<td>(exp) The pointer image when it is over the report.</td>
</tr>
<tr>
<td>Resizeable</td>
<td>x</td>
<td>Whether the user can resize the report.</td>
</tr>
<tr>
<td>SlideLeft</td>
<td>x</td>
<td>(exp) Whether the report moves left to fill in empty space.</td>
</tr>
<tr>
<td>SlideUp</td>
<td>x</td>
<td>(exp) How the report moves up to fill in empty space.</td>
</tr>
<tr>
<td>Tag</td>
<td>x</td>
<td>(exp) The tag text for the report.</td>
</tr>
<tr>
<td>Trail_Footer</td>
<td>x</td>
<td>Where to print the footer (composite only).</td>
</tr>
<tr>
<td>Type</td>
<td></td>
<td>The control’s type, which is report.</td>
</tr>
<tr>
<td>Visible</td>
<td>x</td>
<td>(exp) Whether the Report control is visible.</td>
</tr>
<tr>
<td>Width</td>
<td>x</td>
<td>(exp) The width of the report.</td>
</tr>
<tr>
<td>X</td>
<td>x</td>
<td>(exp) The x coordinate of the report.</td>
</tr>
<tr>
<td>Y</td>
<td>x</td>
<td>(exp) The y coordinate of the report.</td>
</tr>
</tbody>
</table>
### Properties for the Style keyword

You use these properties when generating DataWindow source code with the `DataWindowSyntaxFromSql` method.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detail_Bottom_Margin</td>
<td>Bottom margin of the detail area.</td>
</tr>
<tr>
<td>Detail_Top_Margin</td>
<td>Top margin of the detail area.</td>
</tr>
<tr>
<td>Header_Bottom_Margin</td>
<td>Bottom margin of the header area.</td>
</tr>
<tr>
<td>Header_Top_Margin</td>
<td>Top margin of the header area.</td>
</tr>
<tr>
<td>Horizontal_Spread</td>
<td>Horizontal space between columns in the detail area.</td>
</tr>
<tr>
<td>Left_Margin</td>
<td>The left margin of the DataWindow.</td>
</tr>
<tr>
<td>Report</td>
<td>Whether the DataWindow is a read-only report.</td>
</tr>
<tr>
<td>Type</td>
<td>The presentation style.</td>
</tr>
<tr>
<td>Vertical_Size</td>
<td>The height of the columns in the detail area.</td>
</tr>
<tr>
<td>Vertical_Spread</td>
<td>The vertical space between columns in the detail area.</td>
</tr>
</tbody>
</table>

### Properties for TableBlob controls in DataWindow objects

<table>
<thead>
<tr>
<th>Property for a TableBlob</th>
<th>M</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attributes</td>
<td></td>
<td>A list of the properties of the TableBlob.</td>
</tr>
<tr>
<td>Band</td>
<td></td>
<td>The band containing the TableBlob.</td>
</tr>
<tr>
<td>Border</td>
<td>x</td>
<td><code>(exp)</code> The type of border around the TableBlob.</td>
</tr>
<tr>
<td>ClientName</td>
<td>x</td>
<td>The name of the OLE client in the server window.</td>
</tr>
<tr>
<td>Height</td>
<td>x</td>
<td><code>(exp)</code> The height of the TableBlob.</td>
</tr>
<tr>
<td>HideSnaked</td>
<td>x</td>
<td>Whether the control appears once per page when printing newspaper columns.</td>
</tr>
<tr>
<td>ID</td>
<td></td>
<td>The number of the TableBlob.</td>
</tr>
<tr>
<td>KeyClause</td>
<td>x</td>
<td><code>(exp)</code> The key clause used when retrieving the blob.</td>
</tr>
<tr>
<td>Moveable</td>
<td>x</td>
<td>Whether the user can move the TableBlob.</td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td>The name of the TableBlob.</td>
</tr>
<tr>
<td>OLEClass</td>
<td>x</td>
<td><code>(exp)</code> The name of the TableBlob’s OLE column.</td>
</tr>
<tr>
<td>Pointer</td>
<td>x</td>
<td><code>(exp)</code> The pointer image when it is over the TableBlob.</td>
</tr>
<tr>
<td>Resizerable</td>
<td>x</td>
<td>Whether the user can resize the TableBlob.</td>
</tr>
<tr>
<td>SlideLeft</td>
<td>x</td>
<td><code>(exp)</code> Whether the TableBlob moves left to fill empty space.</td>
</tr>
<tr>
<td>SlideUp</td>
<td>x</td>
<td><code>(exp)</code> How the TableBlob moves up to fill empty space.</td>
</tr>
<tr>
<td>Tag</td>
<td>x</td>
<td><code>(exp)</code> The tag text for the control.</td>
</tr>
<tr>
<td>Template</td>
<td>x</td>
<td><code>(exp)</code> The file used to start the OLE application.</td>
</tr>
<tr>
<td>Type</td>
<td></td>
<td>The control’s type, which is TableBlob.</td>
</tr>
<tr>
<td>Visible</td>
<td>x</td>
<td><code>(exp)</code> Whether the TableBlob is visible.</td>
</tr>
</tbody>
</table>
**Controls in a DataWindow and their properties**

<table>
<thead>
<tr>
<th>Property for a TableBlob</th>
<th>M</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Width</td>
<td>x</td>
<td><em>(exp)</em> The width of the TableBlob.</td>
</tr>
<tr>
<td>X</td>
<td>x</td>
<td><em>(exp)</em> The x coordinate of the TableBlob.</td>
</tr>
<tr>
<td>Y</td>
<td>x</td>
<td><em>(exp)</em> The y coordinate of the TableBlob.</td>
</tr>
</tbody>
</table>

**Properties for Text controls in DataWindow objects**

<table>
<thead>
<tr>
<th>Property for text</th>
<th>M</th>
<th>S</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AccessibleDescription</td>
<td>x</td>
<td></td>
<td>A description of the control for use by assistive technology tools.</td>
</tr>
<tr>
<td>AccessibleName</td>
<td>x</td>
<td></td>
<td>A descriptive label for the control.</td>
</tr>
<tr>
<td>AccessibleRole</td>
<td></td>
<td></td>
<td>A description of the kind of user-interface element that the control is.</td>
</tr>
<tr>
<td>Alignment</td>
<td>x</td>
<td>x</td>
<td>The alignment of the text.</td>
</tr>
<tr>
<td>Attributes</td>
<td></td>
<td></td>
<td>A list of the properties of the text control.</td>
</tr>
<tr>
<td>Background.property</td>
<td>x</td>
<td>x</td>
<td><em>(exp)</em> Background settings for the text control.</td>
</tr>
<tr>
<td>Band</td>
<td></td>
<td></td>
<td>The band containing the text control.</td>
</tr>
<tr>
<td>Border</td>
<td>x</td>
<td>x</td>
<td><em>(exp)</em> The type of border around the text control.</td>
</tr>
<tr>
<td>Color</td>
<td>x</td>
<td>x</td>
<td><em>(exp)</em> The text color.</td>
</tr>
<tr>
<td>Fontproperty</td>
<td>x</td>
<td>x</td>
<td><em>(exp)</em> Font settings for the text.</td>
</tr>
<tr>
<td>Height</td>
<td>x</td>
<td></td>
<td><em>(exp)</em> The height of the text control.</td>
</tr>
<tr>
<td>Height.AutoSize</td>
<td>x</td>
<td></td>
<td>Whether the control’s height is adjusted to fit the data.</td>
</tr>
<tr>
<td>HideSnaked</td>
<td>x</td>
<td></td>
<td>Whether the control appears once per page when printing newspaper columns.</td>
</tr>
<tr>
<td>HTML.property</td>
<td>x</td>
<td></td>
<td><em>(exp)</em> Settings for creating a hyperlink for the text.</td>
</tr>
<tr>
<td>Moveable</td>
<td>x</td>
<td></td>
<td>Whether the user can move the text control.</td>
</tr>
<tr>
<td>Name</td>
<td></td>
<td></td>
<td>The name of the text control.</td>
</tr>
<tr>
<td>Pointer</td>
<td>x</td>
<td></td>
<td><em>(exp)</em> The pointer image when it is over the text control.</td>
</tr>
<tr>
<td>Resizeable</td>
<td>x</td>
<td></td>
<td>Whether the user can resize the text control.</td>
</tr>
<tr>
<td>SlideLeft</td>
<td>x</td>
<td></td>
<td><em>(exp)</em> Whether the text control moves left to fill space.</td>
</tr>
<tr>
<td>SlideUp</td>
<td>x</td>
<td></td>
<td><em>(exp)</em> How the text control moves up to fill empty space.</td>
</tr>
<tr>
<td>Tag</td>
<td>x</td>
<td></td>
<td><em>(exp)</em> The tag text for the text control.</td>
</tr>
<tr>
<td>Text</td>
<td>x</td>
<td></td>
<td><em>(exp)</em> The displayed text.</td>
</tr>
<tr>
<td>Type</td>
<td></td>
<td></td>
<td>The control’s type, which is Text.</td>
</tr>
<tr>
<td>Visible</td>
<td>x</td>
<td></td>
<td><em>(exp)</em> Whether the control is visible.</td>
</tr>
<tr>
<td>Width</td>
<td>x</td>
<td></td>
<td><em>(exp)</em> The width of the text control.</td>
</tr>
<tr>
<td>X</td>
<td>x</td>
<td></td>
<td><em>(exp)</em> The x coordinate of the text control.</td>
</tr>
<tr>
<td>Y</td>
<td>x</td>
<td></td>
<td><em>(exp)</em> The y coordinate of the text control.</td>
</tr>
</tbody>
</table>
Title keyword

You use this property when generating DataWindow source code with the DataWindowSyntaxFromSql method.

<table>
<thead>
<tr>
<th>Property</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Title(&quot;string&quot;)</td>
<td>The title for the DataWindow.</td>
</tr>
</tbody>
</table>

**Alphabetical list of DataWindow object properties**

The properties for DataWindow objects and controls within a DataWindow object follow in alphabetical order.

The syntax for Describe and Modify arguments is shown for all properties. The syntax forGetProperty and SetProperty is the same for all properties.

For properties of the DataWindow object itself, the syntax is:

```vbscript
propertyvalue = dwc.GetProperty ( "propertyname" );
dwc.SetProperty ( "propertyname", "propertyvalue" );
```

For properties of controls in the DataWindow object, the syntax is:

```vbscript
propertyvalue = dwc.GetProperty ( "controlname.propertyname" );
dwc.SetProperty ( "controlname.propertyname", "propertyvalue" );
```

The simple Visual Basic example shown for most properties can be used in C# by adding a semicolon to the end of each statement.

To see the properties organized by type of control or syntax keyword, see “Controls in a DataWindow and their properties” on page 145.
Accelerator

Description
The accelerator key that a user can press to select a column in the DataWindow object.

Applies to
Column controls

Syntax
Describe and Modify argument:
"columnname.Accelerator { = 'acceleratorkey' }

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columnname</td>
<td>The name of the column for which you want to get or set the accelerator key.</td>
</tr>
<tr>
<td>acceleratorkey</td>
<td>(exp) A string expression whose value is the letter that will be the accelerator key for columnname. Acceleratorkey can be a quoted DataWindow expression.</td>
</tr>
</tbody>
</table>

Usage
An accelerator key for a column allows users to select a column (change focus) with a keystroke rather than with the mouse. The user changes focus by pressing the accelerator key in combination with the Alt key.

In the painter
Select the control and set the value in the Properties window, Behavior category.

Displaying the accelerator
The column does not display the key. To let users know what key to use, you can include an underlined letter in a text control that labels the column. When you enter the text control’s label, precede the character you want underlined with an ampersand (&).

Examples

[Visual Basic]
Dim AccKey as String
dw1.SetProperty("EmpName.Accelerator","A")
AccKey = dw1.GetProperty("EmpName.Accelerator")
AccKey = dw1.Describe("EmpName.Accelerator")
dw1.Modify("EmpName.Accelerator='A'")

[C#]
string AccKey;
dw1.SetProperty("EmpName.Accelerator","A");
AccKey = dw1.GetProperty("EmpName.Accelerator");
AccKey = dw1.Describe("EmpName.Accelerator");
dw1.Modify("EmpName.Accelerator='A'");
CHAPTER 3  DataWindow Object Properties

AccessibleDescription

Description  A description of the control and/or its purpose for use by accessibility tools such as readers for visually-impaired users.

Applies to  Column, computed field, picture, text, graph, group box, and button controls

Syntax  Describe and Modify argument:

"controlname. { = 'description' }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columnname</td>
<td>The name of the control for which you want to get or set the accessible description</td>
</tr>
<tr>
<td>description</td>
<td>(exp) A string that describes the control’s purpose or appearance</td>
</tr>
</tbody>
</table>

Usage  You do not need to supply a description if the AccessibleName and AccessibleRole properties adequately describe the control, as in the case of a button with the label OK. You should provide a description for a picture or report control.

In the painter  In the Accessibility category in the Properties window, type a description in the AccessibleDescription text box.

Examples  [Visual Basic]

```
strData = dw1.Describe("b_1.AccessibleDescription")
dw1.Modify("b_1.AccessibleDescription='Scrolls to next row'")
```

AccessibleName

Description  A label that briefly describes the control for use by accessibility tools such as readers for visually-impaired users.

Applies to  Column, computed field, picture, text, graph, group box, and button controls

Syntax  Describe and Modify argument:

"controlname.AccessibleName { = 'description' }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columnname</td>
<td>The name of the control for which you want to get or set the accessible description</td>
</tr>
<tr>
<td>description</td>
<td>(exp) A string that briefly describes the control</td>
</tr>
</tbody>
</table>
AccessibleRole

Usage

The AccessibleName property is a brief description, such as the text in a button or the name of a menu item.

In the painter

In the Accessibility category in the Properties window, type a name in the AccessibleName text box.

Examples

[Visual Basic]
ls_data = dw1.Describe("b_1.AccessibleName")
dw1.Modify("b_1.AccessibleName='Next'")

AccessibleRole

Description

A description of the kind of user-interface element that the control is, for use by accessibility tools such as readers for visually-impaired users.

Applies to

Column, computed field, picture, text, graph, group box, and button controls

Syntax

Describe and Modify argument:
"controlname.AccessibleRole { = 'enumeratedvalue' }"

Parameter | Description
---|---
controlname | The name of the control for which you want to get or set the accessible description
description | (exp) A number specifying the type of AccessibleRole as a numeric value of the AccessibleRole DataWindow constant.

Usage

The description is a member of the AccessibleRole enumerated variable. The default role is defaultrole! and is used when the role cannot be determined.

Table 3-1: AccessibleRole values for DataWindow controls

<table>
<thead>
<tr>
<th>Control</th>
<th>AccessibleRole</th>
</tr>
</thead>
<tbody>
<tr>
<td>Button</td>
<td>pushbuttonrole!</td>
</tr>
<tr>
<td>Column</td>
<td>textrole!</td>
</tr>
<tr>
<td>Computed field</td>
<td>statictextrole!</td>
</tr>
<tr>
<td>Graph</td>
<td>diagramrole!</td>
</tr>
<tr>
<td>Group box</td>
<td>groupingrole!</td>
</tr>
<tr>
<td>Picture</td>
<td>graphicrole!</td>
</tr>
<tr>
<td>Text</td>
<td>statictextrole!</td>
</tr>
</tbody>
</table>

In the painter

In the Accessibility category in the Properties window, select a value in the AccessibleRole drop-down list.

Examples

[Visual Basic]
ls_data = dw1.Describe("b_1.AccessibleRole")
CHAPTER 3  DataWindow Object Properties

Action
Description  The action a user can assign to a button control.
Applies to   Button controls
Syntax      Describe and Modify argument:

"buttonname.Action { = 'value'}"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>buttonname</td>
<td>The name of the button for which you want to assign an action.</td>
</tr>
<tr>
<td>value</td>
<td>The action value assigned to the button. Values are listed in the following table.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Value</th>
<th>Action</th>
<th>Description</th>
<th>Value returned to ButtonClicked event</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>UserDefined</td>
<td>(Default) Allows for programming of the ButtonClicked and ButtonClicking events with no intervening action occurring.</td>
<td>Return code from the user’s coded event script.</td>
</tr>
<tr>
<td>1</td>
<td>Retrieve (Yield)</td>
<td>Retrieves rows from the database. Before retrieval actually occurs, option to yield is turned on. This allows the Cancel action to take effect during a long retrieve.</td>
<td>Number of rows retrieved.</td>
</tr>
<tr>
<td>2</td>
<td>Retrieve</td>
<td>Retrieves rows from the database. The option to yield is not automatically turned on.</td>
<td>Number of rows retrieved.</td>
</tr>
<tr>
<td>3</td>
<td>Cancel</td>
<td>Cancels a retrieval that has been started with the option to yield.</td>
<td>0</td>
</tr>
<tr>
<td>4</td>
<td>PageNext</td>
<td>Scrolls to the next page.</td>
<td>The row displayed at the top of the DataWindow control when the scrolling is complete or attempts to go past the first row. -1 if an error occurs.</td>
</tr>
<tr>
<td>5</td>
<td>PagePrior</td>
<td>Scrolls to the prior page.</td>
<td>The row displayed at the top of the DataWindow control when the scrolling is complete or attempts to go past the first row. -1 if an error occurs.</td>
</tr>
<tr>
<td>6</td>
<td>PageFirst</td>
<td>Scrolls to the first page.</td>
<td>1 if successful. -1 if an error occurs.</td>
</tr>
</tbody>
</table>
### Action

<table>
<thead>
<tr>
<th>Value</th>
<th>Action</th>
<th>Description</th>
<th>Value returned to ButtonClicked event</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>PageLast</td>
<td>Scrolls to the last page.</td>
<td>The row displayed at the top of the DataWindow control when the scrolling is complete or attempts to go past the first row. -1 if an error occurs.</td>
</tr>
<tr>
<td>8</td>
<td>Sort</td>
<td>Displays Sort dialog box and sorts as specified.</td>
<td>1 if successful. -1 if an error occurs.</td>
</tr>
<tr>
<td>9</td>
<td>Filter</td>
<td>Displays Filter dialog box and filters as specified.</td>
<td>Number of rows filtered. Number &lt; 0 if an error occurs.</td>
</tr>
<tr>
<td>10</td>
<td>DeleteRow</td>
<td>If button is in detail band, deletes row associated with button; otherwise, deletes the current row.</td>
<td>1 if successful. -1 if an error occurs.</td>
</tr>
<tr>
<td>11</td>
<td>AppendRow</td>
<td>Inserts row at the end.</td>
<td>Row number of newly inserted row.</td>
</tr>
<tr>
<td>12</td>
<td>InsertRow</td>
<td>If button is in detail band, inserts row using row number associated with the button; otherwise, inserts row using the current row.</td>
<td>Row number of newly inserted row.</td>
</tr>
<tr>
<td>13</td>
<td>Update</td>
<td>Saves changes to the database. If the update is successful, a COMMIT is issued. If the update fails, a ROLLBACK is issued</td>
<td>1 if successful. -1 if an error occurs.</td>
</tr>
<tr>
<td>14</td>
<td>SaveRowsAs</td>
<td>Displays Save As dialog box and saves rows in the format specified.</td>
<td>Number of rows filtered.</td>
</tr>
<tr>
<td>15</td>
<td>Print</td>
<td>Prints one copy of the DataWindow object.</td>
<td>0</td>
</tr>
<tr>
<td>16</td>
<td>Preview</td>
<td>Toggles between preview and print preview.</td>
<td>0</td>
</tr>
<tr>
<td>17</td>
<td>PreviewWithRulers</td>
<td>Toggles between rulers on and off.</td>
<td>0</td>
</tr>
<tr>
<td>18</td>
<td>QueryMode</td>
<td>Toggles between query mode on and off.</td>
<td>0</td>
</tr>
<tr>
<td>19</td>
<td>QuerySort</td>
<td>Specifies sorting criteria (forces query mode on).</td>
<td>0</td>
</tr>
<tr>
<td>20</td>
<td>QueryClear</td>
<td>Removes the WHERE clause from a query (if one was defined).</td>
<td>0</td>
</tr>
</tbody>
</table>

**Usage**

**In the painter** Select the control and set the value in the Properties window, General category.

**Examples**

```Visual Basic
Dim ActionValue as String
dw1.SetProperty("RetrieveButton.Action", "2")
ActionValue = dw1.Describe("RetrieveButton.Action")
dw1.Modify("RetrieveButton.Action = '2'")
```
CHAPTER 3 DataWindow Object Properties

[C#]
string ActionValue;
dw1.SetProperty("RetrieveButton.Action", "2"
ActionValue = dw1.Describe("RetrieveButton.Action")
dw1.Modify("RetrieveButton.Action = '2'")

Activation

Description
The way the server for the OLE object in the OLE Object control is activated. Choices include letting the user activate the object by double-clicking or putting activation under program control.

Applies to
OLE Object controls

Syntax
Describe and Modify argument:

"olecontrolname.Activation { = 'activationtype' }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>olecontrolname</td>
<td>The name of the OLE Object control for which you want to get or set the activation method.</td>
</tr>
<tr>
<td>activationtype</td>
<td>(exp) A number specifying the method of activation for the OLE object. Activationtype can be a quoted DataWindow expression. Values are:</td>
</tr>
<tr>
<td></td>
<td>0 – The object has to be activated with the Activate method.</td>
</tr>
<tr>
<td></td>
<td>1 – The user can activate the object by double-clicking on it.</td>
</tr>
<tr>
<td></td>
<td>2 – The object activates when the container gets focus.</td>
</tr>
</tbody>
</table>

Usage
In the painter Select the control and set the value in the Properties window, Options category.

Examples

[Visual Basic]
Dim ActType as String
dw1.GetProperty("OLEReport.Activation")
ActType = dw1.Describe("OLEReport.Activation")
dw1.Modify("OLEReport.Activation='2'")

[C#]
string ActType;
dw1.GetProperty("OLEReport.Activation");
ActType = dw1.Describe("OLEReport.Activation");
dw1.Modify("OLEReport.Activation='2'");
## Alignment

**Description**
The alignment of the control’s text within its borders.

**Applies to**
Column, Computed Field, and Text controls

**Syntax**
Describe and Modify argument:

```
"controlname.Alignment { = 'alignmentvalue' }"
```

DataWindowSyntaxFromSql:

```
Text ( ... Alignment = alignmentvalue ... )
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlname</td>
<td>The name of the control for which you want to get or set the alignment.</td>
</tr>
<tr>
<td>alignmentvalue</td>
<td>(exp) A number specifying the type of alignment for the text of controlname. Alignmentvalue can be a quoted DataWindow expression. Values are: 0 – (Default) Left 1 – Right 2 – Center 3 – Justified When generating DataWindow syntax with DataWindowSyntaxFromSql, the setting for Alignment applies to all text controls used as column labels.</td>
</tr>
</tbody>
</table>

**Usage**
When you select justified, the last line of text is not stretched to fill the line. Controls with only one line of text look left aligned.

**In the painter**
Select the control and set the value using the Properties window, Appearance category.

**Examples**

### Visual Basic

```vbnet
Dim AlignVal As String
WC1.SetProperty("EmpName.Alignment", "2")
AlignVal = dw1.Describe("EmpName.Alignment")
dw1.Modify("EmpName.Alignment='2'")
```

### C#

```csharp
string AlignVal;
WC1.SetProperty("EmpName.Alignment", "2");
AlignVal = dw1.Describe("EmpName.Alignment");
dw1.Modify("EmpName.Alignment='2'");
```
Arguments

Description
The retrieval arguments required by the data source. You specify retrieval arguments in the DataWindow’s SELECT statement and you provide values for the retrieval arguments when you call the Retrieve method.

Applies to
Database table for the DataWindow object
Used in DataWindow syntax.

Syntax
`Table(Arguments = ( (name1, type), (name2, type) ... ) ... )`

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>name</td>
<td>The name of the retrieval argument</td>
</tr>
</tbody>
</table>
| type      | The type of the argument:  
  - Date or a Date list  
  - DateTime or a DateTime list  
  - Number or a Number list  
  - String or a String list  
  - Time or a Time list |

Usage
**In the painter** Set the value in the SQL Select painter or Query painter.

Open the SQL Select painter by selecting Design>Data Source from the menu bar in the DataWindow painter, or create or open a query in the Query painter. Then select Design>Retrieval Arguments.

Attributes

Description
A tab-separated list of all the properties that apply to a control.

Applies to
DataWindow, Button, Column, Computed Field, Graph, GroupBox, Line, OLE, Oval, Picture, Rectangle, Report, RoundRectangle, TableBlob, and Text controls

Syntax
Describe argument:
```
"controlname.Attributes"
```

Examples

```
[Visual Basic]
Dim AttrStr as String
AttrStr = dw1.GetProperty("EmpNameText.Attributes")
AttrStr = dw1.Describe("DataWindow.Attributes")
AttrStr = dw1.Describe("EmpNameText.Attributes")
```
Axis

Description
The list of items or the expression associated with an axis of a graph. Each item is separated by a comma. You can ask for the list of categories on the Category axis, the series on the Series axis, or the values on the Values axis.

Applies to
Graph controls

Syntax
Describe and Modify argument:
"graphname.axis (= 'list')"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>graphname</td>
<td>The name of the graph within the DataWindow object for which you want to get or set the list of items for axis.</td>
</tr>
<tr>
<td>axis</td>
<td>An axis name. Values are:</td>
</tr>
<tr>
<td></td>
<td>• Category</td>
</tr>
<tr>
<td></td>
<td>• Series</td>
</tr>
<tr>
<td></td>
<td>• Values</td>
</tr>
<tr>
<td>list</td>
<td>A string listing the categories, series, or values for the graph. The content of the list depends on the axis you specify. The items in the list are separated by commas. List is quoted.</td>
</tr>
</tbody>
</table>

Usage
In the painter Select the graph control and set the value by selecting a column or expression for each axis in the Properties window, Axis category.

Examples
[Visual Basic]
ls_data = dw1.Describe("gr1.Category")
ls_data = dw1.Describe("gr1.Series")
ls_data = dw1.Describe("gr1.Values")
dw1.Modify("gr1.Series='Actual, Budget'")

[C#]
string AttrStr;
AttrStr = dw1.GetProperty("EmpNameText.Attributes");
AttrStr = dw1.Describe("DataWindow.Attributes");
AttrStr = dw1.Describe("EmpNameText.Attributes");
Axis.property

Description | Settings that control the appearance of an axis on a graph.
Applies to | Graph controls
Syntax | Describe and Modify argument:

"graphname.axis.property { = value }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>graphname</td>
<td>The name of the graph within the DataWindow object for which you want to get or set a property value for an axis.</td>
</tr>
</tbody>
</table>
| axis | An axis name. Values are:
- Category
- Series
- Values |
| property | A property for the axis. Properties and their settings are listed in the table that follows. |
| value | The value to be assigned to the property. For axis properties, value can be a quoted DataWindow expression. |

Property for Axis | Value
---|---
AutoScale | *(exp)* A boolean number specifying whether the DataWindow server scales the axis automatically. Enabled when the axis displays nonstring data. Values are:
- 0 – No, do not automatically scale the axis.
- 1 – Yes, automatically scale the axis.

DispAttr. fontproperty | *(exp)* Properties that control the appearance of the text that labels the axis divisions. For a list of font properties, see the main entry for DispAttr.fontproperty. Painter: Text category. Choose Category Axis Text, Series Axis Text, or Values Axis Text from the TextObject list, and set font properties.

DisplayEvery NLabels | *(exp)* An integer specifying which major axis divisions to label. For example, 2 means label every other tick mark. Values 0 and 1 both mean label every tick mark. If the labels are too long, they are clipped.
## Axis Property

<table>
<thead>
<tr>
<th>Property for Axis</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DropLines</td>
<td><em>(exp)</em> An integer indicating the type of drop line for the axis. Values are: 0 – None, 1 – Solid, 2 – Dash, 3 – Dot, 4 – DashDot, 5 – DashDotDot.</td>
</tr>
<tr>
<td>Frame</td>
<td><em>(exp)</em> An integer indicating the type of line used for the frame. Values are 0–5. See DropLines in this table for their meaning. Available for 3D graph types.</td>
</tr>
<tr>
<td>Label</td>
<td><em>(exp)</em> A string whose value is the axis label.</td>
</tr>
<tr>
<td>LabelDispAttr.</td>
<td><em>(exp)</em> Properties that control the appearance of the axis label. For a list of font properties, see the main entry for DispAttr.fontproperty. Painter: Text category. Choose Category Axis Label, Series Axis Label, or Values Axis Label from the TextObject list, and set font properties.</td>
</tr>
<tr>
<td>MajorDivisions</td>
<td><em>(exp)</em> An integer specifying the number of major divisions on the axis.</td>
</tr>
<tr>
<td>MajorGridLine</td>
<td><em>(exp)</em> An integer specifying the type of line for the major grid. Values are 0–5. See DropLines in this table for their meaning.</td>
</tr>
<tr>
<td>MajorTic</td>
<td><em>(exp)</em> An integer specifying the type of the major tick marks. Values are: 1 – None, 2 – Inside, 3 – Outside, 4 – Straddle.</td>
</tr>
<tr>
<td>MaximumValue</td>
<td><em>(exp)</em> A double specifying the maximum value for the axis.</td>
</tr>
<tr>
<td>MinimumValue</td>
<td><em>(exp)</em> A double specifying the minimum value for the axis.</td>
</tr>
<tr>
<td>MinorDivisions</td>
<td><em>(exp)</em> An integer specifying the number of minor divisions on the axis.</td>
</tr>
<tr>
<td>MinorGridLine</td>
<td><em>(exp)</em> An integer specifying the type of line for the minor grid. Values are 0–5. See DropLines in this table for their meaning.</td>
</tr>
<tr>
<td>MinorTic</td>
<td><em>(exp)</em> An integer specifying the type of the minor tick marks. Values are: 1 – None, 2 – Inside, 3 – Outside, 4 – Straddle.</td>
</tr>
<tr>
<td>OriginLine</td>
<td><em>(exp)</em> An integer specifying the type of origin line for the axis. Values are 0–5. See DropLines in this table for their meaning. Enabled for numeric data axes.</td>
</tr>
<tr>
<td>PrimaryLine</td>
<td><em>(exp)</em> An integer specifying the type of primary line for the axis. Values are 0–5. See DropLines in this table for their meaning.</td>
</tr>
</tbody>
</table>
### Property for Axis

<table>
<thead>
<tr>
<th>Property for Axis</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>RoundTo</strong></td>
<td>((exp)) A double specifying the value to which you want to round the axis values. Specify both a value and a unit (described next).</td>
</tr>
</tbody>
</table>
| **RoundToUnit**   | \((exp)\) An integer specifying the units for the rounding value. The units must be appropriate for the axis datatype.  
  Values are:  
  0 – Default, for an axis of any datatype  
  1 – Years, for an axis of type date or DateTime  
  2 – Months, for an axis of type date or DateTime  
  3 – Days, for an axis of type date or DateTime  
  4 – Hours, for an axis of type time or DateTime  
  5 – Minutes, for an axis of type time or DateTime  
  6 – Seconds, for an axis of type time or DateTime  
  7 – Microseconds, for an axis of type time or DateTime |
| **ScaleType**     | \((exp)\) An integer specifying the type of scale used for the axis.  
  Values are:  
  1 – Scale_Linear  
  2 – Scale_Log10  
  3 – Scale_Loge |
| **ScaleValue**    | \((exp)\) An integer specifying the scale of values on the axis.  
  Values are:  
  1 – Scale_Actual  
  2 – Scale_Cumulative  
  3 – Scale_Percentage  
  4 – Scale_CumPercent |
| **SecondaryLine** | \((exp)\) An integer specifying the type of secondary line for the axis. The line is parallel to and opposite the primary line and is usually not displayed in 2D graphs. Values are 0–5. See DropLines in this table for their meaning. |
| **ShadeBackEdge** | \((exp)\) A boolean number specifying whether the back edge of the axis is shaded.  
  Values are:  
  0 – No, the back edge is not shaded  
  1 – Yes, the back edge is shaded |
| **Sort**          | \((exp)\) An integer specifying the way the axis values should be sorted. (Does not apply to the Values axis.)  
  Values are:  
  0 – Unsorted  
  1 – Ascending  
  2 – Descending |
## BackColor

**Description**  
The background color of a graph in a DataWindow.

**Applies to**  
Graph controls

**Syntax**  
Describe and Modify argument:

```
"graphname.BackColor { = long }
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>graphname</td>
<td>The graph whose background color you want to get or set.</td>
</tr>
<tr>
<td>long</td>
<td><em>(exp)</em> A long expression specifying the color (red, green, and blue values) to be used as the graph’s background color. <em>Long</em> can be a quoted DataWindow expression.</td>
</tr>
</tbody>
</table>

**Usage**  

**In the painter**  
Select the graph control and set the value in the Properties window, General category.

**Examples**

[Visual Basic]
```
Dim PropVal As String
PropVal = dw1.GetProperty("Graph1.Category.AutoScale")
dw1.SetProperty("Category.LabelDispAttr.Alignment", "2")
PropVal = dw1.Describe("Graph1.Category.AutoScale")
dw1.Modify("Graph1.Series.AutoScale=0")
dw1.Modify("Graph1.Values.Label='Cities'")
dw1.Modify("Graph1.Category.LabelDispAttr.Alignment=2")
```

[C#]
```
string PropVal;
PropVal = dw1.GetProperty("Graph1.Category.AutoScale");
dw1.SetProperty("Category.LabelDispAttr.Alignment", "2");
PropVal = dw1.Describe("Graph1.Category.AutoScale");
dw1.Modify("Graph1.Series.AutoScale=0");
dw1.Modify("Graph1.Values.Label='Cities'");
dw1.Modify("Graph1.Category.LabelDispAttr.Alignment=2");
```
CHAPTER 3  DataWindow Object Properties

Examples

[Visual Basic]
Dim BColor As String
dw1.SetProperty("Graph1.BackColor", "250")
BColor = dw1.Describe("Graph1.BackColor")
dw1.Modify("Graph1.BackColor=250")

[C#]
string BColor;
dw1.SetProperty("Graph1.BackColor", "250");
BColor = dw1.Describe("Graph1.BackColor");
dw1.Modify("Graph1.BackColor=250");

Background.property

Description  Settings for the color and transparency of a control.
Applies to Button, Column, Computed Field, GroupBox, Line, Oval, Rectangle, RoundRectangle, and Text controls
Syntax Describe and Modify argument:
"controlname.Background.property { = 'value' }"

DataWindowSyntaxFromSql:

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlname</td>
<td>The control whose Background properties you want to get or set.</td>
</tr>
<tr>
<td></td>
<td>When generating DataWindow syntax with DataWindowSyntaxFromSql, the Background settings apply to all columns or all text controls.</td>
</tr>
<tr>
<td>property</td>
<td>A property that applies to the background of a control, as listed in the Property table below.</td>
</tr>
<tr>
<td>value</td>
<td>Values for the properties are shown below. Value can be a quoted DataWindow expression.</td>
</tr>
</tbody>
</table>
Background.property

<table>
<thead>
<tr>
<th>Property for Background</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td><em>(exp)</em> A long expression specifying the color (the red, green, and blue values) to be used as the control’s background color.</td>
</tr>
</tbody>
</table>
| Mode                    | *(exp)* A number expression specifying the mode of the background of *controlname*. Values are:  
0 – Make the control’s background opaque.  
1 – Make the control’s background transparent. |

Usage

**In the painter** Select the control and set the value in the Properties window, Appearance category.

When you choose a Brush Hatch fill pattern other than Solid for an Oval, Rectangle, or RoundRectangle control, the Background Color and the Brush Color are used for the pattern colors.

**Background color of a button** The Background.Color property is not supported on Windows XP by default because the current XP theme controls the appearance of the button. Set the ShowBackColorOnXP property of the DataWindow object to force the color change to take effect.

**Background color of a line** The background color of a line is the color that displays between the segments of the line when the pen style is not solid.

**Transparent background** If Background.Mode is transparent (1), Background.Color is ignored.

**DropDownDataWindows and GetChild** When you set Background.Color and Background.Mode for a column with a DropDownDataWindow, references to the DropDownDataWindow become invalid. Call GetChild again after changing these properties to obtain a valid reference.

Examples

```Visual Basic
Dim BGProperty As String
dw1.SetProperty("Oval2.Background.Color", "RGB(255, 0, 128)")
BGProperty = dw1.Describe("Oval2.Background.Color")
dw1.Modify("EmpName.Background.Color='11665407'")
BGProperty = dw1.Describe("EmpName.Background.Mode")
dw1.Modify("EmpName.Background.Mode='1'")
dw1.Modify("RndRect1.Background.Mode='0'")
```
CHAPTER 3 DataWindow Object Properties

[C#]
string BGProperty;
dw1.SetProperty("Oval2.Background.Color", "RGB(255, 0, 128)");
BGProperty = dw1.Describe("Oval2.Background.Color");
dw1.Modify("EmpName.Background.Color='11665407'");
BGProperty = dw1.Describe("EmpName.Background.Mode");
dw1.Modify("EmpName.Background.Mode='1'");
dw1.Modify("RndRect1.Background.Mode='0'");

BackImage
Description The column that contains the background image for an InkPicture control in a DataWindow.
Applies to InkPicture controls
Syntax Describe and Modify argument:
"BackImage{ = colname }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>colname</td>
<td>A string value specifying the name of the long binary column that contains the background image for the control.</td>
</tr>
</tbody>
</table>

Usage In the painter Select the InkPicture control and set the value in the Properties window, Definition category. The image format can be JPEG, GIF, BMP, or ICO. If you change the image, call the Retrieve method to force the DataWindow to retrieve the new image.

Examples

[Visual Basic]
DWBand = dw1.Describe("inkPic1.Band")

[C#]
DWBand = dw1.Describe("inkPic1.Band");
Band

Description
The band or layer in the DataWindow object that contains the control. The returned text is one of the following, where # is the level number of a group: detail, footer, header, header.#, summary, trailer.#, tree.level.#, foreground, background.

Applies to
Button, Column, Computed Field, Graph, GroupBox, Line, OLE, Oval, Picture, Rectangle, Report, RoundRectangle, TableBlob, and Text controls

Syntax
Describe and Modify argument:
"controlname.Band"

Usage
In the painter Select the control and set the value in the Properties window, Layout category. When the control’s layer is Band, you can drag the control into another band.

Examples
[Visual Basic]
DWBand = dw1.GetProperty("emp_title.Band")
DWBand = dw1.Describe("emp_title.Band")

[C#]
DWBand = dw1.GetProperty("emp_title.Band");
DWBand = dw1.Describe("emp_title.Band");
Bandname.property

Description: Settings for the color, size, and pointer of a band in the DataWindow object.

Applies to: DataWindows

Syntax: Describe and Modify argument:

"DataWindow.bandname{.#}.property { = value }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| bandname  | The identifier of a band in the DataWindow object. Values are:  
• Detail  
• Footer  
• Summary  
• Header  
• Trailer  
• Tree.Level |
| #         | The number of the group or TreeView level you want when bandname is Header, Trailer, or Tree.Level. The group must exist. |
| property  | A property that applies to the band, as listed in the table below. |
| value     | Values for the properties are shown in the following table. |

<table>
<thead>
<tr>
<th>Property for Bandname</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td>(exp) A long specifying the color (the red, green, and blue values) to be used as the band’s background color. Value can be a quoted DataWindow expression. Painter: General category.</td>
</tr>
</tbody>
</table>
| Height                | An integer specifying the height of the detail area in the unit of measure specified for the DataWindow. Painter: General category.  
For another way of setting the height of the detail band, see the SetDetailHeight method. |
### Property for Bandname

<table>
<thead>
<tr>
<th>Property for Bandname</th>
<th>Value</th>
</tr>
</thead>
</table>
| Height.AutoSize       | Allows the band to grow to display a row, picture, or nested report without cutting off any of its content. In the detail band, selecting this property sets the minimum height for all rows to the size specified by the Height property for the band. Values are:  
  No – Fixes the band height to the size set for the Height property of the band.  
  Yes – Adjusts the band height to accommodate the full content of a row or the controls in the band. However, the band height cannot be reduced below the value set for the Height property of the band.  
This property can be especially useful to set on the detail band when it contains rows with a text column that you want to display without cutting off any of the text. The height of the detail band must not grow larger than a page, except when it contains nested DataWindows with the Report.Height.AutoSize property set to Yes.  
You can set this property on individual columns and controls as well as on the band itself. For more information, see the HeightAutoSize property for DataWindow objects.  
There are some limitations on the use of this property:  
- The Height.AutoSize property is not supported on DataWindows with Graph or Label presentation styles.  
- Nested report overflow to the next page is supported in detail bands only.  
- Bands cannot be autosized if autosizing would preclude the display of at least one detail band row per page. |

| Pointer                | (exp) A string specifying a value of the Pointer enumerated datatype or the name of a cursor file (.CUR) to be used for the pointer. See the SetPointer method for a list of Pointer values. Pointername can be a quoted DataWindow expression. This property is not supported in Web DataWindows. Painter: Pointer category. |

| Suppress               | A boolean that lets you suppress group headers after page breaks. You can set this property on group header bands only. When a group listing straddles a page break, all group headers for which you set this property will be suppressed. The suppressed headers do not display at the top of the page. However, if the page break coincides with the start of a new group, only headers above the current group header can be suppressed. Values are:  
  No – Does not suppress group headers.  
  Yes – Suppresses group headers. Painter: General category when a group header band is selected. |

| Usage                  | **In the painter** Select the band by clicking the gray divider for the band. Set the value in the Properties window. |
Examples

**[Visual Basic]**

Dim PropVal As String
PropVal = dw1.GetProperty("DataWindow.Detail.Height")
dw1.SetProperty("DataWindow.Detail.Pointer", "hand.cur")

PropVal = dw1.Describe("DataWindow.Detail.Height")
PropVal = dw1.Describe("DataWindow.Detail.Height.AutoSize")
dw1.Modify("DataWindow.Detail.Pointer='hand.cur'")

dw1.Modify("DataWindow.Detail.Pointer=' ~Cross!~ ~t if(emp_status='a-', ~'HourGlass!~', ~'Cross!~')'")

dw1.Modify("DataWindow.Footer.Height=250")
ColorVal = RGB(200, 200, 500)
dw1.Modify("DataWindow.Trailer.2.Height=500")
dw1.Modify("DataWindow.Summary.Pointer='total.cur'")

**[C#]**

string PropVal;
PropVal = dw1.GetProperty("DataWindow.Detail.Height");
dw1.SetProperty("DataWindow.Detail.Pointer", "hand.cur");

PropVal = dw1.Describe("DataWindow.Detail.Height");
PropVal = dw1.Describe("DataWindow.Detail.Height.AutoSize");
dw1.Modify("DataWindow.Detail.Pointer='hand.cur'");

dw1.Modify("DataWindow.Detail.Pointer=' ~Cross!~ ~t if(emp_status='a-', ~'HourGlass!~', ~'Cross!~')'")

dw1.Modify("DataWindow.Footer.Height=250");
ColorVal = RGB(200, 200, 500);
dw1.Modify("DataWindow.Trailer.2.Height=500");
dw1.Modify("DataWindow.Summary.Pointer='total.cur'");
**Bandname.Text**

**Description**
(RichText presentation style only—not in DataWindow .NET) The rich text content of the specified band as an ASCII string.

**Applies to**
DataWindows in the RichText presentation style

**Syntax**
Describe and Modify argument:

"DataWindow.bandname.Text { = rtfstring }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bandname</td>
<td>The identifier of a band in the DataWindow object that has the RichText presentation style. Values are:</td>
</tr>
<tr>
<td></td>
<td>• Detail</td>
</tr>
<tr>
<td></td>
<td>• Header</td>
</tr>
<tr>
<td></td>
<td>• Footer</td>
</tr>
<tr>
<td>rtfstring</td>
<td>A string whose value is the rich text content of the band. The string includes the rich text formatting codes, text, and input fields.</td>
</tr>
<tr>
<td></td>
<td>Text assigned to the header or footer band is ignored if</td>
</tr>
</tbody>
</table>
|               | When you assign text using the Modify method, nested quotes | must be represented with tildes and quotes. If your data is a pure | RTF string, use the PasteRTF method.

**Bands**

**Description**
A list of the bands in the DataWindow object. The list can include one or more of the following band identifiers, where # is the level number of a group: Detail, Footer, Header, Header.#, Summary, Trailer.#, Tree.Level.#. The items in the list are separated by tabs.

**Applies to**
DataWindows

**Syntax**
Describe argument:

"DataWindow.Bands"

**Examples**

[Visual Basic]
Dim BandList As String
BandList = dw1.GetProperty("DataWindow.Bands")
BandList = dw1.Describe("DataWindow.Bands")
[C#]
string BandList;
BandList = dw1.GetProperty("DataWindow.Bands");
BandList = dw1.Describe("DataWindow.Bands");

**BinaryIndex**

**Description**
An internal index that the DataWindow server uses to manage the OLE Object control in the library. There is no reason to get this value; the value has no external significance.

**Applies to**
OLE Object controls

**Syntax**
"olecontrolname.BinaryIndex"

**BitmapName**

**Description**
Whether the DataWindow server interprets the column’s value as the name of a picture file and displays the picture instead of the text. BitmapName’s value is either Yes or No.

**Applies to**
Column controls

**Syntax**
Describe argument:
"columnname.BitmapName"

**Usage**
*In the painter* Select the control and set the value in the Properties window, General category.

**Examples**
ls_data = dw1.Describe("emp_name.BitmapName")

**Border**

**Description**
The type of border for the control.

**Applies to**
Column, Computed Field, Graph, GroupBox, OLE, Picture, Report, TableBlob, and Text controls

**Syntax**
Describe and Modify argument:
"controlname.Border ( = 'value' )"

DataWindowSyntaxFromSql:

Column ( ... Border = value ... )
**Border**

Text ( ... Border = value ... )

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlname</td>
<td>The name of the control whose border you want to get or set. When generating DataWindow syntax with DataWindowSyntaxFromSql, the Border setting applies to all columns or all text controls.</td>
</tr>
<tr>
<td>value</td>
<td>(exp) A number specifying the type of border. Values are: 0 – None, 1 – Shadow, 2 – Rectangle, 3 – Resize, 4 – Line, 5 – 3D Lowered, 6 – 3D Raised. The value can be a quoted DataWindow painter expression. When you change between Resize and another border, change the Resizeable property too so that the control’s appearance and behavior match.</td>
</tr>
</tbody>
</table>

**Usage**

**In the painter** Select the control and set the value in the Properties window, Appearance category.

Changing the Border setting between Resize and another border affects the Resizeable option in the Layout category. To make another border resizable, choose the border then reset Resizeable.

On Windows XP, to display the border of a text column with the XP style (by default, a blue box), set the Border property to Lowered and the BackgroundColor of the font to Window Background.

For a Picture in a Web DataWindow that is a link, the default border displays unless you set the Border property to 0.

For examples of other ways to set properties, using Border as an example, see the DataWindow .NET Programmer’s Guide.

**Examples**

[Visual Basic]

```vbnet
ls_data = dw1.Describe("emp_name_t.Border")
dw1.Modify("emp_name_t.Border=’6’")
```
**Brush.property**

**Description**
Settings for the fill pattern and color of a graphic control.

**Applies to**
Oval, Rectangle, and RoundRectangle controls

**Syntax**
Describe and Modify argument:

```
"controlname.Brush.property { = 'value' }
```

**Parameter** | **Description**
--- | ---
controlname | The name of the Line, Oval, Rectangle, RoundRectangle, or Text control whose Brush property you want to get or set.
property | A property that applies to the Brush characteristics of a control, as listed in the table below.
value | Values for the properties are shown in the next table. Value can be a quoted DataWindow expression.

<table>
<thead>
<tr>
<th>Property for Brush</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Color</td>
<td>(exp) A long expression specifying the color (the red, green, and blue values) to be used to fill the control.</td>
</tr>
<tr>
<td>Hatch</td>
<td>(exp) A number expression specifying the fill pattern of controlname. Values are: 0 – Horizontal 1 – Bdiagonal (lines from lower left to upper right) 2 – Vertical 3 – Cross 4 – Fdiagonal (lines from upper left to lower right) 5 – DiagCross 6 – Solid 7 – Transparent</td>
</tr>
</tbody>
</table>

**Usage**

**In the painter**
Select the control and set the value in the Properties window, Appearance category.

When you choose a Brush Hatch fill pattern other than Solid, the Background Color and the Brush Color are used for the pattern colors.

**Examples**

[Visual Basic]
```
ls_data = dw1.Describe("oval_1.Brush.Hatch")
dw1.Modify("oval_1.Brush.Color='16731766'")
```
**Category**

See Axis, Axis.property, and DispAttr.fontproperty.

**CheckBox.property**

Description: Settings for a column whose edit style is CheckBox.

Applies to: Column controls

Syntax: DataWindow .NET dot notation:

```csharp
((CheckBox) colname.EditStyle).property
```

CType(colname.EditStyle, CheckBox).property [Visual Basic]

Describe and Modify argument:

"columnname.CheckBox.property { = value }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columnname</td>
<td>The column whose edit style is CheckBox for which you want to get or set property values.</td>
</tr>
<tr>
<td>property</td>
<td>A property for the CheckBox edit style, as listed in the table below.</td>
</tr>
<tr>
<td>value</td>
<td>Values for the properties are shown in the table below. For CheckBox properties, value cannot be a DataWindow expression.</td>
</tr>
</tbody>
</table>

**Property for CheckBox**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>LeftText</td>
<td>Whether the CheckBox label is to the left or right of the CheckBox. Values are:</td>
</tr>
<tr>
<td></td>
<td>Yes – Display the label on the left. No – Display the label on the right.</td>
</tr>
<tr>
<td>Off</td>
<td>A string constant specifying the column value when the CheckBox is off (unchecked). The resulting value must be the same datatype as the column.</td>
</tr>
<tr>
<td>On</td>
<td>A string constant specifying the value that will be put in the column when the CheckBox is on (checked). The resulting value must be the same datatype as the column.</td>
</tr>
</tbody>
</table>
### Usage

In the painter, select the control and set values in the Properties window, Behavior category, when EditStyle is CheckBox.

In DataWindow.NET, you can use the CheckBox class to set CheckBox style properties using dot notation. See the description of the CheckBox class in the online Help in Visual Studio.NET for a complete list of properties. Some properties have different names in DataWindow.NET.

#### Examples

```visualbasic
    dw1.Modify("emp_gender.CheckBox.3D=no")
    dw1.Modify("emp_status.CheckBox.Off='Terminated'")
    dw1.Modify("emp_status.CheckBox.On='Active'")
    dw1.Modify("emp_status.CheckBox.Other='Unknown'")
```  

#### Other

A string constant specifying the value that will be put in the column when the CheckBox is in the third state (neither checked nor unchecked). The value must be the same datatype as the column.

Painter: Behavior category. This option is available when ThreeStates is True.

#### Scale

Whether you want to scale the 2D CheckBox. Takes effect only when the ThreeD property is False.

Values are:
- Yes – Scale the CheckBox.
- No – Do not scale the CheckBox.

Painter: Behavior category.

#### Text

A string specifying the CheckBox’s label text.

Painter: Behavior category.

#### ThreeD

Whether the CheckBox should be 3D.

Values are:
- Yes – Make the CheckBox 3D
- No – Do not make the CheckBox 3D

Painter: Behavior category.

#### ThreeStates

Whether the CheckBox should have three states.

Values are:
- Yes – The CheckBox has three states
- No – The CheckBox does not have three states

Painter: Behavior category.
ClientName

Description
The name of the OLE client. The default is “Untitled.” ClientName is used by some applications in the server window’s title.

Applies to
OLE Object and TableBlob controls

Syntax
Describe and Modify argument:

"controlname.ClientName { = 'clientname ' }"

Usage
In the painter Select the control and set the value in the Properties window, Definition category.

Examples
[Visual Basic]
cname = dw1.Describe("emppict_blob.ClientName")
dw1.Modify("emppict_blob.ClientName=" + _ "-t-"Data for -" + String(emp_id)"")
Color

Description: The text color of the column or the background color of the DataWindow. The color affected by the Color property depends on the control:

- For the DataWindow, Color specifies the background color
- For columns, computed fields, and text, Color specifies the text color
- For graphs, Color specifies the line color used for axes, borders around data markers, tick marks, and the outline of the box for 3D graphs

Applies to: DataWindow, Button, Column, Graph, and GroupBox controls

Syntax

Describe and Modify argument:

```
"DataWindow.Color { = long }

"controlname.Color { = long }
```

DataWindowSyntaxFromSql:

```
DataWindow ( Color = long )
Column ( Color = long )
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlname</td>
<td>The column whose text color you want to set or the graph whose line color you want to set.</td>
</tr>
<tr>
<td>long</td>
<td>(exp for columns only) A long value specifying the color of the column text or the DataWindow background. When you are specifying the text color of a column, you can specify a DataWindow expression in quotes. You cannot specify an expression for the DataWindow background color. When generating DataWindow syntax with DataWindowSyntaxFromSql, the Color setting for Column applies to all columns.</td>
</tr>
</tbody>
</table>

Usage

In the painter For the DataWindow background, click the DataWindow to deselect all controls and set the value in the Properties window, General category.

For a column’s text color, select the column and set the value in the Properties window, Appearance category.

For a graph’s line color, select the graph and set the value in the Properties window, Appearance category.
**ColType**

**Examples**

```vbnet
dw_back_color = dw1.Describe("DataWindow.Color")
column_text_color = dw1.Describe("emp_name.Color")
dw1.Modify(  
    "salary.Color='0-tIf(salary>90000,255,65280)'"
)
```

**See also**  
Background, BackColor

---

**ColType**

**Description**  
The datatype of the column or computed field.

**Applies to**  
Column and Computed Field controls

**Syntax**

```
"controlname.ColType"
```

**Parameter** | **Description**
--- | ---
`controlname` | The column for which you want the datatype. Possible datatypes are:
  - Char (n) – n is the number of characters
  - Date
  - DateTime
  - Decimal (n) – n is the number of decimal places
  - Int
  - Long
  - Number
  - Real
  - Time
  - Timestamp
  - ULong

**Usage**

**In the painter**  
The value of ColType is derived from the data or expression you specify for the control. The value is displayed in the Column Specifications view.
**Date column types**

If you define a DataWindow with a column of type Date and deploy it with a DBMS that uses the DateTime datatype, set the StaticBind database parameter to 0 or No. This forces the DataWindow server to get a result set description before retrieving data and adjust the bind information if necessary.

For more information, see the StaticBind DBParm parameter in the online Help.

**Examples**

[Visual Basic]
```
ls_coltype = dw1.Describe("emp_id.ColType")
```

**Column.Count**

**Description**
The number of columns in the DataWindow object.

**Applies to**
DataWindows

**Syntax**
DataWindow .NET dot notation:
```
dw_control.ColumnCount
```
Describe argument:
```
"DataWindow.Column.Count"
```

**Usage**

**In the painter**  The value is determined by the number of columns you select in the SQL Select painter, whether or not they are displayed.

In DataWindow .NET, you can use the ColumnCount property on DataWindowControl, WebDataWindowControl, DataStore, and DataWindowChild to get the number of columns.

**Column limit**
There is a limit of 1000 on the number of columns in a DataWindow object.

**Examples**

[C#]
```
short ColCount;
ColCount = dw1.ColumnCount;
```

[Visual Basic]
```
ls_colcount = dw1.Describe("DataWindow.Column.Count")
```
ContentsAllowed

Description
The way the OLE Object control holds the OLE object. You can restrict the container to only embedded or only linked objects, or you can allow either type.

Applies to
OLE Object controls

Syntax
Describe and Modify argument:
"olecontrolname.ContentsAllowed { = 'contentstype' }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>olecontrolname</td>
<td>The name of the OLE Object control for which you want to get or set the type of contents.</td>
</tr>
<tr>
<td>contentstype</td>
<td>A number specifying whether the OLE object in the control has to be embedded, has to be linked, or can be either embedded or linked. Values are: 0 – Embedded 1 – Linked 2 – Any</td>
</tr>
</tbody>
</table>

Usage
In the painter
Select the control and set the value in the Properties window, Options category.

Examples
[Visual Basic]
ls_data = dw1.Describe("ole_report.ContentsAllowed")
dw1.Modify("ole_report.ContentsAllowed='2'")

Criteria

Description
The search condition of the WHERE clause for a related report. The Criteria property defines the connection between the related report and the DataWindow.

Applies to
Report controls

Syntax
Describe and Modify argument:
"reportname.Criteria { = string }")
### Criteria.property

**Description**
Settings for the Prompt for Criteria dialog box. When Prompt for Criteria is enabled, the DataWindow server prompts the user to specify criteria for retrieving data whenever the Retrieve method is called. Note that the Required property also affects query mode.

**Syntax**
Describe and Modify argument:

```
"columnname.Criteria.property ( = value )"
```

<table>
<thead>
<tr>
<th><strong>Parameter</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>columnname</td>
<td>The name of the column for which you want to get or set Prompt for Criteria properties.</td>
</tr>
<tr>
<td>property</td>
<td>A property for the Prompt for Criteria dialog box. Properties and their settings are listed in the table below.</td>
</tr>
<tr>
<td>value</td>
<td>A Yes or No value to be assigned to the property. For Criteria properties, value cannot be a DataWindow expression.</td>
</tr>
</tbody>
</table>

**Property for Criteria**

<table>
<thead>
<tr>
<th><strong>Value</strong></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Dialog</td>
<td>Whether Prompt for Criteria is on for columnname.</td>
</tr>
<tr>
<td></td>
<td>Values are:</td>
</tr>
<tr>
<td></td>
<td>Yes – Include columnname in the Prompt for Criteria dialog box.</td>
</tr>
<tr>
<td></td>
<td>No – (Default) Do not include columnname in the Prompt for Criteria dialog box.</td>
</tr>
<tr>
<td></td>
<td>If the Dialog property is Yes for at least one column in the DataWindow, then the DataWindow server displays the Prompt for Criteria dialog box when the Retrieve method is called.</td>
</tr>
<tr>
<td></td>
<td>Painter: Column Specifications view, Prompt check box.</td>
</tr>
</tbody>
</table>
Crosstab.property

<table>
<thead>
<tr>
<th>Property for Criteria</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Override_Edit</td>
<td>Whether the user must enter data in the Prompt for Criteria dialog box according to the edit style defined for the column in the DataWindow object or be allowed to enter any specifications in a standard edit control.</td>
</tr>
<tr>
<td></td>
<td>Values are:</td>
</tr>
<tr>
<td></td>
<td>Yes – Allow the user to override the column’s edit style and enter data in a standard edit control.</td>
</tr>
<tr>
<td></td>
<td>No – (Default) Constrain the user to the edit style for the column.</td>
</tr>
<tr>
<td></td>
<td>Painter: Properties window, Appearance category.</td>
</tr>
<tr>
<td>Required</td>
<td>Whether the user is restricted to the equality operator (=) when specifying criteria in query mode and in the Prompt for Criteria dialog box.</td>
</tr>
<tr>
<td></td>
<td>Values are:</td>
</tr>
<tr>
<td></td>
<td>Yes – Require the user to use the equality operator only.</td>
</tr>
<tr>
<td></td>
<td>No – (Default) Allow the user to use any relational operator, including =, &lt;&gt;, &lt;, &gt;, &gt;=, and &lt;=.</td>
</tr>
<tr>
<td></td>
<td>Painter: Properties window, Appearance category.</td>
</tr>
</tbody>
</table>

Usage

In the painter  Set the values using the menus and Properties window as described in the table above.

Examples

[Visual Basic]

```
setting = dw1.Describe("empname.Criteria.Dialog")
dw1.Modify("empname.Criteria.Dialog=Yes")
dw1.Modify("empname.Criteria.Override_Edit=Yes")
dw1.Modify("empname.Criteria.Required=No")
```

Crosstab.property

Description

Settings for a DataWindow object whose presentation style is Crosstab.

Applies to

DataWindows

Syntax

Describe and Modify argument:

"DataWindow.Crosstab.property { = value }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>property</td>
<td>A property for a Crosstab DataWindow. Properties and their settings are listed in the table below.</td>
</tr>
<tr>
<td>value</td>
<td>A string expression listing the items to be assigned to the property. For Crosstab properties, value is always quoted and can be a DataWindow expression.</td>
</tr>
</tbody>
</table>
## CHAPTER 3  DataWindow Object Properties

### Property for Crosstab | Value
---|---
Columns | *(exp)* A string containing a comma- or tab-separated list of the names of columns that make up the columns of the crosstab. These are the columns that display across the top of the crosstab.
Rows | *(exp)* A string containing a comma- or tab-separated list of the names of columns that make up the rows of the crosstab.
SourceNames | *(exp)* A string containing a comma-separated list of column names to be displayed in the Crosstab Definition dialog box. The default names are the column names from the database.
StaticMode | A string indicating whether a dynamic crosstab should be put into a static mode. The dynamic crosstab remains in static mode until you set StaticMode to No. While the dynamic crosstab is in static mode, you can manipulate the properties of individual columns.
Values | *(exp)* A string containing a comma- or tab-separated list of expressions that will be used to calculate the values of the crosstab.

### Usage
**In the painter** For DataWindow objects with the Crosstab presentation style, set the values in the Crosstab Definition dialog box. To display the dialog box, right-click in the Design view to display the pop-up menu and select Crosstab.

### Examples
```
[Visual Basic]
setting = dw1.Describe("DataWindow.Crosstab.Columns")
dw1.Modify("DataWindow.Crosstab.Columns='dept_id'")
dw1.Modify("DataWindow.Crosstab.Rows='salary'")
dw1.Modify("DataWindow.Crosstab.Values='empname'")
dw1.Modify("DataWindow.Crosstab.StaticMode='yes'")
```

### See also
CrosstabDialog function
**CSSGen.property**

**Description**
Settings that specify the physical path to which a generated CSS style sheet is published and the URL where the style sheet is located.

**Applies to**
DataWindow objects

**Syntax**
Describe and Modify argument:

```
"DataWindow.CSSGen.property { = ' value ' }
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>property</td>
<td>One of the following:</td>
</tr>
<tr>
<td></td>
<td>• PublishPath</td>
</tr>
<tr>
<td></td>
<td>• ResourceBase</td>
</tr>
<tr>
<td></td>
<td>• SessionSpecific</td>
</tr>
<tr>
<td>value</td>
<td>(exp) PublishPath – a string that specifies the physical path of the Web site folder to which the DataWindow server publishes the generated CSS style sheet</td>
</tr>
<tr>
<td></td>
<td>(exp) ResourceBase – a string that specifies the URL of the generated CSS style sheet to be referenced in a link element in the XHTML page</td>
</tr>
<tr>
<td></td>
<td>(exp) SessionSpecific – a boolean that when set to “yes” forces a session-specific ID to be applied to any generated document names that would otherwise be shared</td>
</tr>
</tbody>
</table>

**Usage**
The PublishPath folder must correspond to the URL specified in the ResourceBase property. At runtime, after the DataWindow server generates the CSS style sheet to the PublishPath folder, it includes it in the final XHTML page by referencing it with the ResourceBase property in a `<link>` element.

Typically you share style (CSS), layout (XSLT), and control definitions (JS) for use by all clients; however, if you use dynamic DataWindow objects customized for specific clients, you can force generation of the DataWindow presentation-related document names to be specific to each client. You do this by setting the CSSGen.SessionSpecific property to “yes”. This eliminates the possibility of server-side contention for presentation formats when the DataWindow generation is specific to the client.

**In the painter**
In the Web Generation category in the Properties window for the DataWindow object, select CSS from the WebDW list, specify the Resource Base and Publish Path locations, and set the SessionSpecific property to True if you want to force generation of client-specific names.
In DataWindow .NET  In DataWindow .NET, you can specify the physical directory in which dynamically created files, such as .css, .js, .xml, and .xslt files, and URL references are stored, using the XmlConfigurations.UrlPath property. If you specify a value for XmlConfigurations.UrlPath, it overrides the values for PublishPath and ResourceBase set in the DataWindow painter.

If you want to specify different paths for each of the different file types, if you want to specify a full path, or if you want to specify different paths for PublishPath and ResourceBase, set the properties in the DataWindow painter and leave the XmlConfigurations.UrlPath property empty in DataWindow .NET.

If you do not set these properties in the DataWindow painter or in DataWindow .NET, the files are saved in the current Web application’s path.

Examples

These statements set the CSSGen.ResourceBase and CSSGen.PublishPath properties:

[Visual Basic]
dw1.Modify("DataWindow.CSSGen.PublishPath= 'C:\Inetpub\wwwroot\MyWebApp\generatedfiles'" )

This statement sets the CSSGen.SessionSpecific property for a JSP page:

dwGen.Modify
("DataWindow.CSSGen.SessionSpecific='Yes'" );

Data

Description  A tab-separated list describing the data in the DataWindow object.

Applies to  DataWindows

Syntax  Describe argument:

"DataWindow.Data"

Examples

[Visual Basic]
setting = dw1.Describe("DataWindow.Data")
**Data.HTML**

**Description**
A string containing HTML and JavaScript that represents data and presentation of the DataWindow object.

The data is presented in a read-only HTML table or data-entry form, depending on settings of other properties.

**Applies to**
DataWindows

**Syntax**
Describe argument:
"DataWindow.Data.HTML"

**Usage**
When HTMLDW is set to False, the value of Data.HTML is the same as the value of HTMLTable—a read-only HTML table that displays all retrieved rows.

When the HTMLDW property is set to True, the value of Data.HTML is a form that supports data input with client scripts for data validation and events. The generated string for Data.HTML includes:

- HTML input elements
- JavaScript for validating newly entered data based on validation rules in the DataWindow object
- HTML and JavaScript for navigation based on DataWindow Button controls with scrolling actions
- State information about the modification status of data items

JavaScript for navigation passes the state of the DataWindow back to the page server in two variables: `objectname_action` and `objectname_context`. It also passes back any page parameters defined in the HTMLGen.SelfLinkArgs property. All the HTMLGen.property values affect the way HTML is generated.

The resulting Web DataWindow is a client-side control for a Web page with events and methods that can cooperate with a server component for a Web-based data entry application. For more information about the Web DataWindow, see the *Programmer’s Guide*.

**Exceptions**
If the DataWindow is in print preview mode, or there are no columns with non-zero tab order, the setting of HTMLDW is ignored and the generated HTML is a read-only table, not a data-entry form.

**Examples**

**[Visual Basic]**

```vbnet
strHtml = dw1.Describe("DataWindow.Data.HTML")
```
Data.HTMLTable

**Description**
The data in the DataWindow object described in HTML table format. This property is used in the process of dynamically creating Web pages from a database.

**Applies to**
DataWindows

**Syntax**
Describe argument:

```
"DataWindow.Data.HtmlTable"
```

**Usage**
Some presentation styles translate better into HTML than others. The Tabular, Group, Freeform, Crosstab, and Grid presentation styles produce good results. The Composite, TreeView, and Graph presentation styles produce HTML tables based on the result set only and not on the presentation style. DataWindows with overlapping controls in them might not produce the desired results. Nested reports are ignored; they are not included in the generated HTML.

The generated HTML for Data.HTMLTable is a read-only HTML Table element that includes:

- All retrieved rows (in contrast to the Web DataWindow, which paginates the result set)
- Hyperlinks for text, pictures, computed fields, and columns as defined in the HTML.property settings

Data.HTMLTable is not affected by the HTMLEDW property and does not generate a client control with events and support for scripting in the Web page.

The values of HTMLGen.Browser and HTMLGen.Version affect the generated HTML. Setting these properties causes the generated HTML to be optimized for a specific level of HTML support or specific browser using style sheets and absolute positioning, if possible. For more information, see HTMLGen.property.

The resulting HTML table does not allow data entry.

**An easy way to see a DataWindow in a Web browser**
The HTML string that the Data.HTMLTable property returns is equivalent to the string that is saved when you use either the File>Save Rows As HTML Table option in the DataWindow painter workspace or the SaveAs method.

To see what a DataWindow will look like, save it as an HTML file and open the file in a Web browser such as Netscape.
In the painter  When HTMLDW is not selected, the Design>HTML Preview displays the value of Data.HTMLTable. Save an HTML file that you can use later in a browser with File>Save Rows As; set the Save As Type to HTML Table.

Examples

[Visual Basic]
ls_html = dwl.Describe("DataWindow.Data.HTMLTable")

Data.XHTML

Description  A string containing the row data content of the DataWindow object in XHTML format.

Applies to  DataWindows

Syntax  Describe argument:

"DataWindow.Data.XHTML"

Usage  If any of the Export.XHTML properties have been set, the string that is generated reflects the values of these properties.

The resulting XHTML string contains a <form> element that supports data input, which works with separate client scripts for data validation and events.

The generated XHTML string also includes:

• XHTML input elements

• XHTML and JavaScript for navigation based on DataWindow button controls with scrolling actions

• State information about the modification status of data items

JavaScript for navigation passes the state of the DataWindow back to the page server in two variables: objectname_action and objectname_context. It also passes back any page parameters defined in the HTMLGen.SelfLinkArgs property. All applicable HTMLGen.property values also affect the way the XHTML is generated.

The resulting XML Web DataWindow is a client-side control for a Web page, such as a JSP page, with events and methods that can cooperate with a server component for a Web-based data entry application.
Examples

The following statements set the template used by the DataWindow `dw1` to `t_report` and return the generated XHTML document to the string `ls_XHTML`. To generate the string, the final statement invokes the XML Web DataWindow generator to generate the XHTML, CSS, and JavaScript components, applying the `t_report` template to the generated XHTML and CSS style sheet.

```visualbasic
[Visual Basic]
dw1.Modify("DataWindow.Export.XHTML.UseTemplate = 't_report'")
strXHTML = dw1.Describe("DataWindow.Data.XHTML")
```

### Data.XML

**Description**

A string containing the row data content of the DataWindow object in XML format.

**Applies to**

DataWindows

**Syntax**

Describe argument:

"DataWindow.Data.XML"

**Usage**

If any of the Export.XML properties have been set, the string that is generated reflects the values of these properties.

**Note**

If Export.XML.SaveMetaData is set to MetaDataExternal!, no metadata is generated in the string.

**Examples**

The following statements set the template used by the DataWindow `dw1` to `t_report`, specify that metadata in the XMLSchema! format should be included in the generated XML, and return the generated XML document to the string `ls_xml`.

```visualbasic
[Visual Basic]
dw1.Modify("DataWindow.Export.XML.UseTemplate = 't_report'")
dw1.Modify("DataWindow.Export.XML.SaveMetaData = MetaDataInternal!")
dw1.Modify
("DataWindow.Export.XML.MetaDataType = XMLSchema!")
```
**Data.XMLDTD**

**Description**
A string containing the full document type definition (DTD) of the XML output for a DataWindow object.

**Applies to**
DataWindows

**Syntax**
Describe argument:
"DataWindow.Data.XMLDTD"

**Usage**
Use this property to return the full DTD of the XML output of a DataWindow object separately from the generated XML document itself. The export template used affects the generated DTD.

**Data.XMLSchema**

**Description**
A string containing the full schema of the XML output of a DataWindow object.

**Applies to**
DataWindows

**Syntax**
Describe argument:
"DataWindow.Data.XMLSchema"

**Usage**
Use this property to return the full schema of the XML output of a DataWindow object separately from the generated XML document itself. The export template used affects the generated schema.

**Data.XMLWeb**

**Description**
A string containing browser-specific JavaScript that performs the XSLT transformation on the browser after the XML Web DataWindow generator generates all necessary components.

**Applies to**
DataWindows

**Syntax**
Describe argument:
"DataWindow.Data.XMLWeb"

**Usage**
If any of the Export.XHTML properties have been set, the string that is generated reflects the values of these properties. The resulting XHTML string contains a `<form>` element that supports data input, which works with separate client scripts for data validation and events.
The generated XHTML string also includes:

- XHTML input elements
- XHTML and JavaScript for navigation based on DataWindow button controls with scrolling actions
- State information about the modification status of data items

JavaScript for navigation passes the state of the DataWindow back to the page server in two variables: `objectname_action` and `objectname_context`. It also passes back any page parameters defined in the HTMLGen.SelfLinkArgs property. All applicable HTMLGen.property values also affect the way the XHTML is generated.

The resulting XML Web DataWindow is a client-side control for a Web page, such as a JSP page, with events and methods that can cooperate with a server component for a Web-based data entry application.

### Data.XSLFO

**Description**
A string containing XSL Formatting Objects (XSL-FO) that represents the data and presentation of the DataWindow object.

This property is not supported in DataWindow.NET.

**Applies to**
DataWindows

**Syntax Describe and Modify argument:**

```
"DataWindow.Data.XSLFO"
```

**Usage**
Use this property to return the data and presentation of a DataWindow object in XSL-FO format. The export template associated with the DataWindow object does not affect the generated string.

### DataObject

**Description**
The name of the DataWindow object that is the nested report within the main DataWindow object.

**Applies to**
Report controls

**Syntax Describe and Modify argument:**

```
"reportname.DataObject = 'dwname' "
```
### dbAlias

#### Description

The name of the database column but with the table alias in place of the table name, if any. This value can be used to construct the update DataWindow syntax dynamically when an alias name is used for a column.

#### Applies to

Column controls

#### Syntax

Describe and Modify argument:

```
"columnname.dbAlias { = 'dbcolumnname' }
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columnname</td>
<td>The name of the column for which you want the name of the corresponding database column qualified with the table alias name</td>
</tr>
<tr>
<td>dbcolumnname</td>
<td>The name of the database column associated with columnname qualified with the alias of the table name</td>
</tr>
</tbody>
</table>

#### Usage

DbAlias is the name of the database column in the format `tablealiasname.columnname`. The value of dbAlias does not include the quotes that can be part of the SQL syntax. This property can be used to construct update DataWindow syntax dynamically when an alias is used for a column name.

**In the painter**

You can specify an alias for a table in the SQL Select painter if you convert the SQL statement for a DataWindow object to syntax. Select Design> Data Source to open the SQL Select painter, then select Design> Convert to Syntax. In the text window that displays, add the alias name to the FROM clause using the syntax:

```
FROM tablename tablealiasname
```
Suppose a DataWindow object has the following SQL Select syntax, with the alias “emp” for the table “employee”:

```
SELECT "emp"."emp_id",
    "emp"."emp_fname",
    "emp"."emp_lname",
    "emp"."dept_id",
    "emp"."salary"
FROM "employee" "emp"
WHERE ( "emp"."salary" > 50000 )
```

Then the following statements would return the string “employee.emp_id” in `ls_name` and the string “emp.emp_id” in `ls_alias`:

```
[Visual Basic]
strName = dw1.Describe("emp_id.dbName")
strName = dw1.Describe("emp_id.dbAlias")
```

### dbName

**Description**

The name of the database column. The DataWindow server uses this value to construct the update syntax.

**Applies to**

Column controls

**Syntax**

Describe and Modify argument:

```
"columnname.dbName { = 'dbcolumnname' }
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>columnname</code></td>
<td>The name of the column for which you want the name of the corresponding database column</td>
</tr>
<tr>
<td><code>dbcolumnname</code></td>
<td>The name of the database column associated with <code>columnname</code></td>
</tr>
</tbody>
</table>

**Usage**

DbName is the name of the database column in the format `tablename.columnname`. The value of dbName does not include the quotes that can be part of the SQL syntax.

**In the painter**

The Syntax view in the SQL Select painter displays the database column names (they can be shown with quotes).

**Examples**

```
[Visual Basic]
dbcol = dw1.Describe("emp_id.dbName")
dw1.Modify("emp_id.dbName='emp_id'")
```

**See also**

`dbAlias`
Properties that control the appearance and behavior of a column with the DropDownDataWindow edit style.

Applies to Column controls

Syntax DataWindow .NET dot notation:

```csharp
(( DDDW) colname.EditStyle).property
```

```
CType(colname.EditStyle, DDDW).property
```

Describe and Modify argument:

"columnname.dddw.property { = value }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columnname</td>
<td>The name of a column that has the DropDownDataWindow edit style.</td>
</tr>
<tr>
<td>property</td>
<td>A property for the DropDownDataWindow column. Properties and their settings are listed in the table below.</td>
</tr>
<tr>
<td>value</td>
<td>The value to be assigned to the property. For dddw properties, value cannot be a DataWindow expression.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property for dddw</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllowEdit</td>
<td>Whether the user can type a value as well as choose from the DropDownDataWindow’s list.</td>
</tr>
<tr>
<td></td>
<td>Values are:</td>
</tr>
<tr>
<td></td>
<td>Yes – Typing is allowed.</td>
</tr>
<tr>
<td></td>
<td>No – (Default) Typing is not allowed.</td>
</tr>
<tr>
<td></td>
<td>Call GetChild after setting dddw.AllowEdit to get a valid reference to the column’s DropDownDataWindow.</td>
</tr>
<tr>
<td>AutoHScroll</td>
<td>Whether the DropDownDataWindow automatically scrolls horizontally when the user enters or deletes data.</td>
</tr>
<tr>
<td></td>
<td>Values are:</td>
</tr>
<tr>
<td></td>
<td>Yes – (Default) Scroll horizontally automatically.</td>
</tr>
<tr>
<td></td>
<td>No – Do not scroll automatically.</td>
</tr>
<tr>
<td>AutoRetrieve</td>
<td>Whether the DropDownDataWindow data is retrieved when the parent DataWindow data is retrieved.</td>
</tr>
<tr>
<td></td>
<td>Values are:</td>
</tr>
<tr>
<td></td>
<td>Yes – (Default) Data is automatically retrieved.</td>
</tr>
<tr>
<td></td>
<td>No – Data must be retrieved separately.</td>
</tr>
<tr>
<td>Property for dddw</td>
<td>Value</td>
</tr>
<tr>
<td>------------------</td>
<td>-------</td>
</tr>
</tbody>
</table>
| Case             | The case of the text in the DropDownDataWindow. Values are:  
|                  | Any – Character of any case allowed.  
|                  | Upper – Characters converted to uppercase.  
|                  | Lower – Characters converted to lowercase.  
|                  | Call GetChild after setting dddw.Case to get a valid reference to the column’s DropDownDataWindow. |
| DataColumn       | A string whose value is the name of the data column in the associated DropDownDataWindow. Value is quoted.  
|                  | Call GetChild after setting dddw.DataColumn to get a valid reference to the column’s DropDownDataWindow. |
| DisplayColumn    | A string whose value is the name of the display column in the associated DropDownDataWindow. Value is quoted.  
|                  | Call GetChild after setting dddw.DisplayColumn to get a valid reference to the column’s DropDownDataWindow. |
| HScrollBar       | Whether a horizontal scroll bar displays in the DropDownDataWindow. Values are:  
|                  | Yes – Display a horizontal scroll bar.  
|                  | No – Do not display a horizontal scroll bar. |
| HSplitScroll     | Whether the horizontal scroll bar is split. The user can adjust the split position. Values are:  
|                  | Yes – Split the horizontal scroll bar so the user can scroll the display and data columns separately.  
|                  | No – The horizontal scroll bar is not split. |
| Limit            | An integer from 0 to 32767 specifying the maximum number of characters that can be entered in the DropDownDataWindow. Zero means unlimited. |
| Lines            | An integer from 0 to 32767 specifying the number of lines (values) to display in the DropDownDataWindow. This property does not apply in Web pages because the browser controls how the DropDownDataWindow displays. |
| Name             | A string whose value is the name of the DropDownDataWindow associated with the column.  
|                  | Call GetChild after setting dddw.Name to get a valid reference to the column’s DropDownDataWindow. |
| NilIsNull        | Whether to set the data value of the DropDownDataWindow to null when the user leaves the edit box blank. Values are:  
|                  | Yes – Make the Empty string null.  
|                  | No – Do not make the empty string null. |
**Usage**

**DropDownDataWindows and GetChild**  When you set some of the `dddw` properties, as noted in the table, references to the `DropDownDataWindow` become invalid. Call `GetChild` after changing these properties to obtain a valid reference.

To retrieve a `DropDownDataWindow` when the `AutoRetrieve` property is set to `False`, you can access the object data as follows:

```vbnet
[Visual Basic]
_dw1.GetChild ("dept_head_id", _mgr_id)
_mgr_id.SetTransaction (SQLCA)
_mgr_id.Retrieve ()
```

You can also pass a retrieval argument for the retrieve on the child `DataWindow` object.
Doing a reset to clear the data  When a DropDownDataWindow is retrieved, its data is kept with its own Data Object. If you retrieve the DropDownDataWindow and then set the AutoRetrieve property on the parent to False, the data for the child is not cleared on a reset and re-retrieve of the parent.

To clear data from a DropDownDataWindow, you must call Reset on the child DataWindow object:

```java
    dw1.GetChild("dept_head_id", mgr_id)
    mgr_id.reset()
```

In the painter  Select the control and set values in the Properties window, Behavior category, when EditStyle is DropDownDW.

In DataWindow .NET, you can use the DDDW class to set DropDownDW style properties using dot notation. See the description of the DDDW class in the online Help in Visual Studio .NET for a complete list of properties. Some properties have different names and are set differently. For example, the Case DataWindow object property is equivalent to the CharacterCasing property in DataWindow .NET, which uses an enumeration.

Examples

```vbnet
    ls_data = dw1.Describe("emp_status.dddw.AllowEdit")
    dw1.Modify("emp_status.dddw.Case='Any'")
    dw1.Modify("emp_status.dddw.DataColumn='status_id'")
    dw1.Modify("emp_status.dddw.Limit=30")
    dw1.Modify("emp_status.dddw.Name='d_status'")
    dw1.Modify("emp_status.dddw.PercentWidth=120")

    If TypeOf state.EditStyle Is DDDW Then
        CType(state.EditStyle, DDDW).AllowEdit = True
    ElseIf
        ...
    End If

    If state.EditStyle Is DDDW Then
        CType(state.EditStyle, DDDW).AllowEdit = True
    Else
        ...
```
**ddlb.property**

**Description**
Properties that control the appearance and behavior of a column with the DropDownListBox edit style.

**Applies to**
Column controls

**Syntax**
DataWindow .NET dot notation:

C#:
```
((DDLBP\noname\n.EditStyle).\n\nproperty \n```

Visual Basic:
```
CType(muname\n\n.EditStyle, DDLBP\n\nproperty \n```

**Describe and Modify argument:**

```
"\nunamename\n\n.ddlb.property \n\n( = value )"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columnname</td>
<td>The name of a column that has the DropDownListBox edit style.</td>
</tr>
<tr>
<td>property</td>
<td>A property for the DropDownListBox column. Properties and their settings are listed in the table below.</td>
</tr>
<tr>
<td>value</td>
<td>The value to be assigned to the property. For ddb properties, value cannot be a DataWindow expression.</td>
</tr>
</tbody>
</table>

**Property for ddb**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AllowEdit</td>
<td>Whether the user can type a value as well as choose from the DropDownListBox’s list. Values are: Yes – Typing is allowed. No – (Default) Typing is not allowed.</td>
</tr>
<tr>
<td>AutoHScroll</td>
<td>Whether the DropDownListBox automatically scrolls horizontally when the user enters or deletes data. Values are: Yes – (Default) Scroll horizontally automatically. No – Do not scroll automatically.</td>
</tr>
<tr>
<td>Case</td>
<td>The case of the text in the DropDownListBox. Values are: Any – Character of any case allowed. Upper – Characters converted to uppercase. Lower – Characters converted to lowercase.</td>
</tr>
<tr>
<td>Limit</td>
<td>An integer from 0 – 32767 specifying the maximum number of characters that can be entered in the DropDownListBox. Zero means unlimited.</td>
</tr>
</tbody>
</table>
Usage

In the painter  Select the control and set the value in the Properties window, Behavior category, when EditStyle is DropDownListBox.

In DataWindow .NET, you can use the DDLB class to set DropDownListBox style properties using dot notation. See the description of the DDLB class in the online Help in Visual Studio .NET for a complete list of properties. Some properties have different names and are set differently. For example, the Case DataWindow object property is equivalent to the CharacterCasing property in DataWindow .NET, which uses an enumeration.
**DefaultPicture**

**Description**
Specifies whether a button displays a default picture for the button’s action.

**Applies to**
Button controls

**Syntax**
Describe and Modify argument:

```
"buttonname.DefaultPicture { = 'value' }
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>buttonname</td>
<td>The name of the button to which you want to assign an action.</td>
</tr>
<tr>
<td>value</td>
<td>Whether the action’s default picture is used.</td>
</tr>
<tr>
<td></td>
<td>Values are:</td>
</tr>
<tr>
<td></td>
<td>Yes – Use the default picture.</td>
</tr>
<tr>
<td></td>
<td>No – Do not use the default picture.</td>
</tr>
</tbody>
</table>

**Usage**
Default pictures can be associated with all button action types. However, the only default pictures provided for use on a Web DataWindow are: InsertRow, PageFirst, PageLast, PageNext, PagePrior, Retrieve, and Update. These pictures are included as GIF files in the DWACTION110.JAR file in the Sybase\DataWindow Designer 2.5 directory.

For the Web DataWindow, you must uncompress the dwaction110.jar file, deploy the individual GIF files to your Web site, and specify their location with the DataWindow HTMLGen.ResourceBase property that you can set in the JavaScript Generation category in the DataWindow’s Property window.

```vbnet
ls_data = dw1.Describe("emp_status.ddlb.AllowEdit")
dw1.Modify("emp_status.ddlb.Case='Any'")
dw1.Modify("emp_status.ddlb.Limit=30")

[Visual Basic]
If TypeOf status.EditStyle Is DDLB Then
    CType(status.EditStyle, DDLB).AllowEdit = True
ElseIf
...

[C#]
if (emp_status.EditStyle is DDLB)
{
    ((DDLB)emp_status.EditStyle).AllowEdit = true;
}
else
....
```
You can add your own action pictures by setting the DefaultPicture property to False and setting the Filename property to the file name for the picture you want. You can use a URL instead of a complete path to qualify the file name, and you can leave off the URL server name, mapping prefix, and folder name if you set them in the HTMLGen.ResourceBase property.

A user-defined action does not have a default picture associated with it.

**In the painter**  
Select the control and set the value in the Properties window, Appearance category. When the DefaultPicture is not set, you can specify a picture file name in the FileName property. Button pictures can be BMP, GIF, or JPEG files.

**Examples**

```
[Visual Basic]
setting = dw1.Describe("b_name.DefaultPicture")
dw1.Modify("b_name.DefaultPicture = 'No'")
```

**See also**

HTMLGen.property  
DefaultPicture  
Filename

## Depth

**Description**  
The depth of a 3D graph.

**Applies to**  
Graph controls

**Syntax**  
Describe and Modify argument:

```
"graphname.Depth { = 'depthpercent' }
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>graphname</td>
<td>The graph control within the DataWindow for which you want to set the depth.</td>
</tr>
<tr>
<td>depthpercent</td>
<td>(exp) An integer whose value is the depth of the graph, specified as a percentage of the graph’s width. Depthpercent can be a quoted DataWindow expression.</td>
</tr>
</tbody>
</table>

**Usage**  
**In the painter**  
Select the control and set the value in the Properties window, General category.

**Examples**

```
[Visual Basic]
setting = dw1.Describe("graph_1.Depth")
dw1.Modify("graph_1.Depth='70'")
```
**Detail_Bottom_Margin**

**Description**
The size of the bottom margin of the DataWindow’s detail area.

**Applies to**
Style keywords

**Syntax**
DataWindowSyntaxFromSql:

```
Style ( Detail_Bottom_Margin = value )
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>An integer specifying the size of the bottom margin of the detail area in the units specified for the DataWindow.</td>
</tr>
</tbody>
</table>

**Examples**

[Visual Basic]

```
DWSyntaxGenerator.DataWindowSyntaxFromSql(myTrans, sqlstring, 'Style(...Detail_Bottom_Margin = 25 ...)')
```

**Detail_Top_Margin**

**Description**
The size of the top margin of the DataWindow’s detail area.

**Applies to**
Style keywords

**Syntax**
DataWindowSyntaxFromSql:

```
Style ( Detail_Top_Margin = value )
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>An integer specifying the size of the top margin of the detail area in the units specified for the DataWindow.</td>
</tr>
</tbody>
</table>

**Examples**

[Visual Basic]

```
DWSyntaxGenerator.DataWindowSyntaxFromSql(myTrans, sqlstring, 'Style(...Detail_Top_Margin = 25 ...)')
```

**Detail.property**

See Bandname.property.
**DispAttr.fontproperty**

**Description**  
Settings for the appearance of various text components of a graph.

**Applies to**  
Properties of Graph controls, as noted throughout this discussion

**Syntax**  
Describe and Modify argument:

```
"graphname.property DispAttr.fontproperty { = value }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>graphname</code></td>
<td>The Graph control in a DataWindow for which you want to get or set font appearance values.</td>
</tr>
<tr>
<td><code>property</code></td>
<td>A text component of the graph, such as an Axis keyword (Category, Series, or Values), Legend, Pie, or Title, specifying the graph component whose appearance you want to get or set. These properties have their own entries. These values are listed in the following table. You can also set font properties for the label of an axis with the following syntax: <code>&quot;graphname.axis LabelDispAttr.fontproperty { = value }&quot;</code></td>
</tr>
<tr>
<td><code>fontproperty</code></td>
<td>A property that controls the appearance of text in the graph. Properties and their settings are listed in the table below.</td>
</tr>
<tr>
<td><code>value</code></td>
<td>The value to be assigned to <code>fontproperty</code>. Value can be a quoted DataWindow expression.</td>
</tr>
</tbody>
</table>

**Property for DispAttr**

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alignment</td>
</tr>
<tr>
<td>AutoSize</td>
</tr>
<tr>
<td>BackColor</td>
</tr>
<tr>
<td>DisplayExpression</td>
</tr>
</tbody>
</table>
## Property forDispAttr

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Font.CharSet</td>
<td><em>(exp)</em> An integer specifying the character set to be used. Values are: 0 – ANSI, 1 – The default character set for the specified font, 2 – Symbol, 128 – Shift JIS, 255 – OEM</td>
</tr>
<tr>
<td>Font.Escapement</td>
<td><em>(exp)</em> An integer specifying the rotation for the baseline of the text in tenths of a degree. For example, a value of 450 rotates the text 45 degrees. 0 is horizontal.</td>
</tr>
<tr>
<td>Font.Face</td>
<td><em>(exp)</em> A string specifying the name of the font face, such as Arial or Courier.</td>
</tr>
<tr>
<td>Font.Family</td>
<td><em>(exp)</em> An integer specifying the font family (Windows uses both face and family to determine which font to use). Values are: 0 – AnyFont, 1 – Roman, 2 – Swiss, 3 – Modern, 4 – Script, 5 – Decorative</td>
</tr>
<tr>
<td>Font.Height</td>
<td><em>(exp)</em> An integer specifying the height of the text in the unit of measure for the DataWindow. To specify size in points, specify a negative number. Not available when AutoSize is checked.</td>
</tr>
<tr>
<td>Font.Italic</td>
<td><em>(exp)</em> Whether the text should be italic. Values are: 0 – Not italic (default), 1 – Italic</td>
</tr>
<tr>
<td>Font.Orientation</td>
<td>Same as Escapement.</td>
</tr>
<tr>
<td>Font.Pitch</td>
<td><em>(exp)</em> The pitch of the font. Values are: 0 – The default pitch for your system, 1 – Fixed, 2 – Variable</td>
</tr>
<tr>
<td>Font.Strikethrough</td>
<td><em>(exp)</em> Whether the text should be crossed out. Values are: 0 – Not crossed out (default), 1 – Crossed out</td>
</tr>
</tbody>
</table>
### Chapter 3  DataWindow Object Properties

#### DataWindow Object Reference

**Usage**

In the painter  Select the control and set values in the Properties window, Text category. Settings apply to the selected item in the Text Object list box.

**Examples**

```visual-basic
setting = _
   dw1.Describe("Category.LabelDispAttr.Font.Face")

dw1.Modify("Category.LabelDispAttr.Font.Face='Arial'")
dw1.Modify("Title.DispAttr.DisplayExpression=
   "'Title + ~-n-'' + Today()"")
```

**DisplayType**

**Description**

The way the OLE Object control displays the OLE object it contains. It can display an icon or an image of the object’s contents. The image is reduced to fit inside the OLE container.

Both the icon and the image are provided by the OLE server. If the OLE server does not support a contents view, the DataWindow server displays an icon even if DisplayType is set to contents.

**Applies to**

OLE Object controls

**Syntax**

Describe and Modify argument:

```
"olecontrolname.DisplayType { = ' type ' }"
```

<table>
<thead>
<tr>
<th>Property for DispAttr</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Font.Underline</td>
<td>(exp) Whether the text should be underlined. Values are: 0 – Not underlined (default) 1 – Underlined</td>
</tr>
<tr>
<td>Font.Weight</td>
<td>(exp) An integer specifying the weight of the text, for example, 400 for normal or 700 for bold. Painter: Set indirectly using the Bold option.</td>
</tr>
<tr>
<td>Font.Width</td>
<td>(exp) An integer specifying the width of the font in the unit of measure specified for the DataWindow. Width is usually unspecified, which results in a default width based on the other properties.</td>
</tr>
<tr>
<td>Format</td>
<td>(exp) A string containing the display format for the text.</td>
</tr>
<tr>
<td>TextColor</td>
<td>(exp) A long specifying the color to be used for the text.</td>
</tr>
</tbody>
</table>

DataWindow Object Reference  223
In the painter  Select the control and set the value in the Properties window, Options category.

Examples

[Visual Basic]
ls_data = dw1.Describe("ole_report.DisplayType")
dw1.Modify("ole_report.DisplayType='1'")

**Edit.property**

Description  Settings that affect the appearance and behavior of columns whose edit style is Edit.

Applies to  Column controls

Syntax  DataWindow .NET dot notation:

* (((SimpleEdit) colname.EditStyle).property [C#])

CType(colname/EditStyle, SimpleEdit).property [Visual Basic]

Describe and Modify argument:

"columnname.Edit.property { = value }"

DataWindowSyntaxFromSql:

Column ( Edit.property = value )

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columnname</td>
<td>The column with the Edit edit style for which you want to get or set property values. You can specify the column name or a pound sign (#) and the column number.</td>
</tr>
<tr>
<td>property</td>
<td>A property for the column’s Edit style. Properties and their settings are listed in the table below. The table identifies the properties you can use with DataWindowSyntaxFromSql.</td>
</tr>
<tr>
<td>value</td>
<td>The value to be assigned to the property. For most Edit properties, you cannot specify a DataWindow expression. The exception is Edit.Format.</td>
</tr>
<tr>
<td>Property for Edit</td>
<td>Value</td>
</tr>
<tr>
<td>------------------</td>
<td>-------</td>
</tr>
<tr>
<td>AutoHScroll</td>
<td>Whether the edit control scrolls horizontally automatically when data is entered or deleted. &lt;br&gt;Values are: &lt;br&gt; Yes – Scroll horizontally automatically. &lt;br&gt; No – Do not scroll horizontally automatically. &lt;br&gt;You can use AutoHScroll with DataWindowSyntaxFromSql. The setting applies to all the columns in the generated syntax.</td>
</tr>
<tr>
<td>AutoSelect</td>
<td>Whether to select the contents of the edit control automatically when it receives focus. &lt;br&gt;Values are: &lt;br&gt; Yes – Select automatically. &lt;br&gt; No – Do not select automatically. &lt;br&gt;You can use AutoSelect with DataWindowSyntaxFromSql. The setting applies to all the columns in the generated syntax.</td>
</tr>
<tr>
<td>AutoVScroll</td>
<td>Whether the edit box scrolls vertically automatically when data is entered or deleted. &lt;br&gt;Values are: &lt;br&gt; Yes – Scroll vertically automatically. &lt;br&gt; No – Do not scroll vertically automatically. &lt;br&gt;You can use AutoVScroll with DataWindowSyntaxFromSql. The setting applies to all the columns in the generated syntax.</td>
</tr>
<tr>
<td>Case</td>
<td>The case of the text in the edit control. &lt;br&gt;Values are: &lt;br&gt; Any – Character of any case allowed. &lt;br&gt; Upper – Characters converted to uppercase. &lt;br&gt; Lower – Characters converted to lowercase.</td>
</tr>
<tr>
<td>CodeTable</td>
<td>Whether the column has a code table. &lt;br&gt;Values are: &lt;br&gt; Yes – Code table defined. &lt;br&gt; No – No code table defined.</td>
</tr>
<tr>
<td>DisplayOnly</td>
<td>Whether the column is display only. &lt;br&gt;Values are: &lt;br&gt; Yes – Do not allow the user to enter data; make the column display only. &lt;br&gt; No – (Default) Allow the user to enter data. &lt;br&gt;For conditional control over column editing, use the Protect property.</td>
</tr>
</tbody>
</table>
### Edit.property

<table>
<thead>
<tr>
<th>Property for Edit</th>
<th>Value</th>
</tr>
</thead>
</table>
| FocusRectangle    | Whether a dotted rectangle (the focus rectangle) will surround the current row of the column when the column has focus. Values are:  
  - Yes – (Default) Display the focus rectangle.  
  - No – Do not display the focus rectangle.  
You can use FocusRectangle with DataWindowSyntaxFromSql. The setting applies to all the columns in the generated syntax. |
| Format            | (exp) A string containing the display format of the edit control. The value for Format is quoted and can be a DataWindow expression. |
| HScrollBar        | Whether a horizontal scroll bar displays in the edit control. Values are:  
  - Yes – Display the horizontal scroll bar.  
  - No – Do not display the horizontal scroll bar. |
| Limit             | A number specifying the maximum number of characters (0 to 32,767) that the user can enter. 0 means unlimited. |
| Name              | A string whose value is the name of the predefined edit style associated with the column. Named styles are defined in the Database painter and can be reused. Specifying a name that has not been previously defined associates the name with the column but does not define a new edit style. |
| NilIsNull         | Whether to set the value of the edit control to null when the user leaves it blank. Values are:  
  - Yes – Make the Empty string null.  
  - No – Do not make the empty string null. |
| Password          | Whether to assign secure display mode to the column. When the user enters characters, they display as asterisks (*). Values are:  
  - Yes – Assign secure display mode to the column.  
  - No – Do not assign secure-display mode to the column.  
If you change the Password property, you should also change the Format property to display the results you want (for example, ****). |
| Required          | Whether the column is required. Values are:  
  - Yes – It is required.  
  - No – It is not required. |
| Style             | (Describe only) Returns the edit style of the column. Painter: EditStyle option. |
### Usage

**In the painter**  Select the control and set values in the Properties window, Behavior category, when EditStyle is Edit.

In DataWindow .NET, you can use the SimpleEdit class to set Edit style properties using dot notation. See the description of the SimpleEdit class in the online Help in Visual Studio .NET for a complete list of properties. Some properties have different names and are set differently. For example, the Case DataWindow object property is equivalent to the CharacterCasing property in DataWindow .NET, which uses an enumeration.

### Examples

```visual basic
[Visual Basic]
dw1.emp_name.Edit.Required = false
setting = dw1.Describe("emp_name.Edit.AutoHScroll")
dw1.Modify("emp_name.Edit.Required=no")
dw1.Modify("coll.Edit.UseEllipsis=Yes")
```

<table>
<thead>
<tr>
<th>Property for Edit</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>UseEllipsis</td>
<td>Whether an ellipsis (three dots) displays when a column with the Edit edit style contains character data that is too long for the display column in the DataWindow. The ellipsis does not display when the column has focus. Values are:</td>
</tr>
<tr>
<td></td>
<td>Yes – Truncate the data and add an ellipsis.</td>
</tr>
<tr>
<td></td>
<td>No – Truncate the data. Do not add an ellipsis.</td>
</tr>
<tr>
<td></td>
<td>The property is ignored if you:</td>
</tr>
<tr>
<td></td>
<td>• Set the Height.Autosize property.</td>
</tr>
<tr>
<td></td>
<td>• Specify an expression for the Font.Escapement property to rotate the text.</td>
</tr>
<tr>
<td></td>
<td>The UseEllipsis DataWindow object property is not supported in Web Forms applications.</td>
</tr>
<tr>
<td>ValidateCode</td>
<td>Whether the code table will be used to validate user-entered values.</td>
</tr>
<tr>
<td></td>
<td>Values are:</td>
</tr>
<tr>
<td></td>
<td>Yes – Use the code table.</td>
</tr>
<tr>
<td></td>
<td>No – Do not use the code table.</td>
</tr>
<tr>
<td>VScrollBar</td>
<td>Whether a vertical scroll bar displays in the line edit.</td>
</tr>
<tr>
<td></td>
<td>Values are:</td>
</tr>
<tr>
<td></td>
<td>Yes – Display vertical scroll bars.</td>
</tr>
<tr>
<td></td>
<td>No – Do not display vertical scroll bars.</td>
</tr>
</tbody>
</table>
**EditMask.property**

**Description**
Settings that affect the appearance and behavior of columns with the EditMask edit style.

**Applies to**
Column controls

**Syntax**
DataWindow .NET dot notation:

```csharp
((EditMask) colname/EditStyle).property
```

```visualbasic
CType(colname/EditStyle, EditMask).property
```

Describe and Modify argument:

"columnname.EditMask.property { = value }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>columnname</code></td>
<td>The column with the EditMask edit style for which you want to get or set property values. You can specify the column name or a pound sign (#) and the column number.</td>
</tr>
<tr>
<td><code>property</code></td>
<td>A property for the column’s EditMask style. Properties and their settings are listed in the table below.</td>
</tr>
<tr>
<td><code>value</code></td>
<td>The value to be assigned to the property. For EditMask properties, you cannot specify a DataWindow expression.</td>
</tr>
</tbody>
</table>

**Property for EditMask**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>AutoSkip</code></td>
<td>Whether the EditMask will automatically skip to the next field when the maximum number of characters has been entered.</td>
</tr>
<tr>
<td></td>
<td>Values are:</td>
</tr>
<tr>
<td></td>
<td>Yes – Skip automatically.</td>
</tr>
<tr>
<td></td>
<td>No – Do not skip automatically.</td>
</tr>
<tr>
<td><code>CodeTable</code></td>
<td>Whether the column has a code table.</td>
</tr>
<tr>
<td></td>
<td>Values are:</td>
</tr>
<tr>
<td></td>
<td>Yes – Code table defined.</td>
</tr>
<tr>
<td></td>
<td>No – No code table defined.</td>
</tr>
<tr>
<td>Property for EditMask</td>
<td>Value</td>
</tr>
<tr>
<td>----------------------</td>
<td>-------</td>
</tr>
<tr>
<td>DDCalendar</td>
<td>Whether a drop-down calendar control displays when a user clicks in a column with a Date or DateTime edit mask. Values are: Yes – Drop-down calendar control displays. No – (Default) Drop-down calendar control does not display. For Web DataWindows, to make sure that dates selected with the drop-down calendar option are displayed with the desired edit mask, you should specify that the Client Formatting option be included with the static JavaScript generated and deployed for the DataWindow. To conserve bandwidth, JavaScript for client formatting is not included by default. If you do not include script for client formatting, the drop-down calendar will use a default edit mask to display the column data based on the client machine's default localization settings.</td>
</tr>
<tr>
<td>DDCal_AlignRight</td>
<td>Whether the drop-down calendar is aligned with the right side of the column. Values are: Yes – Column is right aligned. No – (Default) Column is left aligned.</td>
</tr>
<tr>
<td>DDCalBackColor</td>
<td>The background color of the drop-down calendar. The default is Window Background.</td>
</tr>
<tr>
<td>DDCalTextColor</td>
<td>The color of text in the drop-down calendar. The default is Window Text.</td>
</tr>
<tr>
<td>DDCal_TitleBackColor</td>
<td>The background color of the title in the drop-down calendar. The default is Highlight.</td>
</tr>
<tr>
<td>DDCal_TitleTextColor</td>
<td>The color of text in the title of the drop-down calendar. The default is Highlight Text.</td>
</tr>
<tr>
<td>DDCal_TrailingTextColor</td>
<td>The color of trailing text (days in the previous and next months) in the drop-down calendar. The default is Disabled Text.</td>
</tr>
<tr>
<td>FocusRectangle</td>
<td>Whether a dotted rectangle (the focus rectangle) will surround the current row of the column when the column has focus. Values are: Yes – (Default) Display the focus rectangle. No – Do not display the focus rectangle.</td>
</tr>
<tr>
<td>Mask</td>
<td>A string containing the edit mask for the column.</td>
</tr>
<tr>
<td>ReadOnly</td>
<td>Whether the column is read-only. This property is valid only if EditMask.Spin is set to Yes. Values are: Yes – Do not allow the user to enter data; make the column read-only. No – (Default) Allow the user to enter data.</td>
</tr>
<tr>
<td>Required</td>
<td>Whether the column is required. Values are: Yes – It is required. No – It is not required.</td>
</tr>
</tbody>
</table>
## EditMask.property

<table>
<thead>
<tr>
<th>Property for EditMask</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DDCalendar</td>
<td>Whether a drop-down calendar control displays when a user clicks in a column with a Date or DateTime edit mask.</td>
</tr>
<tr>
<td></td>
<td>Values are:</td>
</tr>
<tr>
<td></td>
<td>- Yes – Drop-down calendar control displays.</td>
</tr>
<tr>
<td></td>
<td>- No – (Default) Drop-down calendar control does not display.</td>
</tr>
<tr>
<td></td>
<td>For Web DataWindows, to make sure that dates selected with the drop-down calendar option are displayed with the desired edit mask, you should specify that the Client Formatting option be included with the static JavaScript generated and deployed for the DataWindow. To conserve bandwidth, JavaScript for client formatting is not included by default.</td>
</tr>
<tr>
<td></td>
<td>If you do not include script for client formatting, the drop-down calendar will use a default edit mask to display the column data based on the client machine's default localization settings.</td>
</tr>
<tr>
<td>DDCal_AlignRight</td>
<td>Whether the drop-down calendar is aligned with the right side of the column.</td>
</tr>
<tr>
<td></td>
<td>Values are:</td>
</tr>
<tr>
<td></td>
<td>- Yes – Column is right aligned.</td>
</tr>
<tr>
<td></td>
<td>- No – (Default) Column is left aligned.</td>
</tr>
<tr>
<td>DDCal_BackColor</td>
<td>The background color of the drop-down calendar. The default is Window Background.</td>
</tr>
<tr>
<td>DDCal_TextColor</td>
<td>The color of text in the drop-down calendar. The default is Window Text.</td>
</tr>
<tr>
<td>DDCal_TitleBackColor</td>
<td>The background color of the title in the drop-down calendar. The default is Highlight.</td>
</tr>
<tr>
<td>DDCal_TitleTextColor</td>
<td>The color of text in the title of the drop-down calendar. The default is Highlight Text.</td>
</tr>
<tr>
<td>DDCal_TrailingTextColor</td>
<td>The color of trailing text (days in the previous and next months) in the drop-down calendar. The default is Disabled Text.</td>
</tr>
<tr>
<td>FocusRectangle</td>
<td>Whether a dotted rectangle (the focus rectangle) will surround the current row of the column when the column has focus.</td>
</tr>
<tr>
<td></td>
<td>Values are:</td>
</tr>
<tr>
<td></td>
<td>- Yes – (Default) Display the focus rectangle.</td>
</tr>
<tr>
<td></td>
<td>- No – Do not display the focus rectangle.</td>
</tr>
<tr>
<td>Mask</td>
<td>A string containing the edit mask for the column.</td>
</tr>
<tr>
<td>ReadOnly</td>
<td>Whether the column is read-only. This property is valid only if EditMask.Spin is set to Yes.</td>
</tr>
<tr>
<td></td>
<td>Values are:</td>
</tr>
<tr>
<td></td>
<td>- Yes – Do not allow the user to enter data; make the column read-only.</td>
</tr>
<tr>
<td></td>
<td>- No – (Default) Allow the user to enter data.</td>
</tr>
<tr>
<td>Required</td>
<td>Whether the column is required.</td>
</tr>
<tr>
<td></td>
<td>Values are:</td>
</tr>
<tr>
<td></td>
<td>- Yes – It is required.</td>
</tr>
<tr>
<td></td>
<td>- No – It is not required.</td>
</tr>
</tbody>
</table>
### Property for EditMask

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
</table>
| Spin         | Whether the user can scroll through a list of possible values for the column with a spin control.  
Values are:  
Yes – Display a spin control.  
No – (Default) Do not display a spin control.  
This setting has no effect in Web DataWindows. |
| SpinIncr     | An integer indicating the amount to increment the spin control’s values. The default for numeric values is 1; for dates, 1 year; and for time, 1 minute. Available for numeric, date, and time columns.  
For columns that are not numeric, date, or time, the spin control scrolls through values in an associated code table. If the EditMask.CodeTable property is No, the spin increment has no effect for these columns. |
| SpinRange    | A string containing the maximum and minimum values for the column that will display in the spin control. The two values are separated by a tilde (~). This property is effective only if EditMaskSpin is True. Available for numeric, date, and time columns.  
Because the SpinRange string is within another quoted string, the tilde separator becomes four tildes in DataWindow .NET, which reduces to a single tilde when parsed. The format for the string is:  
"EditMask.SpinRange = 'minval~~~~maxval'"|
| UseEllipsis  | Whether an ellipsis (three dots) displays when a column with the EditMask edit style contains character data that is too long for the display column in the DataWindow.  
The ellipsis does not display when the column has focus.  
Values are:  
Yes – Truncate the data and add an ellipsis.  
No – Truncate the data. Do not add an ellipsis.  
The property is ignored if you:  
• Set the Height.Autosize property.  
• Specify an expression for the Font.Escapement property to rotate the text.  
The UseEllipsis DataWindow object property is not supported in Web Forms applications. |
| UseFormat    | Whether a Format Display mask is used for a column’s display. A Format Display mask is used only when the column does not have focus.  
Values are:  
Yes – Use a Format Display mask.  
No – (Default) Do not use a Format Display mask. |

### Usage

**In the painter**  
Select the control and set values in the Properties window, Behavior category, when Style is EditMask.
In DataWindow .NET, you can use the EditMask class to set EditMask style properties using dot notation. See the description of the EditMask class in the online Help in Visual Studio .NET for a complete list of properties. Some properties have different names and are set differently. For example, spin values are set differently.

Examples

[Visual Basic]
setting = dwl.Describe("emp_status.EditMask.Spin")
dwl.Modify("empBonus.EditMask.SpinIncr=1000")
dwl.Modify("empBonus.EditMask.SpinRange='0~~~5000'")

[Visual Basic]
If TypeOf strtdt.EditStyle Is EditMask Then
  CType(strtdt.EditStyle, EditMask).Mask = _
    "yyyy-mm-dd"
  CType(strtdt.EditStyle, EditMask).Spin.Minimum _
    = "100"
  CType(strtdt/EditStyle, EditMask).Spin.Maximum _
    = "10000"
  CType(strtdt/EditStyle, EditMask).Spin.Increment _
    = 10
ElseIf ...

[C#]
if (strtdt.EditStyle is EditMask)
{
  ((EditMask)strtdt.EditStyle).Mask = "yyyy-mm-dd";
}
else
  ....

dwl.Modify("coll.EditMask.UseEllipsis=Yes")

---

Elevation

Description
The elevation in a 3D graph.

Applies to
Graph controls

Syntax
Describe and Modify argument:
"graphname.Elevation { = ' integer' }"
CHAPTER 3  DataWindow Object Properties

Usage

**In the painter**  Select the control and set the value in the Properties window, General category, Elevation (enabled when a 3D graph type is selected).

Examples

```visualbasic
setting = dw1.Describe("graph_1.Elevation")
dw1.Modify("graph_1.Elevation=35")
dw1.Modify("graph_1.Elevation='10-tIf(...,20,30)'")
```

### EllipseHeight

**Description**
The radius of the vertical part of the corners of a RoundRectangle.

**Applies to**
RoundRectangle controls

**Syntax**
Describe and Modify argument:

```
"rrectname.EllipseHeight { = ' integer' }
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>rrectname</code></td>
<td>The name of the RoundRectangle control in the DataWindow for which you want to get or set the ellipse height.</td>
</tr>
<tr>
<td><code>integer</code></td>
<td><em>(exp)</em> An integer specifying the radius of the vertical part of the corners of a RoundRectangle in the DataWindow’s unit of measure. EllipseHeight can be a quoted DataWindow expression.</td>
</tr>
</tbody>
</table>

**Usage**

**In the painter**  Select the control and set the value in the Properties window, General category.

**Examples**

```visualbasic
setting = dw1.Describe("rrect_1.EllipseHeight")
dw1.Modify("rrect_1.EllipseHeight=35")
dw1.Modify("rrect_1.EllipseHeight='10-tIf(...,20,30)'")
```
**EllipseWidth**

**Description**
The radius of the horizontal part of the corners of a RoundRectangle.

**Applies to**
RoundRectangle controls

**Syntax**
Describe and Modify argument:

```
"rrectname.EllipseWidth { = 'integer' }"
```

**Parameter** | **Description**
--- | ---
*rrectname* | The name of the RoundRectangle control in the DataWindow for which you want to get or set the ellipse width.
*integer* | *(exp)* An integer specifying the radius of the horizontal part of the corners of a RoundRectangle in the DataWindow’s unit of measure. EllipseWidth can be a quoted DataWindow expression.

**Usage**
In the painter
Select the control and set the value in the Properties window, General category.

**Examples**
[Visual Basic]
```
setting = dw1.Describe("rrect_1.EllipseWidth")
dw1.Modify("rrect_1.EllipseWidth=35")
dw1.Modify("rrect_1.EllipseWidth='10-tIf(...,20,30)'")
```

**Enabled**

**Description**
Determines whether a control in a DataWindow is enabled.

**Applies to**
Button, InkPicture controls

**Syntax**
Describe and Modify argument:

```
"buttonname.Enabled { = 'value' }"
```

**Parameter** | **Description**
--- | ---
*buttonname* | The name of the button that you want to enable or disable.
*value* | Whether the button is enabled. Values are:
  - Yes – (Default) The button is enabled.
  - No – The button is disabled.

**Usage**
In the painter
Select the control and set the value in the Properties window, Behavior category.
When the Enabled check box is cleared, or the Enabled property is otherwise set to false, the button control is grayed and its actions are not performed.

**Examples**

```java
setting = dw1.Describe("b_name.Enabled")
dw1.Modify("b_name.Enabled = 'No'")
```

---

**Export.PDF.Distill.CustomPostScript**

**Description**

Setting that enables you to specify the PostScript printer driver settings used when data is exported to PDF using the Distill! method.

**Applies to**

DataWindow objects

**Syntax**

Describe and Modify argument:

```
"DataWindow.Export.PDF.Distill.CustomPostScript { = 'value' }"
```

**Parameter** | **Description**
--- | ---
`value` | `(exp)` Whether the printer specified in the DataWindow.Printer property is used when data is exported to PDF.<br>Values are:<br>• 1 – The printer specified in DataWindow.Printer is used for PDF export.<br>• 0 – The default printer is used for PDF export (default).

**Usage**

The Distill! method performs a PostScript “print to file” before distilling to PDF. This property can be set to specify that you want to use a custom PostScript printer before you call the SaveAs method with PDF! as the SaveAsType or select File>Save Rows As with the file type PDF in the DataWindow painter.

Set this property if you want to use a PostScript printer driver for which you have set specific print options such as options for font and graphic handling. If this property is not set, a default PostScript printer driver specifically designed for distilling purposes is used.

This property has no effect if the Export.PDF.Method property is set to XSLFOP!.

**In the**

In the Data Export category in the Properties window for the DataWindow object, select PDF from the Export list and Distill! from the Method list, and then select Distill Custom PostScript.


Export.PDF.Method

Examples

This example uses Modify to set the PDF export properties and specify a network printer:

[Visual Basic]

```vbnet
dw1.Modify("DataWindow.Export.PDF.Method = Distill!"")
dw1.Modify("Printer = '\print-server\pr-18' ")
dw1.Modify _
("DataWindow.Export.PDF.Distill.CustomPostScript='1'")
```

See also

Export.PDF.Method

Export.PDF.Method

Description

Setting that determines whether data is exported to PDF from a DataWindow object by printing to a PostScript file and distilling to PDF, or by saving in XSL Formatting Objects (XSL-FO) format and processing to PDF.

The XSL-FO option is not supported in DataWindow .NET.

Applies to

DataWindow objects

Syntax

Describe and Modify argument:

"DataWindow.Export.PDF.Method { = 'value' }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>A string specifying a value of the PDFMethod enumerated datatype</td>
</tr>
</tbody>
</table>

See also

Export.PDF.Distill.CustomPostScript
Export.PDF.XSLFOP.Print

Export.PDF.XSLFOP.Print

Description

Setting that enables you to send a DataWindow object directly to a printer using platform-independent Java printing when using the XSL-FO method to export to PDF. This is an option of the Apache FOP processor.

This property is not supported in DataWindow .NET.

Applies to

DataWindow objects

Syntax

Describe argument:

"DataWindow.PDF.XSLFOP.Print { = 'value' }"
### Export.XHTML.TemplateCount

**Description**
The number of XHTML export templates associated with a DataWindow object.

**Applies to**
DataWindow objects

**Syntax**
Describe argument:

```plaintext
"DataWindow.Export.XHTML.TemplateCount"
```

**Usage**
This property is used to get a count of the XHTML export templates associated with a DataWindow object. It returns a long specifying the number of XHTML export templates previously saved in the DataWindow painter for the specified DataWindow object. The count is used with the `DataWindow.Export.XHTML.Template[i].Name` property to enable an application to select an export template at runtime.

**Examples**
The TemplateCount property gets the number of templates associated with a DataWindow object. You can use this number as the upper limit in a `FOR` loop that populates a drop-down list with the template names. The `FOR` loop uses the `Template[i].Name` property.

```plaintext
Dim count As String
Dim templateName As String
Dim i As Long

count=wdw.Describe("DataWindow.Export.XHTML.TemplateCount")

for i=1 to CLng(count)
    templateName = wdw.Describe("DataWindow.Export.XHTML.Template[" + Cstr(i) + "]Name")
```

### Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>value</code></td>
<td>(exp) Whether the exported PDF is sent directly to the default printer. Values are:</td>
</tr>
<tr>
<td></td>
<td>• Yes – The DataWindow object is exported to a PDF file and sent directly to a printer.</td>
</tr>
<tr>
<td></td>
<td>• No – The DataWindow object is exported to a PDF file but is not printed (default).</td>
</tr>
</tbody>
</table>

**See also**
Export.PDF.Method
Before generating the XHTML, set the export template using the value in the drop-down list box:

```csharp
wdw.SetProperty("DataWindow.Export.XHTML.UseTemplate", DDL1.SelectedValue())
```

See also

- Export.XHTML.Template[ ].Name
- Export.XHTML.UseTemplate

---

**Export.XHTML.Template[ ].Name**

**Description**
The name of an XHTML export template associated with a DataWindow object.

**Applies to**
DataWindow objects

**Syntax**
Describe argument:

"DataWindow.Export.XHTML.Template[num ]Name"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>num</td>
<td>(exp) A long specifying the index of the export template</td>
</tr>
</tbody>
</table>

**Usage**
This property returns the names of the XHTML export templates associated with a DataWindow object by index. The index can range from 1 to the value of the DataWindow.Export.XHTML.TemplateCount property. The order reflects the serialized storage order of all templates, which is a read-only setting. These properties, with DataWindow.Export.XHTML.UseTemplate, enable an application to select an export template dynamically at runtime.

**Examples**
See Export.XHTML.TemplateCount.

**See also**
- Export.XHTML.TemplateCount
- Export.XHTML.UseTemplate
Export.XHTML.UseTemplate

Description  Setting that optionally controls the logical structure of the XHTML generated by a DataWindow object from a DataWindow data expression.

Applies to  DataWindow objects

Syntax  Describe and Modify argument:

"DataWindow.Export.XHTML.UseTemplate { = 'value' }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| value     | (exp) A string specifying the name of an XHTML export template previously saved in the DataWindow painter for the specified DataWindow object

Usage  This property uses a template defined in the DataWindow painter to specify the logical structure and attribute overrides that the DataWindow server should use to generate XHTML from a DataWindow object. It is designed to be used with the data expression for the DataWindow object, and should be set before a data expression statement.

In the painter  In the Data Export category in the Properties window for the DataWindow object, select XHTML from the Export list and select a template from the Use Template list.

Examples  This example stores the name of the export template used in dw1 in the string ls_template. If no template is selected in dw1, an empty string is returned.

[V] [Visual Basic]

ls_template_name = dw1.Describe("DataWindow.Export.XHTML.UseTemplate")

This example sets the name of the current XHTML export template used in dw1 to t_report. If t_report does not exist, the current template is not changed.

dw1.Modify("DataWindow.Export.XHTML.UseTemplate = 't_report' ")

See also  Export.XHTML.TemplateCount
Export.XHTML.Template[ ].Name
Export.XML.HeadGroups

Description
Setting that causes elements, attributes, and all other items above the Detail Start element in an XML export template for a group DataWindow to be iterated for each group in the exported XML.

Applies to
DataWindow objects

Syntax
Describe and Modify argument:

"DataWindow.Export.XML.HeadGroups { = 'value' }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| value     | (exp) Whether the contents of the header section in an export template iterate in the generated XML. Values are:
  • Yes – The header section is repeated for each group (default).
  • No – The header section is not repeated. |

Usage
This property must be set for group DataWindow objects if you want elements and other items added to the header section of an XML export template to be repeated before each group in the exported XML. For DataWindow objects with multiple groups, each XML fragment in the header section between a Group Header element and the next Group Header element or Detail Start element is iterated.

In the painter
In the Data Export category in the Properties window for the DataWindow object, select XML from the Export list and select Iterate header for Groups.

Examples
[Visual Basic]
dw1.Modify("DataWindow.Export.XML.HeadGroups = 'No' ")

Export.XML.IncludeWhitespace

Description
Setting that determines whether the XML document is formatted by inserting whitespace characters (carriage returns, linefeeds, tabs, and spacebar spaces).

Applies to
DataWindow objects
### Syntax

Describe and Modify argument:

```
"DataWindow.Export.XML.IncludeWhitespace { = 'value' }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>(exp) Whether the generated XML is formatted with whitespace characters.</td>
</tr>
<tr>
<td></td>
<td>Values are:</td>
</tr>
<tr>
<td></td>
<td>• Yes – Whitespace characters are inserted.</td>
</tr>
<tr>
<td></td>
<td>• No – Whitespace characters are not inserted (default).</td>
</tr>
</tbody>
</table>

### Usage

This property should be set before you export a DataWindow object if you want to view or verify the exported XML using a text editor.

**In the painter** In the Data Export category in the Properties window for the DataWindow object, select XML from the Export list and select Include Whitespace.

**Examples**

```vbnet
dw1.Modify("DataWindow.Export.XML.IncludeWhitespace = 'Yes' ")
```

### Export.XML.MetaDataType

**Description** Setting that controls the type of metadata generated with the XML exported from a DataWindow object using the SaveAs method or a .Data.XML expression.

**Applies to** DataWindow objects

**Syntax** Describe and Modify argument:

```
"DataWindow.Export.XML.MetaDataType { = 'value' }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>(exp) A string specifying a value of the Export.XML.MetaDataType enumerated datatype</td>
</tr>
</tbody>
</table>

**Usage**

This property must be set to specify the type of metadata generated before you call the SaveAs method with XML! as the SaveAsType to save data as an XML document, or use the .Data.XML expression to save data as an XML string. The metadata is saved into the exported XML itself or into an associated file, depending on the value of the Export.XML.SaveMetaData property.
The Export.XML.MetaDataType property is an enumerated datatype that can hold the following values:

<table>
<thead>
<tr>
<th>Enumerated value</th>
<th>Numeric value</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMLNone!</td>
<td>0</td>
<td>Metadata (XML Schema or DTD) is not generated when XML is exported</td>
</tr>
<tr>
<td>XMLSchema!</td>
<td>1</td>
<td>XML Schema is generated when XML is exported</td>
</tr>
<tr>
<td>XMLDTD!</td>
<td>2</td>
<td>DTD is generated when XML is exported</td>
</tr>
</tbody>
</table>

**In the painter**  In the Data Export category in the Properties window for the DataWindow object, select XML from the Export list and select a value from the Meta Data Type list.

**Examples**  These statements export the contents of dw1 to the file c:\myxml.xml using the XML export template called t_schema, and generate an external XML schema file at c:\myxml.xsd:

```vbnet
[Visual Basic]

dw1.Modify("DataWindow.Export.XML.UseTemplate = 't_schema'")
dw1.Modify("DataWindow.Export.XML.MetaDataType = 1")
dw1.Modify("DataWindow.Export.XML.SaveMetaData = 1")
dw1.SaveAs("c:\myxml.xml", XML!, false)
```

**See also** Export.XML.SaveMetaData

### Export.XML.SaveMetaData

**Description**  Setting that controls the storage format for the metadata generated with the XML exported from a DataWindow object using the SaveAs method or a .Data.XML expression.

**Applies to**  DataWindow objects

**Syntax**  Describe and Modify argument:

```vbnet
"DataWindow.Export.XML.SaveMetaData { = 'value' }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>(exp) A string specifying a value of the Export.XML.SaveMetaData enumerated datatype</td>
</tr>
</tbody>
</table>
Usage
This property must be set to specify how to store the generated metadata before
you call the SaveAs method with XML! as the SaveAsType to save data as an
XML document, or use the .Data.XML expression to save data as an XML
string. The metadata can be saved into the exported XML document or string
or into an associated file.

Note
If Export.XML.MetaDataType is set to XMLNone!, the value of the
Export.XML.SaveMetaData property is not used.

The Export.XML.SaveMetaData property is an enumerated datatype that can
hold the following values:

<table>
<thead>
<tr>
<th>Enumerated value</th>
<th>Numeric value</th>
<th>Meaning</th>
</tr>
</thead>
</table>
| MetaDataInternal! | 0            | The metadata is saved into the generated XML
document or string. To save metadata using the
.Data.XML expression syntax, you must use this
value. |
| MetaDataExternal! | 1            | With the SaveAs method, metadata is saved as an
external file with the same name as the XML
document but with the extension .xsd (for
XMLSchema! type) or .did (for XMLDTD! type).
A reference to the name of the metadata file is
included in the output XML document.
With .Data.XML, no metadata is generated in the
XML string. |

In the painter In the Data Export category in the Properties window for the
DataWindow object, select XML from the Export list and select a value from
the Save Meta Data list.

Examples [Visual Basic]
dw1.Modify("DataWindow.Export.XML.SaveMetaData =
MetaDataExternal!")

See also Export.XML.MetaDataType
Export.XML.TemplateCount

Description The number of XML export templates associated with a DataWindow object.
Applies to DataWindow objects
Syntax Describe argument:
   "DataWindow.Export.XML.TemplateCount"
Usage This property is used to get a count of the XML export templates associated with a DataWindow object. It returns a long specifying the number of XML export templates previously saved in the DataWindow painter for the specified DataWindow object. The count is used with the DataWindow.Export.XML.Template[ ].Name property to enable an application to select an export template at runtime.
See also Export.XML.Template[ ].Name
Export.XML.UseTemplate

Export.XML.Template[ ].Name

Description The name of an XML export template associated with a DataWindow object.
Applies to DataWindow objects
Syntax Describe argument:
   "DataWindow.Export.XML.Template[num ]Name"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>num</td>
<td>(exp) A long specifying the index of the export template</td>
</tr>
</tbody>
</table>

Usage This property is used to get the names of the XML export templates associated with a DataWindow object. It returns a string specifying the name of an export template previously saved in the DataWindow painter for the specified DataWindow object. The property is used with the DataWindow.Export.XML.TemplateCount property to enable an application to select an export template at runtime.
Examples See Export.XML.TemplateCount.
See also Export.XML.TemplateCount
Export.XML.UseTemplate
Export.XML.UseTemplate

Description  Setting that optionally controls the logical structure of the XML exported from a DataWindow object using the SaveAs method or the .Data.XML property.

Applies to  DataWindow objects

Syntax  Describe and Modify argument:

"DataWindow.Export.XML.UseTemplate { = 'value' }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>(exp) A string specifying the name of an export template previously saved in the DataWindow painter for the specified DataWindow object</td>
</tr>
</tbody>
</table>

Usage  This property should be set to specify the logical structure of the XML generated before you call the SaveAs method with XML! as the SaveAsType to save data as an XML document, or use the .Data.XML expression to save data as an XML string.

In the painter  In the Data Export category in the Properties window for the DataWindow object, select XML from the Export list and select a template from the Use Template list.

Examples  This example stores the name of the export template used in dw1 in the string ls_template. If no template is selected in dw1, an empty string is returned.

[Visual Basic]
ls_template_name = dw1.Describe("DataWindow.Export.XML.UseTemplate")

This example sets the name of the current XML export template used in dw1 to t_report. If t_report does not exist, the current template is not changed.

dw1.Modify("DataWindow.Export.XML.UseTemplate = 't_report' ")

See also  Export.XML.MetaDataType  Export.XML.SaveMetaData
Expression

**Description**
The expression for a computed field control in the DataWindow. The expression is made up of calculations and DataWindow expression functions. The DataWindow evaluates the expression to get the value it will display in the computed field.

**Applies to**
Computed field controls

**Syntax**
Describe and Modify argument:

```
"computename.Expression { = 'string' }
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>computename</td>
<td>The name of the computed field control in the DataWindow for which you want to get or set the expression</td>
</tr>
<tr>
<td>string</td>
<td>A string whose value is the expression for the computed field</td>
</tr>
</tbody>
</table>

**Usage**

**In the painter**
Select the control and set the value in the Properties window, General category. The button displays the Modify Expression dialog, which provides help in specifying the expression. The Verify button tests the expression.

**Examples**

```
[Visual Basic]
setting = dw1.Describe("comp_1.Expression")
dw1.Modify("comp_1.Expression='avg(salary for all)'")
```

Filename

**Description**
The file name containing the image for a Picture or Button control in the DataWindow. If no image is specified for a Button control, only text is used for the button label.

**Applies to**
Picture and Button controls

**Syntax**
Describe and Modify argument:

```
"controlname.Filename { = 'filestring' }
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlname</td>
<td>The name of the Picture or Button control in the DataWindow for which you want to get or set the image file name.</td>
</tr>
</tbody>
</table>
CHAPTER 3  DataWindow Object Properties

Usage

In the painter  Select the control and set the value in the Properties window, Appearance category.

Examples

Example for a Picture control:

[Visual Basic]
ssetting = dw1.Describe("bitmap_1.Filename")
dw1.Modify("bitmap_1.Filename='exclaim.bmp'")

Example for a Button control:

[Visual Basic]
ls_data = dw1.Describe("b_name.FileName")
dw1.Modify("b_name.FileName = 'logo.jpg'")

See also

DefaultPicture

FirstRowOnPage

Description

The first row currently visible in the DataWindow.

Applies to

DataWindows

Syntax

DataWindow .NET dot notation:

dw_control.FirstNameOnPage

Describe argument:

"DataWindow.FirstNameOnPage"

Examples

[Visual Basic]
Dim First As Integer
First = dw1.FirstNameOnPage

setting = dw1.Describe("DataWindow.FirstNameOnPage")

Parameter | Description
--- | ---
filestring | *(exp)* A string containing the name of the file that contains the image. *Filestring* can be a quoted DataWindow expression. Button pictures can be BMP, GIF, or JPEG files. You can use a URL instead of a full path name, and if you set the HTMLGen.ResourceBase property to the URL address, you need to specify only a relative file name for this string. If you include the name of the file containing the image in the executable for the application, the DataWindow server will always use that image; you cannot use Modify to change the image.
Font.Bias

Description
The way fonts are manipulated in the DataWindow at runtime.

Applies to
DataWindows

Syntax
Describe and Modify argument:
"DataWindow.Font.Bias { = biasvalue }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>biasvalue</td>
<td>An integer indicating how the fonts will be manipulated at execution. Biasvalue cannot be a DataWindow expression. Values are: 0 – As display fonts 1 – As printer fonts 2 – Neutral; no manipulation will take place</td>
</tr>
</tbody>
</table>

Examples
[Visual Basic]

setting = dw1.Describe("DataWindow.Font.Bias")
dw1.Modify("DataWindow.Font.Bias=1")

Font.property

Description
Settings that control the appearance of fonts within a DataWindow, except for graphs, which have their own settings (see DispAttr).

Applies to
Button, Column, Computed Field, GroupBox, and Text controls

Syntax
Describe and Modify argument:
"controlname.Font.property { = ' value ' }"

DataWindowSyntaxFromSql:

Column(Font.property = value)
Text(Font.property = value)

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlname</td>
<td>The name of a column, computed field, or text control for which you want to get or set font properties. For a column, you can specify the column name or a pound sign (#) followed by the column number. When you generate DataWindow syntax with DataWindowSyntaxFromSql, the Font settings apply to all columns or all text controls.</td>
</tr>
</tbody>
</table>
### Parameter | Description
--- | ---
property | A property of the text. The properties and their values are listed in the table below.
value | The value to be assigned to the property. Value can be a quoted DataWindow expression.

#### Property for Font | Value
--- | ---
CharSet | \((exp)\) An integer specifying the character set to be used. (Not supported in DataWindow Designer.)
Values are:
0 – ANSI
1 – The default character set for the specified font
2 – Symbol
128 – Shift JIS
255 – OEM

Escapement | \((exp)\) An integer specifying the rotation for the baseline of the text in tenths of a degree. For example, a value of 450 rotates the text 45 degrees. 0 is horizontal.

Face | \((exp)\) A string specifying the name of the font face, such as Arial or Courier.

Family | \((exp)\) An integer specifying the font family (Windows uses both face and family to determine which font to use). (Not supported in DataWindow Designer.)
Values are:
0 – AnyFont
1 – Roman
2 – Swiss
3 – Modern
4 – Script
5 – Decorative

Height | \((exp)\) An integer specifying the height of the text in the unit measure for the DataWindow. To specify size in points, specify a negative number.

Italic | \((exp)\) Whether the text should be italic. The default is no.

Pitch | \((exp)\) The pitch of the font. (Not supported in DataWindow Designer.)
Values are:
0 – The default pitch for your system
1 – Fixed
2 – Variable

Strikethrough | \((exp)\) Whether the text should be crossed out. The default is no.

Underline | \((exp)\) Whether the text should be underlined. The default is no.

Weight | \((exp)\) An integer specifying the weight of the text; for example, 400 for normal or 700 for bold.
Painter: Set indirectly using the Bold option.
### Footer.property

<table>
<thead>
<tr>
<th>Property for Font</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Width</td>
<td>(exp) An integer specifying the average character width of the font in the unit of measure specified for the DataWindow. Width is usually unspecified, which results in a default width based on the other properties. Painter: Set indirectly using the font selection.</td>
</tr>
</tbody>
</table>

#### Usage

**In the painter** Select the control and set the value using the Properties window, Font category:

#### Examples


### Footer.property

See Bandname.property.

### Format

**Description**

The display format for a column.

You can use the GetFormat and SetFormat methods instead of Describe and Modify to get and change a column’s display format. The advantage to using Modify is the ability to specify an expression.

**Applies to**

Column and Computed Field controls

**Syntax**

Describe and Modify argument:

```
"controlname.Format { = 'value' }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlname</td>
<td>The name of the column or computed field for which you want to get or set the display format.</td>
</tr>
<tr>
<td>value</td>
<td>(exp) A string specifying the display format. See the User’s Guide for information on constructing display formats. Value can be a quoted DataWindow expression.</td>
</tr>
</tbody>
</table>
CHAPTER 3 DataWindow Object Properties

Usage
In the painter Select the control and set the value in the Properties window, Behavior category.

If you want to add text to a numeric display format and use a color attribute, you must include the escape character (\) before each literal in the mask. For example:

```
[red]\D\e\p\t\: ###
```

Examples
[Visual Basic]
```
setting = dw1.Describe("phone.Format")
dw1.Modify('  
"phone.Format='[red](@@@)@@@-@@@@;~~~'None~~~''
```

GraphType
Description The type of graph, such as bar, pie, column, and so on.
Applies to Graph controls
Syntax Describe and Modify argument:
```
"graphname.GraphType { = ' typeinteger' }
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>graphname</code></td>
<td>The graph control for which you want to get or change the type.</td>
</tr>
<tr>
<td><code>typeinteger</code></td>
<td>(exp) An integer identifying the type of graph in the DataWindow object. Typeinteger can be a quoted DataWindow expression.</td>
</tr>
</tbody>
</table>

Values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Graph Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Area</td>
</tr>
<tr>
<td>2</td>
<td>Bar</td>
</tr>
<tr>
<td>3</td>
<td>Bar3D</td>
</tr>
<tr>
<td>4</td>
<td>Bar3DObj</td>
</tr>
<tr>
<td>5</td>
<td>BarStacked</td>
</tr>
<tr>
<td>6</td>
<td>BarStacked3DObj</td>
</tr>
<tr>
<td>7</td>
<td>Col</td>
</tr>
<tr>
<td>8</td>
<td>Col3D</td>
</tr>
<tr>
<td>9</td>
<td>Col3DObj</td>
</tr>
<tr>
<td>10</td>
<td>ColStacked</td>
</tr>
<tr>
<td>11</td>
<td>ColStacked3DObj</td>
</tr>
<tr>
<td>12</td>
<td>Line</td>
</tr>
<tr>
<td>13</td>
<td>Pie</td>
</tr>
<tr>
<td>14</td>
<td>Scatter</td>
</tr>
<tr>
<td>15</td>
<td>Area3D</td>
</tr>
<tr>
<td>16</td>
<td>Line3D</td>
</tr>
<tr>
<td>17</td>
<td>Pie3D</td>
</tr>
</tbody>
</table>

Usage
In the painter Select the control and set the value in the Properties window, General category.

Examples
[Visual Basic]
```
setting = dw1.Describe("graph_1.GraphType")
dw1.Modify("graph_1.GraphType=17")
```
**Grid.ColumnMove**

**Description**: Whether the user can rearrange columns by dragging.

**Applies to**: DataWindows

**Syntax**

```
"DataWindow.Grid.ColumnMove { = value }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Whether the user can rearrange columns. Values are: Yes – The user can drag columns. No – The user cannot drag columns.</td>
</tr>
</tbody>
</table>

**Usage**

*In the painter* Select the DataWindow object by deselecting all controls; then set the value in the Properties window, General category (available when the presentation style is Grid, Crosstab, or TreeView with the Grid Style option selected).

**Examples**

```Visual Basic
setting = dw1.Describe("DataWindow.Grid.ColumnMove")
dw1.Modify("DataWindow.Grid.ColumnMove=No")
```

**Grid.Lines**

**Description**

The way grid lines display and print in a DataWindow whose presentation style is Grid, Crosstab, or TreeView.

**Applies to**

DataWindows

**Syntax**

```
"DataWindow.Grid.Lines { = value }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>An integer specifying whether grid lines are displayed on the screen and printed. Values are: 0 – Yes, grid lines are displayed and printed. 1 – No, grid lines are not displayed and printed. 2 – Grid lines are displayed, but not printed. 3 – Grid lines are printed, but not displayed.</td>
</tr>
</tbody>
</table>
**Usage**

**In the painter**  Select the DataWindow object by deselecting all controls; then set the value in the Properties window, General category (available when the presentation style is Grid, Crosstab, or TreeView with the Grid Style option selected).

**Examples**

[Visual Basic]

```vbnet
setting = dw1.Describe("DataWindow.Grid.Lines")
dw1.Modify("DataWindow.Grid.Lines=2")
```

---

**GroupBy**

**Description**

A comma-separated list of the columns or expressions that control the grouping of the data transferred from the DataWindow to the OLE object. When there is more than one grouping column, the first one is the primary group and the columns that follow are nested groups.

**Applies to**

OLE Object controls

**Syntax**

Describe and Modify argument:

```
"olecontrolname.GroupBy { = 'columnlist' }
```

**Parameter** | **Description**
--- | ---
`olecontrolname` | The name of the OLE Object control for which you want to get or set the grouping columns.
`columnlist` | `(exp)` A list of the columns or expressions that control the grouping. If there is more than one, separate them with commas. `Columnlist` can be a quoted DataWindow expression.

**Usage**

Target and Range also affect the data that is transferred to the OLE object.

**In the painter**  Select the control and set the value in the Properties window, Data category.

**Examples**

[Visual Basic]

```vbnet
dw1.Modify(" _
    ole_report.GroupBy='emp_state, emp_office'")
dw1.Modify("ole_report.GroupBy='year'")
```
**Header_Bottom_Margin**

**Description**

The size of the bottom margin of the DataWindow’s header area.  
Header_Bottom_Margin is meaningful only when type is Grid or Tabular.

**Applies to**

Style keywords

**Syntax**

DataWindowSyntaxFromSql:

```
Style ( Header_Bottom_Margin = value )
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>An integer specifying the size of the bottom margin of the header area in the units specified for the DataWindow. The bottom margin is the distance between the bottom of the header area and the last line of the header.</td>
</tr>
</tbody>
</table>

**Examples**

[Visual Basic]

DWSyntaxGenerator.DataWindowSyntaxFromSql(myTrans, sqlstring, 'Style(...Header_Bottom_Margin = 25 ...)')

---

**Header_Top_Margin**

**Description**

The size of the top margin of the DataWindow’s header area.  
Header_Top_Margin is meaningful only when type is Grid or Tabular.

**Applies to**

Style keywords

**Syntax**

DataWindowSyntaxFromSql:

```
Style ( Header_Top_Margin = value )
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>An integer specifying the size of the top margin of the header area in the units specified for the DataWindow. The top margin is the distance between the top of the header area and the first line of the header.</td>
</tr>
</tbody>
</table>

**Examples**

[Visual Basic]

DWSyntaxGenerator.DataWindowSyntaxFromSql(myTrans, sqlstring, 'Style(...Header_Top_Margin = 500 ...)')

---

**Header.property**

See Bandname.property.


### Header.#.property

See Bandname.property.

### Height

**Description**

The height of a control in the DataWindow.

**Applies to**

Button, Column, Computed Field, Graph, GroupBox, OLE, Oval, Picture, Rectangle, Report, RoundRectangle, TableBlob, and Text controls

**Syntax**

Describe and Modify argument:

```
"controlname.Height { = 'value' }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlname</td>
<td>The control within the DataWindow whose height you want to get or set.</td>
</tr>
<tr>
<td>value</td>
<td>(exp) An integer specifying the height of the control in the unit of measure specified for the DataWindow. Value can be a quoted DataWindow expression.</td>
</tr>
</tbody>
</table>

**Usage**

**In the painter** Select the control and set the value in the Properties window, Layout category.

**Examples**

[Visual Basic]

```vbs
setting = dw1.Describe("empname.Height")
dw1.Modify("empname.Height=50")
```

### Height.AutoScale

**Description**

Whether the control’s width should be held constant and its height adjusted so that all the data is visible. This property is for use with read-only controls and printed reports. It should not be used with data entry fields or controls.

**Applies to**

Column, Computed Field, Report, and Text controls

**Syntax**

Describe and Modify argument:

```
"controlname.Height.AutoScale { = value }
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlname</td>
<td>The control for which you want to get or set the AutoSize property.</td>
</tr>
</tbody>
</table>
Help.property

<table>
<thead>
<tr>
<th><strong>Parameter</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Whether the width or height of the control will be adjusted to display all the data. The height is limited to what can fit on the page. Values are: No – Use the height defined in the painter. Yes – Calculate the height so that all the data is visible.</td>
</tr>
</tbody>
</table>

**Usage**

**In the painter** Select the control and set the value in the Properties window, Layout category.

**Minimum height** The height of the column, computed field, or text will never be less than the minimum height (the height selected in the painter).

When the band has Autosize Height set to true, you should avoid using the RowHeight DataWindow expression function to set the height of any element in the row. Doing so can result in a logical inconsistency between the height of the row and the height of the element. For more information, see the RowHeight function description.

**Examples**

[Visual Basic]

```vbnet
setting = dw1.Describe("empname.Height.AutoSize")
dw1.Modify("empname.Height.AutoSize=Yes")
```

**See also** Bandname.property

**Help.property**

**Description** Settings for customizing the Help topics associated with DataWindow dialog boxes.

**Applies to** DataWindows

**Syntax** Describe and Modify argument:

```
"DataWindow.Help.property { = value }"
```

<table>
<thead>
<tr>
<th><strong>Parameter</strong></th>
<th><strong>Description</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>property</td>
<td>A property for specifying DataWindow Help. Help properties and their settings are listed in the table below. The File property must have a valid file name before the rest of the Help property settings can become valid.</td>
</tr>
<tr>
<td>value</td>
<td>The value to be assigned to the property. For Help properties, value cannot be a DataWindow expression.</td>
</tr>
</tbody>
</table>
### CHAPTER 3  DataWindow Object Properties

**Property for Help** | **Value**
---|---
**Command** | An integer specifying the type of Help command that is specified in the following TypeID properties. Values are: 0 – Index 1 – TopicID 2 – Search keyword
**File** | A string containing the fully qualified name of the compiled Help file (for example, C:\proj\MYHELP.HLP). When this property has a value, Help buttons display on the DataWindow dialog boxes at runtime.
**TypeID** | A string specifying the default Help command to be used when a Help topic is not specified for the dialog using one of the following eight dialog-specific properties listed in this table.
**TypeID. ImportFile** | A string specifying the Help topic for the Import File dialog box, which might display when the ImportFile method is called in code.
**TypeID.Retrieve. Argument** | A string specifying the Help topic for the Retrieval Arguments dialog box, which displays when retrieval arguments expected by the DataWindow’s SELECT statement are not specified for the Retrieve method in code.
**TypeID.Retrieve. Criteria** | A string specifying the Help topic for the Prompt for Criteria dialog box, which displays when the Criteria properties have been turned on for at least one column and the Retrieve method is called in code.
**TypeID.SaveAs** | A string specifying the Help topic for the Save As dialog box, which might display when the SaveAs method is called in code.
**TypeID. SetCrosstab** | A string specifying the Help topic for the Crosstab Definition dialog box, which might display when the CrosstabDialog method is called in code.
**TypeID.SetFilter** | A string specifying the Help topic for the Set Filter dialog box, which might display when the SetFilter and Filter methods are called in code.
**TypeID.SetSort** | A string specifying the Help topic for the Set Sort dialog box, which might display when the SetSort and Sort methods are called in code.
**TypeID. SetSortExpr** | A string specifying the Help topic for the Modify Expression dialog, which displays when the user double-clicks on a column in the Set Sort dialog.

**Usage**

**In the painter**  Can be set only in code, not in the painter.

**Examples**

```Visual Basic
setting = dw1.Describe("DataWindow.Help.Command")
dw1.Modify("DataWindow.Help.File='myhelp.hlp'")
dw1.Modify("DataWindow.Help.Command=1")
dw1.Modify("DataWindow.Help.TypeID.SetFilter = 'filter_topic'")
dw1.Modify("DataWindow.Help.TypeID.Retrieve.Criteria = 'criteria_topic'")
```
**HideGrayLine**

**Description**
Shows or hides a gray line to indicate that a fixed page has been crossed when scrolling in a DataWindow with group headers.

**Applies to**
DataWindow control

**Syntax**
Describe and Modify argument:

"DataWindow.HideGrayLine { = ' value ' }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>( (exp) ) Whether a gray line displays in the Preview view and at runtime. Values are: Yes – The gray line is hidden. No – The gray line displays (default). Value can be a quoted DataWindow expression.</td>
</tr>
</tbody>
</table>

**Usage**
This property can be set in the open event for the window in which the DataWindow displays. Note that you cannot suppress the display of repeating group headers.

**In the painter**
Select the DataWindow object by deselecting all controls; then set the value in the Properties window, General category. This option is enabled only for DataWindows with group headers.

**HideSnaked**

**Description**
Whether the control appears only once per page when you print the DataWindow using the newspaper columns format.

**Applies to**
Button, Column, Computed Field, Graph, GroupBox, Line, OLE, Oval, Picture, Rectangle, Report, RoundRectangle, TableBlob, and Text controls

**Syntax**
Describe and Modify argument:

"controlname.HideSnaked { = ' value ' }"
Usage

In the painter  Select the control and set the value in the Properties window, Appearance category.

Examples

[Visual Basic]
setting = dw1.Describe("graph_1.HideSnaked")
dw1.Modify("text_title.HideSnaked=1")

---

**Horizontal_Spread**

Description  The space between columns in the detail area of the DataWindow object.

Horizontal_Spread is meaningful only when type is Grid or Tabular.

Applies to  Style keywords

Syntax  DataWindowSyntaxFromSql:

```
Style ( Horizontal_Spread = value )
```

Parameter | Description
---|---
*value* | An integer specifying the space between columns in the detail area of the DataWindow object area in the units specified for the DataWindow

Examples

[Visual Basic]
DWSyntaxGenerator.DataWindowSyntaxFromSql(myTrans, sqlstring, 'Style(...Horizontal_Spread = 25 ...)')
**HorizontalScrollMaximum**

**Description**
The maximum width of the scroll box of the DataWindow’s horizontal scroll bar. This value is set by the DataWindow server based on the layout of the DataWindow object and the size of the DataWindow control. Use HorizontalScrollMaximum with HorizontalScrollPosition to synchronize horizontal scrolling in multiple DataWindow objects.

**Applies to**
DataWindows

**Syntax**
Describe argument:
```
"DataWindow.HorizontalScrollMaximum"
```

**Examples**

```
[Visual Basic]
setting = dw1.Describe("DataWindow.HorizontalScrollMaximum")
```

---

**HorizontalScrollMaximum2**

**Description**
The maximum width of the second scroll box when the horizontal scroll bar is split (HorizontalScrollSplit is greater than 0). This value is set by the DataWindow server based on the content of the DataWindow. Use HorizontalScrollMaximum2 with HorizontalScrollPosition2 to synchronize horizontal scrolling in multiple DataWindow objects.

**Applies to**
DataWindows

**Syntax**
Describe argument:
```
"DataWindow.HorizontalScrollMaximum2"
```

**Examples**

```
[Visual Basic]
setting = dw1.Describe("DataWindow.HorizontalScrollMaximum2")
```

---

**HorizontalScrollPosition**

**Description**
The position of the scroll box in the horizontal scroll bar. Use HorizontalScrollMaximum with HorizontalScrollPosition to synchronize horizontal scrolling in multiple DataWindow objects.

**Applies to**
DataWindows

**Syntax**
Describe and Modify argument:
```
"DataWindow.HorizontalScrollPosition { = scrollvalue }
```

260  DataWindow .NET
CHAPTER 3  DataWindow Object Properties

DataWindow Object Reference

Examples

[Visual Basic]
Dim smax1, smax2, spos1, modstring As String
Dim pos2 As Integer
smax1 = dw1.Describe(_
"DataWindow.HorizontalScrollMaximum")
spos1 = dw1.Describe(_
"DataWindow.HorizontalScrollPosition")
smax2 = dw_2.Describe(_
"DataWindow.HorizontalScrollMaximum")
pos2 = CInt(spos1) * CInt(smax2) / CInt(smax1)
modstring = "DataWindow.HorizontalScrollPosition" _
+ Str(pos2)
dw1.Modify(modstring)

**HorizontalScrollPosition2**

**Description**
The position of the scroll box in the second portion of the horizontal scroll bar when the scroll bar is split (HorizontalScrollSplit is greater than 0). Use HorizontalScrollMaximum2 with HorizontalScrollPosition2 to synchronize horizontal scrolling in multiple DataWindow objects.

**Applies to**
DataWindows

**Syntax**
Describe and Modify argument:

"DataWindow.HorizontalScrollPosition2 { = scrollvalue }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scrollvalue</td>
<td>An integer specifying the position of the scroll box in the second portion of a split horizontal scroll bar of the DataWindow</td>
</tr>
</tbody>
</table>

**Examples**

[Visual Basic]
spos = dw1.Describe(_
"DataWindow.HorizontalScrollPosition2")
dw1.Modify(_
"DataWindow.HorizontalScrollPosition2=200")
**HorizontalScrollSplit**

Description: The position of the split in the DataWindow’s horizontal scroll bar. If HorizontalScrollSplit is zero, the scroll bar is not split.

Applies to: DataWindows

Syntax: Describe and Modify argument:

```
"DataWindow.HorizontalScrollSplit { = splitdistance }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>splitdistance</td>
<td>An integer indicating where the split will occur in the horizontal scroll bar in a DataWindow object in the unit of measure specified for the DataWindow object</td>
</tr>
</tbody>
</table>

Examples

[Visual Basic]

```
str = dw1.Describe("DataWindow.HorizontalScrollSplit")
dw1.Modify("DataWindow.HorizontalScrollSplit=250")
```

**HTextAlign**

Description: The way text in a button is horizontally aligned.

Applies to: Button controls

Syntax: Describe and Modify argument:

```
"buttonname.HTextAlign { = 'value' }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>buttonname</td>
<td>The name of the button for which you want to align text.</td>
</tr>
<tr>
<td>value</td>
<td>An integer indicating how the button text is horizontally aligned.</td>
</tr>
<tr>
<td></td>
<td>Values are:</td>
</tr>
<tr>
<td></td>
<td>0 – Center</td>
</tr>
<tr>
<td></td>
<td>1 – Left</td>
</tr>
<tr>
<td></td>
<td>2 – Right</td>
</tr>
</tbody>
</table>

Usage: In the painter Select the control and set the value in the Properties window, Appearance category.

Examples

[Visual Basic]

```
setting = dw1.Describe("b_name.HTextAlign")
dw1.Modify("b_name.HTextAlign ='1'")
```
**HTML.property**

**Description**
Settings for adding user-defined HTML syntax and hyperlinks to controls in a Web DataWindow.

**Applies to**
Column, Computed Field, Picture, and Text controls

**Syntax**
Describe and Modify argument:

```
"controlname.HTML.property { = 'value' }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlname</td>
<td>The name of the control whose HTML properties you want to get or set.</td>
</tr>
<tr>
<td>property</td>
<td>A property for generating HTML syntax and hyperlinks in a Web DataWindow. Properties and their values are listed in the table below.</td>
</tr>
<tr>
<td>value</td>
<td>The value to be assigned to the property. Value can be a quoted DataWindow expression only where noted.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property for HTML</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>AppendedHTML</td>
<td>HTML you want to append to the generated syntax for the rendering of a DataWindow control before the closing bracket of the HTML element for that control.</td>
</tr>
<tr>
<td>Link</td>
<td>((exp)) A URL that is the target of a link (HTML anchor element) generated for each data item in the column or for the specified control. The text or user-visible part of the link will be the data value in the column, the value of the computed field, the text in the Text control, or the image of a Picture control. The URL can include parameters. Other properties, such as LinkArgs, can cause additional parameters to be added when the HTML is generated.</td>
</tr>
</tbody>
</table>
| LinkArgs          | A string in the form: \(\text{argname} = \text{exp} | \text{argname} = \text{exp} \) ...  
\text{Argname} is a page parameter to be passed to the server.  
\text{Exp} is a DataWindow expression whose value is a string. It is evaluated and converted using URL encoding and included in the \text{linkargs} string.  
The evaluated LinkArgs string is appended to the HTML.Link property when HTML is generated to produce a hyperlink for each data item in a column or other DataWindow control. |
**Property for HTML** | **Value**
---|---
**LinkTarget** | *(exp)* The name of a target frame or window for the hyperlink (HTML A element) specified in the Link property. The target is included using the TARGET attribute. You can use the LinkTarget property to direct the new page to a detail window or frame in a master/detail page design. If LinkTarget is null or an empty string (""), then no TARGET attribute is generated.

**ValueIsHTML** *(does not apply to Picture controls)* | *(exp)* A boolean that, if true, allows the control contents (data value in a read-only column, the value of a computed field that is not calculated on the client, or the text in a Text control) to be generated as HTML. For XHTML, the control contents must be well-formed XHTML.

**Usage**
The Link properties are typically used to create master/detail Web pages where a link on a data item jumps to a detail DataWindow for that item. LinkArgs is used to pass a retrieval argument identifying the particular item.

The AppendedHTML property is used to specify attributes and event actions to add to the HTML rendered for Web DataWindow controls.

The HTML generator does not validate the HTML you append to or include in controls in DataWindow objects. If the HTML is invalid, the DataWindow might not display correctly. You must also be careful not to append an event name that is already generated for the control as a coded client-side event.

**In the painter** Select the control and set the value in the Properties window, HTML category.

**Examples**

```Visual Basic```
// EMPID and PAGE are page parameters for the // page server’s session object
dw1.SetProperty("empid.HTML.Link", "empform.html")
dw1.SetProperty("empid.HTML.LinkArgs", "EMPID = 'empid'")
dw1.SetProperty("empid.HTML.LinkTarget", "detail_win")
dw1.SetProperty("empid.HTML.ValueIsHTML", "true")
dw1.SetProperty("helpicon.HTML.Link", "help.html")
dw1.SetProperty("helpicon.LinkArgs", "PAGE = 'empform'")
```Visual Basic```

setting = dw1.Describe("DataWindow.HTML.Link")
dw1.Modify("empid.HTML.Link = 'empform.html'")
HTMLDW

Description
Specifies whether HTML generated for the DataWindow object provides updates and interactivity.

Applies to
DataWindow objects

Syntax
Describe and Modify argument:

"DataWindow.HTMLDW { = 'value' }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| value     | The value to be assigned to the property. Value can be a quoted DataWindow expression. Values are:
|           | • Yes – DataWindow HTML generation uses the HTMLGen properties.
|           | • No – DataWindow HTML generation is a read-only table as described for the Data.HTMLTable property.

Usage
When HTMLDW is set to Yes, the generated HTML supports data entry and takes advantage of browser features that enable user interaction when used with a page server (as described for the Data.HTML property). The generated HTML can be used to produce a page that displays a subset of retrieved rows and can include JavaScript code requesting additional pages with other subsets of the retrieved rows.

DataWindow features that will not be rendered into HTML include:

• Graph presentation styles and controls.
• Client-side expressions that include aggregate functions. Aggregate functions cannot be evaluated in the browser. Instead, they will be evaluated on the server and the resulting value included in the HTML.
• Resizable and movable controls.
• Sliding of controls to fill empty space.
• Autosizing of height or width.
• Edit Masks for column data entry.

In the painter Select the DataWindow object by deselecting all controls; then set the HTMLDW property in the General category in the Properties window.
**HTMLGen.property**

**Description**
Settings that control the level of features incorporated into HTML generated for the DataWindow.

**Applies to**
DataWindow objects

**Syntax**
Describe and Modify argument:

```
"DataWindow.HTMLGen.property { = 'value' }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>property</td>
<td>A property that controls how HTML is generated for a DataWindow. Properties and their values are listed in the table below.</td>
</tr>
<tr>
<td>value</td>
<td>The value to be assigned to the property. Value can be a quoted DataWindow expression where noted.</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Property for HTMLGen</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Browser</td>
<td>((exp)) A string identifying the browser in which you want to display the generated HTML. The value should match the browser identifier part of the text string that the browser specifies in the HTTP header it sends to the server. This property is usually set dynamically on the server according to the HTTP header returned from the client. Recognized strings are listed in “Browser recognition” on page 270.</td>
</tr>
</tbody>
</table>
| ClientComputedFields | \((exp)\) Whether computed fields that reference column data are translated into JavaScript and computed in the client browser. Values are:  
  - Yes – (Default) Computed fields are translated to JavaScript where possible.  
  - No – Computed fields are always calculated on the server. Regardless of this setting, if the computed field includes aggregation functions, the computed field is calculated on the server. For more information about this and the following properties, see “Client properties” on page 271 |
| ClientEvents        | \((exp)\) Whether JavaScript code to trigger events is included in the generated HTML. Values are:  
  - Yes – (Default) JavaScript for triggering events is generated.  
  - No – JavaScript for events is not generated. |
| ClientFormatting    | \((exp)\) Whether display formats are applied to data items that do not have focus. JavaScript for formatting the data is translated from display formats specified in the DataWindow painter. If you want to use regional settings, such as a period as a date separator and a comma as a decimal separator, you must set ClientFormatting to Yes. Values are:  
  - Yes – (Default) Display formats are applied to data.  
  - No – Display formats are not used. |
<table>
<thead>
<tr>
<th>Property for HTMLGen</th>
<th>Value</th>
</tr>
</thead>
</table>
| ClientScriptable    | \((exp)\) Whether client-side JavaScript can interact with the control. Values are:  
  - Yes – Client-side JavaScript can call methods of the control.  
  - No – (Default) Client-side JavaScript cannot call methods.  
  This option adds approximately 20K to the size of the generated HTML. |
| ClientValidation     | \((exp)\) Whether JavaScript code to perform validation of user-entered data is included in the generated HTML. The validation code is translated from validation expressions specified in the DataWindow painter. Values are:  
  - Yes – (Default) Validation expressions are generated.  
  - No – Validation expressions are not generated. |
| CommonJSFile         | \((exp)\) Cache file name for common JavaScript functions required by Web DataWindows at runtime. If you set this property, the file is downloaded to the browser client once per session for use by all Web DataWindows. You can prefix the file name to a URL, or you can use the URL that you set with the HTMLGen.ResourceBase property. See “JavaScript caching” on page 271. |
| DateJSFile           | \((exp)\) Cache file name for common Web DataWindow functions that use a date format. If you set this property, the file is downloaded to the browser client once per session for use by all Web DataWindows. You can prefix the file name with a URL, or you can use the URL that you set with the HTMLGen.ResourceBase property. See “JavaScript caching” on page 271. |
| EncodeSelfLinkArgs   | \((exp)\) A switch to disable HTML 4 encoding of the evaluated HTMLGen.SelfLinkArgs expressions that are generated as hidden fields. The standard encoding limits character replacement to: `, `&`, `<`, and `>`. Disabling the standard encoding allows you to encode additional characters, but you must encode the argument expressions yourself. Values are:  
  - Yes – (Default) Encoding performed by the DataWindow server.  
  - No – Encoding not performed. |
| GenerateDDDWFrames   | \((exp)\) Specifies whether drop-down DataWindows are generated using inline frames (iFrames). The use of iFrames enhances the display so that the drop-down DataWindow displays in a Web application as it would in a Windows application. Using iFrames increases the volume of markup generated. Values are:  
  - Yes – (Default) Drop-down DataWindows are generated in formatted div elements over an iFrame.  
  - No – Drop-down DataWindows are generated in HTML select elements. The use of the GenerateDDDWFrames option for drop-down DataWindows is supported only in the Internet Explorer browser. In other browsers, the HTML select element is always used. |
### Property for HTMLGen

<table>
<thead>
<tr>
<th>Property for HTMLGen</th>
<th>Value</th>
</tr>
</thead>
</table>
| GenerateJavaScript  | *(exp)* Specifies whether to generate JavaScript if the browser is not recognized. Keep in mind that without JavaScript, updating of data is not available. Navigation links are still supported. Values are:  
  - **Yes** – (Default) JavaScript is generated even if the browser is not recognized. The resulting JavaScript is portable and does not use browser-specific features.  
  - **No** – JavaScript is not generated unless the browser is recognized. |
| HTMLVersion          | *(exp)* The version of HTML to generate. Values are:  
  - **3.2** – (Default) The HTML will include style sheets, but no absolute positioning or regular expressions.  
  - **4.0** – The HTML will include style sheets, absolute positioning, and regular expressions. If the browser is recognized, this property is ignored and browser-specific HTML is generated. |
| NetscapeLayers       | *(exp)* Formats the Web DataWindow for Netscape 4.0 or later using absolute positioning (in a manner similar to the formatting for Internet Explorer). See “NetscapeLayers property” on page 272. |
| NumberJSFile         | *(exp)* Cache file name for common Web DataWindow functions that use a number format. If you set this property, the file is downloaded to the browser client once per session for use by all Web DataWindows. You can prefix the file name with a URL, or you can use the URL that you set with the HTMLGen.ResourceBase property. See “JavaScript caching” on page 271. |
| ObjectName           | *(exp)* A string specifying a name used in generated code for the Web DataWindow client control, page parameters, and client-side events. You must specify a unique object name when there will be more than one Web DataWindow on a Web page so that names will not conflict. |
| PageSize             | *(exp)* The number of rows of data to include in a generated Web page. If the Web page does not include all available rows, you can include button controls to navigate to the rest of the data. To include all available rows in the page, specify 0 for PageSize. If the HTMLDW property is set to Yes, PageSize is used. If it is set to No, PageSize is ignored and all rows in the result set are generated in a single page. |
### Property for HTMLGen | Value
--- | ---
**PagingMethod** | A value of the WebPagingMethod enumerated variable that determines how paging is handled.  
Values are:  
- PostBack! (0) – (default) The control posts back to the server to perform paging operations.  
- Callback! (1) – The control calls a service on the client to perform paging operations.  
- XMLClientSide! (2) – The control retrieves the entire XML result set and performs paging operations on the client. This option is only available when the XML rendering format is used, and it cannot be used with the page navigation bar.  
See “PagingMethod” on page 271.

**ResourceBase** | *(exp)* The URL for included JavaScript files. If you set this property, you do not need to include a URL in the values for these other HTMLGen properties: CommonJSFile, DateJSFile, NumberJSFile, and StringJSFile.

**SelfLink** | *(exp)* A string specifying the URL for the current page. It cannot include parameters. Parameters specified in SelfLinkArgs can be added when HTML is generated.  
SelfLink is used to generate URLs for navigation buttons that obtain additional rows from the result set and for other buttons that reload the page, such as Update and Retrieve.

**SelfLinkArgs** | A string in the form:  
\[ argname = 'exp'| argname = 'exp' \] ...  
*Argname* is a page parameter to be passed to the server.  
*Exp* is a DataWindow expression whose value is a string. The DataWindow in the server component evaluates it, converts it using URL encoding, and includes it in the SelfLinkArgs string.  
The evaluated SelfLinkArgs expressions are included in the generated HTML as hidden fields. The arguments supply information that the server needs to render additional pages of the result set, such as retrieval arguments.

**StringJSFile** | *(exp)* Cache file name for common Web DataWindow functions that use a string format. If you set this property, the file is downloaded to the browser client once per session for use by all Web DataWindows. You can prefix the file name with a URL, or you can use the URL that you set with the HTMLGen.ResourceBase property. See “JavaScript caching” on page 271.

**TabIndexBase** | *(exp)* Sets the starting tab order number for a Web DataWindow. This property is useful for a Web page with multiple Web DataWindows when you can tab between columns of the DataWindows. Setting this property has no effect on page functionality when the page is viewed in a browser that does not support the tab index attribute. The maximum tab index allowed for a page is 32767. See “TabIndexBase property” on page 272.
Most of these properties are considered only when the HTMLDW property is set to Yes.

**Browser recognition**  The Browser and HTMLVersion properties are always considered when HTML is generated, regardless of the HTMLDW setting.

Browser identification strings are sent by the client to the server in the HTTP header. The server component can assign the HTTP_USER_AGENT value from the HTTP header to the Browser property. If the string specifies a browser that the DataWindow engine supports, the DataWindow will generate HTML optimized for that browser. Browser-specific HTML is generated only for Microsoft Internet Explorer and Netscape browsers.

If the browser is not recognized or not specified, then the generated HTML will use the HTMLVersion and GenerateJavaScript properties to decide what features to include. DataWindow HTML generation recognizes these browsers:

<table>
<thead>
<tr>
<th>Browser</th>
<th>HTTP header string</th>
<th>HTML features used</th>
</tr>
</thead>
<tbody>
<tr>
<td>Netscape</td>
<td>Mozilla/1.x (</td>
<td>No style sheets, no absolute positioning, no JavaScript.</td>
</tr>
<tr>
<td></td>
<td>Mozilla/2.x (</td>
<td>JavaScript.</td>
</tr>
<tr>
<td></td>
<td>Mozilla/3.x (</td>
<td>No style sheets, no absolute positioning, no regular expressions.</td>
</tr>
<tr>
<td></td>
<td>Mozilla/4.x (</td>
<td>Style sheets, JavaScript, regular expressions. No absolute positioning.</td>
</tr>
<tr>
<td>Microsoft Internet Explorer</td>
<td>Mozilla/1.22 (compatible; MSIE 2.x;</td>
<td>No style sheets, no absolute positioning, no tab order, no JavaScript.</td>
</tr>
<tr>
<td></td>
<td>Mozilla/2.0 (compatible; MSIE 3.x;</td>
<td>Style sheets, tab order, JavaScript. No absolute positioning, no regular expressions.</td>
</tr>
<tr>
<td></td>
<td>Mozilla/4.0 (compatible; MSIE 4.x;</td>
<td>Style sheets, absolute positioning, tab order, regular expressions.</td>
</tr>
<tr>
<td></td>
<td>Mozilla/4.0 (compatible; MSIE 5.x;</td>
<td>Style sheets, absolute positioning, tab order, regular expressions.</td>
</tr>
<tr>
<td></td>
<td>Mozilla/4.0 (compatible; MSIE 6.x;</td>
<td>Style sheets, absolute positioning, tab order, regular expressions.</td>
</tr>
<tr>
<td>Opera</td>
<td>Mozilla/3.0 (compatible; Opera 3.x;</td>
<td>JavaScript, regular expressions. No style sheets, no absolute positioning.</td>
</tr>
</tbody>
</table>
Client properties  The ClientEvents, ClientFormatting, ClientValidation, ClientComputedFields, and ClientScriptable properties control the amount of JavaScript that is generated for the Web DataWindow, which impacts the size of the page that is downloaded to the browser. You can reduce the size of the generated HTML by setting one or more of the properties to No.

In DataWindow .NET, you can set these properties in the Properties window in your .Visual Studio. If you change the default setting in the Properties window, the value you set in the DataWindow painter is overridden.

JavaScript caching  You can also reduce the size of the generated HTML by setting up cache files for common Web DataWindow client-side methods.

With JavaScript caching, you improve performance after the first Web DataWindow page is generated—as long as the browser on the client computer is configured to use cached files. With caching enabled, the browser loads the JS files from the Web server into its cache, and these become available for all the Web DataWindow pages in your application. There is no performance gain if the browser does not find the JS files in its cache since, in this case, it reloads the files from the Web server.

In DataWindow .NET, you can specify the names and locations of cached JavaScript files using the JavaScriptConfigurations property. This property allows you to cache JavaScript for XML and XHTML as well as HTML Web DataWindows.

PagingMethod  The PagingMethod property determines whether the control uses the client-side script callback mechanism introduced in the .NET Framework 2.0 to execute server-side code without posting and refreshing the current page.

The default is to post back to the server (PostBack!).

The Callback! option uses script callbacks to retrieve the next page of XML data. It corresponds to the Microsoft GridView control’s EnableSortingAndPagingCallback property, but applies only to paging. Client-side sorting is handled by another mechanism.

For the XML rendering format, the design of the Callback! option requires that a reusable XSLT stylesheet be generated so that the browser can cache it. The benefit from this requirement is that only the XML data for the next requested page need be returned by the callback. This XML data is always trivial in size (about a 1 to 20 ratio), resulting in significant bandwidth savings. This is unlike other implementations, where the entire presentation is always regenerated and downloaded again from every callback.
The generated XSLT stylesheet is not reusable, and therefore cannot be cached by the browser, if the DataWindow layout is inconsistent page-to-page, or it does not contain a complete first page of data. In these scenarios, the Callback! option defers to PostBack! until a stylesheet can be generated that is reusable, and can therefore be cached in the browser.

The XMLClientSide! option is only available with the XML rendering format. It retrieves the entire XML result set and uses XSLT re-transformation of the cached stylesheet to perform paging on the client. This option can currently be used only if the presentation style is uniform from page to page. For example, it cannot handle a summary band on the last page. This option cannot be used with a page navigation bar.

When PagingMethod is set to XMLClientSide!, InsertRow, AppendRow, and DeleteRow actions do not require a postback or callback to the server. However, computed fields in the DataWindow that are dependent on the RowCount method are not refreshed until an action such as Update or Retrieve forces a postback to the server.

NetscapeLayers property Even if you set the NetscapeLayers property to true, certain functionality in a Netscape browser using absolute positioning might not be identical to the functionality available with Internet Explorer. For example, you cannot tab between DataWindow columns using a Netscape browser on an NT machine (although you can do this using a Netscape browser on a Solaris machine).

TabIndexBase property If you add Web DataWindows to a page that already has a Web DataWindow on it, you can set the TabIndexBase property for each Web DataWindow you add.

For a page with two Web DataWindows, setting the tab index base for the second DataWindow to a number greater than the tab index for the last column of the first DataWindow allows the user (using an Internet Explorer browser) to tab through all the columns of the first DataWindow before tabbing to the second DataWindow. Otherwise, pressing the Tab key could cause the cursor and focus to jump from one DataWindow to another instead of tabbing to the next column in the DataWindow that initially had focus.

In the painter Select the DataWindow object by deselecting all controls; then set the values in the Properties window, Web Generation category or JavaScript Generation category. Select HTML/XHTML from the WebDW or JSGen list to display the properties.
Examples

[Visual Basic]
setting = dw1.Describe
    ("DataWindow.HTMLGen.Browser")

dw1.Modify("DataWindow.HTMLGen.ClientValidation = 'no'")

dw1.Modify("DataWindow.HTMLGen.PublishPath= 'C:\Inetpub\wwwroot\MyWebApp\generatedfiles'")
dw1.Modify("DataWindow.HTMLGen.ResourceBase= '/MyWebApp/generatedfiles'")

This statement sets the XMLGen.Paging property so that the complete result set is downloaded to the client and paging takes place on the client:

[Visual Basic]
dw1.Modify("DataWindow.HTMLGen.PagingMethod=XMLClientSide!")

**HTMLTable.property**

Description
Settings for the display of DataWindow data when displayed in HTML table format. These settings simplify the transfer of data from a database to an HTML page. They are particularly useful when used to create HTML pages dynamically.

Applies to
DataWindow objects

Syntax
Describe and Modify argument:

"DataWindow.HTMLTable.property\(=\ 'value'\)"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>property</td>
<td>A property for a DataWindow to be displayed in HTML table format. Properties and their values are listed in the table below.</td>
</tr>
<tr>
<td>value</td>
<td>The value to be assigned to the property. Value can be a quoted DataWindow expression.</td>
</tr>
</tbody>
</table>

**Property for HTMLTable**

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Border</strong></td>
</tr>
<tr>
<td>((exp)) Border attribute for the HTMLTable element. The default is 1 (line around the table).</td>
</tr>
<tr>
<td><strong>CellPadding</strong></td>
</tr>
<tr>
<td>((exp)) CellPadding attribute for the HTMLTable element. The default is 0.</td>
</tr>
<tr>
<td><strong>CellSpacing</strong></td>
</tr>
<tr>
<td>((exp)) CellSpacing attribute for the HTMLTable element. The default is 0.</td>
</tr>
</tbody>
</table>
### Property for HTMLTable

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>GenerateCSS</td>
<td><em>(exp)</em> Controls whether the DataWindow HTMLTable property’s Table element contains border, cellpadding, cellspacing, nowrap, and width attributes. Also controls whether elements within the table contain CLASS references that control style sheet use. The default is no.</td>
</tr>
<tr>
<td>NoWrap</td>
<td><em>(exp)</em> NoWrap attribute for the HTMLTable element. The default is to include this attribute.</td>
</tr>
<tr>
<td>StyleSheet</td>
<td><em>(exp)</em> HTML cascading style sheet generated for the DataWindow.</td>
</tr>
<tr>
<td>Width</td>
<td>Width attribute for the HTMLTable element. The default is 0.</td>
</tr>
</tbody>
</table>

#### Usage

**In the painter** Set the value using the Properties window, HTML Table category.

#### Examples

**[Visual Basic]**

```visualbasic
setting = dw1.Describe
    ("DataWindow.HTMLTable.StyleSheet")
    dw1.Modify("DataWindow.HTMLTable.NoWrap = 'yes'")
```

#### ID

**Description** The number of the column or TableBlob.

**Applies to** Column and TableBlob controls

**Syntax** Describe and Modify argument:

```plaintext
"controlname.ID"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlname</td>
<td>The name of the column or TableBlob for which you want the ID number</td>
</tr>
</tbody>
</table>

#### Examples

**[Visual Basic]**

```visualbasic
setting = dw1.Describe("empname.ID")
```

#### Identity

**Description** Whether the database is to supply the value of the column in a newly inserted row. If so, the column is not updatable; the column is excluded from the INSERT statement.

Not all DBMSs support the identity property. For more information see the documentation for your DBMS.
Applies to: Column controls

Syntax: Describe and Modify argument:

```
"columnname.Identity { = 'value' }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columnname</td>
<td>A string containing the name of the column for which you want to get or set the identity property.</td>
</tr>
<tr>
<td>value</td>
<td>A string indicating whether a column’s value in a newly inserted row is supplied by the DBMS: Yes – The DBMS will supply the value of the column in a newly inserted row; the column is not updatable. No – The column is updatable.</td>
</tr>
</tbody>
</table>

Examples:

[Visual Basic]

```vbnet
dw1.Modify("empid.Identity='yes'")
```

**Import.XML.Trace**

Description: Setting that determines whether import trace information is written to a log file.

Applies to: DataWindow objects

Syntax: Describe and Modify argument:

```
"DataWindow.Import.XML.Trace { = 'value' }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Whether trace information is written to a log file. Values are: Yes – Trace information is written to a log file. No – Trace information is not written to a log file (default).</td>
</tr>
</tbody>
</table>

Usage: If you want to collect trace information, this property should be set before you call the `ImportClipboard`, `ImportFile`, or `ImportString` method to import data from an XML document. The trace information is appended to the file you specify using the `Import.XML.TraceFile` property. If no trace file is specified, trace information is appended to a file named `pbxmltrc.log` in the current directory.

In the painter: In the Data Import category in the Properties window for the DataWindow object, select XML from the Import list, set XML Trace to True and supply a trace file name.
**Import.XML.TraceFile**

**Examples**

This example specifies that trace information should be written to a file called `xmltrace.log` in the `C:\temp` directory.

```
[Visual Basic]
dw1.Modify("DataWindow.Import.XML.Trace = 'yes' ")
dw1.Modify("DataWindow.Import.XML.TraceFile = 'C:\temp\xmltrace.log' ")
```

See also: Import.XML.TraceFile

**Import.XML.TraceFile**

**Description**

Specifies the name and location of an import trace file.

**Applies to**

DataWindow objects

**Syntax**

Describe and Modify argument:

```
"DataWindow.Import.XML.TraceFile { = 'value' }
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>A string whose value is the name of the trace output file. If the file does not exist, it is created.</td>
</tr>
</tbody>
</table>

**Usage**

If you want to collect trace information, the Import.XML.Trace property should be set before you call the `ImportClipboard`, `ImportFile`, or `ImportString` method to import data from an XML document. The trace information is appended to the file you specify using the Import.XML.TraceFile property. If no trace file is specified, trace information is appended to a file named `pbxmltrc.log` in the current directory.

**In the painter**

In the Data Import category in the Properties window for the DataWindow object, select XML from the Import list, set XML Trace to True and supply a trace file name.

See also: Import.XML.Trace

**Import.XML.UseTemplate**

**Description**

Setting that optionally controls the logical structure of the XML imported from an XML file into a DataWindow object using the `ImportFile` method.

**Applies to**

DataWindow objects

**Syntax**

Describe and Modify argument:

```
"DataWindow.Import.XML.UseTemplate { = 'value' }
```

---

DataWindow .NET
This property should be set to specify the logical structure of the XML imported before you call the `ImportFile` method to import data from an XML document. An import template is not required if the XML document from which data is imported corresponds to the DataWindow column definition.

If an export template for a DataWindow object exists, it can be used as an import template. Only the mapping of column names to element attribute names is used for import. The order of elements within the template is not significant, because import values are located by name match and nesting depth within the XML document. All other information in the template, such as controls and comments, is ignored.

**In the painter** In the Data Import category in the Properties window for the DataWindow object, select XML from the Import list and select a template from the UseTemplate list.

**Examples**

This example sets the name of the current XML import template used in `dw1` to `t_import_report`. If `t_import_report` does not exist, the current template is not changed.

```
[Visual Basic]
dw1.Modify("DataWindow.Import.XML.UseTemplate = 't_import_report'")
```

**See also**

Export.XML.UseTemplate

### Initial

**Description**

The initial value of the column in a newly inserted row.

**Applies to**

Column controls

**Syntax**

Describe and Modify argument:

```
"columnname.Initial { = 'initialvalue' }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>columnname</code></td>
<td>A string containing the name of the column for which you want to get or set the initial property.</td>
</tr>
</tbody>
</table>
### Ink.property

**Description**
Properties that control the attributes of ink in an InkPicture control or a column with the InkEdit edit style.

**Applies to**
Column and InkPicture controls

**Syntax**
Describe and Modify argument:

```
"inkpicname.Ink.property { = value }"
"columnname.Ink.property { = value }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inkpicname</td>
<td>The name of an InkPicture control.</td>
</tr>
<tr>
<td>columnname</td>
<td>The name of a column that has the InkEdit edit style.</td>
</tr>
<tr>
<td>property</td>
<td>A property for the InkPicture control or InkEdit column. Properties and their settings are listed in the table below.</td>
</tr>
<tr>
<td>value</td>
<td>The value to be assigned to the property.</td>
</tr>
</tbody>
</table>

### Property for Ink

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AntiAliased</td>
<td>A drawing attribute that specifies whether the foreground and background colors along the edge of the drawn ink are blended (antialiased) to make the stroke smoother and sharper. Values are: true – The ink stroke appears smoother and sharper (default) false – The ink stroke is not antialiased</td>
</tr>
<tr>
<td>Color</td>
<td>A drawing attribute that specifies the current ink color. The default color is black.</td>
</tr>
<tr>
<td>Height</td>
<td>A drawing attribute that specifies the height of the side of the rectangular pen tip in HIMETRIC units (1 HIMETRIC unit = .01mm). The default is 1.</td>
</tr>
</tbody>
</table>

Examples

```Visual Basic
setting = dw1.Describe("empname.Initial")
dw1.Modify("empname.Initial='empty'")
dw1.Modify("empstatus.Initial='A'")
```
CHAPTER 3  DataWindow Object Properties

DataWindow Object Properties

Usage
In the painter  Select the control and set values in the Properties window, Ink or InkPicture category.

Examples
[Visual Basic]
li_color = dw1.Describe("emp_status.Ink.Color")

See also
InkEdit.property
InkPic.property

Property for Ink

<table>
<thead>
<tr>
<th>Property for Ink</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>IgnorePressure</td>
<td>A drawing attribute that specifies whether the drawn ink gets wider as the pressure of the pen tip on the tablet surface increases. Values are: true – Pressure from the pen tip is ignored false – The width of the ink increases with the pressure of the pen tip (default)</td>
</tr>
<tr>
<td>Pentip</td>
<td>A drawing attribute that specifies whether the pen tip is round or rectangular. Values are: Ball (0) – The pen tip is round (default) Rectangle (1) – The pen tip is rectangular</td>
</tr>
<tr>
<td>Transparency</td>
<td>A drawing attribute that specifies the transparency of drawn ink. The range of values is from 0 for totally opaque (the default) to 255 for totally transparent.</td>
</tr>
<tr>
<td>Width</td>
<td>A drawing attribute that specifies the width of the side of the rectangular pen tip in HIMETRIC units (1 HIMETRIC unit = .01mm). The default is 53.</td>
</tr>
</tbody>
</table>

Usage
In the painter  Select the control and set values in the Properties window, Ink or InkPicture category.

Examples
[Visual Basic]
li_color = dw1.Describe("emp_status.Ink.Color")

See also
InkEdit.property
InkPic.property

InkEdit.property

Description
Properties that control the behavior of a column with the InkEdit edit style.

Applies to
Column controls

Syntax
DataWindow .NET dot notation:

(((InkEdit) colname/EditStyle).property [C#])
CType(colname/EditStyle, InkEdit).property [Visual Basic]

Describe and Modify argument:

"colunnname.InkEdit.property ( = value )"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>colunnname</td>
<td>The name of a column that has the InkEdit edit style.</td>
</tr>
<tr>
<td>property</td>
<td>A property for the InkEdit column. Properties and their settings are listed in the table below.</td>
</tr>
<tr>
<td>value</td>
<td>The value to be assigned to the property.</td>
</tr>
</tbody>
</table>
### InkEdit.property

<table>
<thead>
<tr>
<th>Property for InkEdit</th>
<th>Value</th>
</tr>
</thead>
</table>
| **AutoSelect**       | Whether to select the contents of the edit control automatically when it receives focus. Values are:  
Yes – Select automatically (default).  
No – Do not select automatically.  
You can use AutoSelect with DataWindowSyntaxFromSql. The setting applies to all the columns in the generated syntax. |
| **DisplayOnly**      | Specifies whether the text is display-only and cannot be changed by the user. Values are:  
true – Text cannot be changed by user.  
false – Text can be changed by user (default). |
| **Factoid**          | Specifies a context for ink recognition. Set this property if the input data is of a known type, such as a date or Web address, to constrain the search for a recognition result. Possible values include digit, e-mail, Web, date, time, number, currency, percent, and telephone. For a list of values, see the table that follows. |
| **FocusRectangle**   | Whether a dotted rectangle (the focus rectangle) will surround the current row of the column when the column has focus. Values are:  
Yes – (Default) Display the focus rectangle.  
No – Do not display the focus rectangle (default).  
You can use FocusRectangle with DataWindowSyntaxFromSql. The setting applies to all the columns in the generated syntax. |
| **HScrollbar**       | Whether a horizontal scroll bar displays in the edit control. Values are:  
Yes – Display the horizontal scroll bar.  
No – Do not display the horizontal scroll bar (default). |
| **InkMode**          | Specifies whether ink collection is enabled and whether ink only or ink and gestures are collected. Values are:  
InkDisabled (0) – Ink collection is disabled.  
CollectInkOnly (1) – Only ink is collected.  
CollectInkAndGestures (2) – Ink and gestures are collected (default). |
| **Limit**            | A number specifying the maximum number of characters (0 to 32,767) that the user can enter. 0 means unlimited. |
| **NilIsNull**        | Whether to set the data value of the InkEdit to null when the user leaves the edit box blank. Values are:  
Yes – Make the Empty string null.  
No – Do not make the empty string null (default). |
| **RecognitionTimer** | Specifies the time period in milliseconds between the last ink stroke and the start of text recognition. The default is 2000 (two seconds). |
| **Required**         | Whether the column is required. Values are:  
Yes – Required.  
No – (Default) Not required. |
### Usage

The following values for Factoid are available. After the Default and None factoids, the drop-down list in the Properties view displays factoids for special formats in alphabetical order, followed by single-character factoids and Asian-language factoids. You can set multiple factoids by separating them with the pipe (|) character.

<table>
<thead>
<tr>
<th>Factoid</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Default</td>
<td>Returns recognizer to the default setting. For Western languages, the default setting includes the user and system dictionaries, various punctuation marks, and the Web and Number factoids. For Eastern languages, the default setting includes all characters supported by the recognizer.</td>
</tr>
<tr>
<td>None</td>
<td>Disables all factoids, dictionaries, and the language model.</td>
</tr>
<tr>
<td>Currency</td>
<td>Currency in pounds, dollars, euros, and yen.</td>
</tr>
<tr>
<td>Date</td>
<td>Dates written in English; for example 8/19/2005, Aug 19, 2005, or Friday, August 19, 2005.</td>
</tr>
<tr>
<td>E-mail</td>
<td>E-mail addresses.</td>
</tr>
<tr>
<td>Filename</td>
<td>Windows file name paths. The name cannot include the following characters: / : &quot; &lt; &gt;</td>
</tr>
<tr>
<td>Number</td>
<td>Numeric values, including ordinals, decimals, separators, common suffixes, and mathematical symbols. This factoid includes the Currency and Time factoids.</td>
</tr>
<tr>
<td>Percent</td>
<td>A number followed by the percent symbol.</td>
</tr>
<tr>
<td>Postal Code</td>
<td>Postal codes as written in English, for example 01730 or CT17 9PW.</td>
</tr>
<tr>
<td>System Dictionary</td>
<td>Words in the system dictionary only.</td>
</tr>
<tr>
<td>Telephone</td>
<td>Telephone numbers as written in English, for example (555) 555 5555 or +44 1234 123456.</td>
</tr>
<tr>
<td>Time</td>
<td>Times as written in English, for example 15:05 or 3:05 pm.</td>
</tr>
<tr>
<td>Web</td>
<td>Various URL formats.</td>
</tr>
<tr>
<td>Word List</td>
<td>Words on the word list associated with the recognizer context only.</td>
</tr>
<tr>
<td>Digit</td>
<td>A single digit (0-9).</td>
</tr>
</tbody>
</table>
In addition, the following Asian-language factoids are available:

<table>
<thead>
<tr>
<th>Factoid</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bopomofo</td>
<td>Kanji Common</td>
</tr>
<tr>
<td>Hangul Common</td>
<td>Katakana</td>
</tr>
<tr>
<td>Hiragana</td>
<td>Korean Common</td>
</tr>
<tr>
<td>Jamo</td>
<td>Simplified Chinese Common</td>
</tr>
<tr>
<td>Japanese Common</td>
<td>Traditional Chinese Common</td>
</tr>
</tbody>
</table>

**In the painter**  
Select the control and set values in the Properties window, Behavior category. The EditStyle on the Edit category must be set to InkEdit.

In DataWindow .NET, you can use the InkEdit class to set InkEdit Edit style properties using dot notation. See the description of the InkEdit class in the online Help in Visual Studio .NET for a complete list of properties.

**Examples**

```visualbasic
str = dw1.Describe("emp_bd.InkEdit.Factoid")
dw1.Modify("emp_bd.InkEdit.Factoid=EMAIL")
```

```csharp
if (empBnfts.EditStyle is InkEdit)
{
    ((InkEdit)empBnfts.InkEdit).LeftText = true;
} else
{
...
```

**See also**  
Ink.property

---

**InkPic.property**

**Description**  
Properties that control the behavior of ink in an InkPicture control.

**Applies to**  
InkPicture controls
Syntax Describe and Modify argument:

"inkpicname.InkPic.property { = value }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>inkpicname</td>
<td>The name of an InkPicture control.</td>
</tr>
<tr>
<td>property</td>
<td>A property for the InkPicture control. Properties and their settings are listed in the table below.</td>
</tr>
<tr>
<td>value</td>
<td>The value to be assigned to the property.</td>
</tr>
</tbody>
</table>

### Property for InkPic | Value
--- | ---
AutoErase Specifies whether the auto erase feature available on some styluses is turned on. Values are:
- true – AutoErase is turned on.
- false – AutoErase is turned off (default).

BackColor Specifies the numeric value of the background color: -2 to 16,777,215. For more information about color, see the RGB function.

CollectionMode Specifies whether ink only, gestures only, or ink and gestures are collected. Values are:
- InkOnly (0) – Only ink is collected (default).
- GestureOnly (1) – Only gestures are collected.
- InkAndGesture (2) – Ink and gestures are collected.

DynamicRendering Specifies whether the ink is rendered (displayed in the control) as it is drawn. The default is true.

EditMode Specifies whether the editing mode of the control is set for drawing, deleting, or selecting ink. Values are:
- InkMode (0) – Ink is drawn (default).
- DeleteMode (1) – Ink is deleted.
- SelectMode (2) – Ink is selected.

EraserMode Specifies whether ink is removed by stroke or point. Values are:
- StrokeErase (0) – The entire ink stroke under the stylus is removed (default).
- PointErase (1) – Only the ink under the stylus is removed.

EraserWidth Specifies the width of the eraser pen tip in HIMETRIC units (1 HIMETRIC unit = .01mm). The default is 212. This property applies when EditMode is set to DeleteMode and EraserMode is set to PointErase.

HighContrastInk Specifies whether ink is rendered in a single color when the system is in high contrast mode and draws the selection rectangle and handles in high contrast. Values are:
- true – Ink is rendered in a single color in high contrast mode (default).
- false – Ink is not rendered in a single color in high contrast mode.

InkEnabled Specifies whether the InkPicture control collects pen input. Values are:
- true – The control collects pen input (default).
- false – The control does not collect pen input and no pen-related events fire.
Invert

<table>
<thead>
<tr>
<th>Property for InkPic</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MarginX</td>
<td>Specifies the x-axis margin around the control in PowerBuilder units. The default value is 0.</td>
</tr>
<tr>
<td>MarginY</td>
<td>Specifies the y-axis margin around the control in PowerBuilder units. The default value is 0.</td>
</tr>
<tr>
<td>PictureSizeMode</td>
<td>Specifies how the picture is displayed in the control. Values are:</td>
</tr>
<tr>
<td></td>
<td>Center Image (1) – The picture is centered in the control.</td>
</tr>
<tr>
<td></td>
<td>Normal (2) – The picture is displayed in the upper-left corner of the control and any part of the picture that does not fit in the control is clipped (default).</td>
</tr>
<tr>
<td></td>
<td>Stretch (3) – The picture is stretched to fill the control.</td>
</tr>
</tbody>
</table>

Usage

In the painter Select the control and set values in the Properties window, InkPicture category.

Examples

[Visual Basic]
li_color = dw1.Describe("inkpic1.InkPic.BackColor")

See also

Ink.property

Invert

Description

The way the colors in a Picture control are displayed, either inverted or normal.

Applies to

Picture controls

Syntax

Describe and Modify argument:

"bitmapname.Invert { = 'number' }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>bitmapname</td>
<td>The name of the Picture control in the DataWindow for which you want to invert the colors.</td>
</tr>
<tr>
<td>number</td>
<td>(exp) A boolean number indicating whether the colors of the picture will display inverted. Values are: 0 – (Default) No; do not invert the picture’s colors, 1 – Yes; display the picture with colors inverted.</td>
</tr>
</tbody>
</table>

Number can be a quoted DataWindow expression.

Usage

In the painter Select the control and set the value in the Properties window, Appearance category.

Examples

[Visual Basic]
setting = dw1.Describe("bitmap_1.Invert")
dw1.Modify( _
  "bitmap_1.Invert='0-tIf(empstatus='A--',0,1')"
)
### JSGen.**property**

**Description**
Settings that specify the physical path to which generated JavaScript is published and the URL indicating the location of the generated JavaScript.

**Applies to**
DataWindow objects

**Syntax**
Describe and Modify argument:

```
"DataWindow.JSGen.**property** \{ = ' value ' \}"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>property</strong></td>
<td>One of the following:</td>
</tr>
<tr>
<td>    PublishPath</td>
<td></td>
</tr>
<tr>
<td>    ResourceBase</td>
<td></td>
</tr>
<tr>
<td><strong>value</strong></td>
<td>(exp) PublishPath – A string that specifies the physical path of the Web site folder to which the DataWindow server publishes the generated JavaScript.</td>
</tr>
<tr>
<td>    (exp) ResourceBase – A string that specifies the URL of the generated JavaScript for performing client-side XSLT transformation and instantiation of client-side data.</td>
<td></td>
</tr>
</tbody>
</table>

**Usage**
The PublishPath folder must correspond to the URL specified in the ResourceBase property. At runtime, after the DataWindow server generates JavaScript to the PublishPath folder, it includes it in the final XHTML page by referencing it with the value of the ResourceBase property in a `<script>` element.

**In the painter** In the JavaScript Generation category in the Properties window for the DataWindow object, select XHTML from the JSGen list and specify the ResourceBase and Publish Path locations.

**In DataWindow .NET** In DataWindow .NET, you can specify the physical directory in which dynamically created files, such as `.css`, `.js`, `.xml`, and `.xslt` files, and URL references are stored, using the XmlConfigurations.UrlPath property. If you specify a value for XmlConfigurations.UrlPath, it overrides the values set for PublishPath and ResourceBase set in the DataWindow painter.

If you want to specify different paths for each of the different file types or if you want to specify a full path, set the properties in the DataWindow painter and leave the XmlConfigurations.UrlPath property empty in DataWindow .NET.

If you do not set these properties in the DataWindow painter or in DataWindow .NET, the files are saved in the current Web application’s path.
These statements set the JSGen.ResourceBase and JSGen.PublishPath properties:

```vbnet
dw1.Modify("DataWindow.JSGen.PublishPath= 'C:\Inetpub\wwwroot\MyWebApp\generatedfiles' ")
dw1.Modify("DataWindow.JSGen.ResourceBase= '/MyWebApp/generatedfiles' ")
```

**Key**

**Description**
Whether the column is part of the database table’s primary key.

**Applies to**
Column controls

**Syntax**
Describe and Modify argument:

```
"columnname.Key { = value }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columnname</td>
<td>The column for which you want to get or set primary key status.</td>
</tr>
<tr>
<td>value</td>
<td>Whether the column is part of the primary key. Values are:</td>
</tr>
<tr>
<td></td>
<td>Yes – The column is part of the primary key</td>
</tr>
<tr>
<td></td>
<td>No – The column is not part of the key</td>
</tr>
</tbody>
</table>

**Usage**
In the painter  Set the value using the Rows menu, Update Properties.

**Examples**

```vbnet
setting = dw1.Describe("empid.Key")
dw1.Modify("empid.Key=Yes")
```

**KeyClause**

**Description**
An expression to be used as the key clause when retrieving the blob.

**Applies to**
TableBlob controls

**Syntax**
Describe and Modify argument:

```
"tblobname.KeyClause { = 'keyclause' }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tblobname</td>
<td>The name of the TableBlob for which you want to specify a key clause.</td>
</tr>
</tbody>
</table>
Usage

**In the painter**  Select the control and set the value in the Properties window, Definition category.

**Examples**

With the following setting, the value of key_col will be put in col2 when the DataWindow server constructs the WHERE clause for the SELECTBLOB statement:

```
[Visual Basic]
dw1.Modify(blob_1.KeyClause='Key_col = :col2')
```

### Label.property

**Description**

Settings for a DataWindow whose presentation style is Label.

**Applies to**

DataWindows

**Syntax**

Describe and Modify argument:

```
"DataWindow.Label.property { = value }"
```

DataWindowSyntaxFromSql:

```
DataWindow(Label.property = value)
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>property</td>
<td>A property for the Label presentation style. Properties and their settings are listed in the table below.</td>
</tr>
<tr>
<td>value</td>
<td>The value to be assigned to the property. For Label properties, value cannot be a DataWindow expression.</td>
</tr>
</tbody>
</table>

#### Property for Label

<table>
<thead>
<tr>
<th>Value</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Columns</td>
<td>An integer indicating the number of columns of labels on a sheet. Painter: Label group, Labels Across option.</td>
</tr>
<tr>
<td>Columns.Spacing</td>
<td>An integer indicating the space between columns of labels in the units specified for the DataWindow object. Painter: Arrangement group, Between Columns option.</td>
</tr>
<tr>
<td>Ellipse_Height</td>
<td>An integer specifying the radius of the vertical part of the label in the unit of measure specified for the DataWindow object. Painter: Not set in painter.</td>
</tr>
</tbody>
</table>
### Property for Label

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ellipse_Width</td>
<td>An integer radius of the horizontal part of the label in the unit of measure specified for the DataWindow object.</td>
</tr>
<tr>
<td></td>
<td>Painter: Not set in painter.</td>
</tr>
<tr>
<td>Height</td>
<td>An integer specifying the height of a label in the units specified for the DataWindow object.</td>
</tr>
<tr>
<td></td>
<td>Painter: Label group, Height option.</td>
</tr>
<tr>
<td>Name</td>
<td>A string containing the name of a label.</td>
</tr>
<tr>
<td></td>
<td>Painter: Predefined Label option.</td>
</tr>
<tr>
<td>Rows</td>
<td>An integer indicating the number of rows of labels on a sheet.</td>
</tr>
<tr>
<td></td>
<td>Painter: Label group, Labels Down option.</td>
</tr>
<tr>
<td>Rows.Spacing</td>
<td>An integer indicating the space between rows of labels on a sheet in the units specified for the DataWindow object.</td>
</tr>
<tr>
<td></td>
<td>Painter: Arrangement group, Between Rows option.</td>
</tr>
<tr>
<td>Shape</td>
<td>A string specifying the shape of a label.</td>
</tr>
<tr>
<td></td>
<td>Values are: Rectangle, RoundRectangle, Oval</td>
</tr>
<tr>
<td></td>
<td>Painter: Not set in painter.</td>
</tr>
<tr>
<td>Sheet</td>
<td><em>(Describe only)</em> Whether the paper is sheet fed or continuous.</td>
</tr>
<tr>
<td></td>
<td>Values are: Yes – Sheet fed, No – Continuous</td>
</tr>
<tr>
<td></td>
<td>Painter: Arrangement group, Paper option.</td>
</tr>
<tr>
<td>TopDown</td>
<td><em>(Describe only)</em> Whether the labels will be printed from the top to the bottom or across the page.</td>
</tr>
<tr>
<td></td>
<td>Values are: No – Print labels across the page, Yes – Print labels from top to bottom.</td>
</tr>
<tr>
<td></td>
<td>Painter: Arrangement group, Arrange option.</td>
</tr>
<tr>
<td>Width</td>
<td>An integer specifying the width of a label in the units specified for the DataWindow object.</td>
</tr>
<tr>
<td></td>
<td>Painter: Label group, Width option.</td>
</tr>
</tbody>
</table>

### Usage

**In the painter**  Select the DataWindow object by deselecting all controls; then set the value in the Properties window, General category (when presentation style is Label).
Examples

[Visual Basic]
setting = dw1.Describe("DataWindow.Label.Sheet")
dw1.Modify("DataWindow.Label.Width=250")
dw1.Modify("DataWindow.Label.Height=150")
dw1.Modify("DataWindow.Label.Columns=2")
dw1.Modify("DataWindow.Label.Width=250")
dw1.Modify("DataWindow.Label.Name='Address1'")

LabelDispAttr.fontproperty

See DispAttr.fontproperty.

LastRowOnPage

Description
The last row currently visible in the DataWindow.

Applies to
DataWindows

Syntax
DataWindow .NET dot notation:

dw_control.LastRowOnPage

Describe argument:
"DataWindow.LastRowOnPage"

Examples

[C#]
int first, last;
first = dw1.FirstRowOnPage;
last = dw1.LastRowOnPage;

[Visual Basic]
setting = dw1.Describe("DataWindow.LastRowOnPage")

Left_Margin

Description
The size of the left margin of the DataWindow object.

Applies to
Style keywords

Syntax
DataWindowSyntaxFromSql:

Style ( Left_Margin = value )
### Legend

**Description**
The location of the legend in a Graph control in a DataWindow.

**Applies to**
Graph controls

**Syntax Describe and Modify argument:**

```
"graphname.Legend { = 'value' }
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>graphname</code></td>
<td>The name of the graph control for which you want to specify the location of the legend.</td>
</tr>
<tr>
<td><code>value</code></td>
<td><code>(exp)</code> A number indicating the location of the legend of a graph. Values are: 0 – None 1 – Left 2 – Right 3 – Top 4 – Bottom Value can be a quoted DataWindow expression.</td>
</tr>
</tbody>
</table>

**Usage**
In the painter Select the control and set the value in the Properties window, General category (applicable when the graph has more than one series).

**Examples**

```vbnet
setting = dw1.Describe("graph_1.Legend")
dw1.Modify("graph_1.Legend=2")
dw1.Modify("graph_1.Legend='2-tIf(dept_id=200,0,2)"")
```

### Legend.DispAttr.fontproperty

See DispAttr.fontproperty.


**Level**

**Description**
The grouping level.

Level is used in DataWindow syntax only for the Create method.

**Applies to**
Group keywords

**Syntax**

```
Group ( BY( colnum1, colnum2, ... ) ... Level = n ... )
```

---

**LinkUpdateOptions**

**Description**
When the OLE Object control is linked, the method for updating the link information. If the user tries to activate the OLE object and the DataWindow server cannot find the linked file, which breaks the link, LinkUpdateOptions controls whether the DataWindow server automatically displays a dialog box prompting the user to find the file. If you turn off the automatic dialog box, you can reestablish the link by calling the LinkTo or LinkUpdateDialog in code.

**Applies to**
OLE Object controls

**Syntax**

```
Describe and Modify argument:

"olecontrolname.LinkUpdateOptions ( = ' updatetype' )"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>olecontrolname</code></td>
<td>The name of the OLE Object control for which you want to get or set the link update method.</td>
</tr>
<tr>
<td><code>updatetype</code></td>
<td>A number specifying how broken links will be reestablished. Updatetype can be a quoted DataWindow expression. Values are:</td>
</tr>
<tr>
<td></td>
<td>• LinkUpdateAutomatic!</td>
</tr>
<tr>
<td></td>
<td>• LinkUpdateManual!</td>
</tr>
</tbody>
</table>

**Usage**

**In the painter**
Select the control and set the value in the Properties window, Options category.

**Examples**

[Visual Basic]

```vbnet
ls_data = dw1.Describe("ole_report.LinkUpdateOptions")
dw1.Modify("ole_report.LinkUpdateOptions='0'")
```
Message.Title

Description: The title of the dialog box that displays when an error occurs.

Applies to: DataWindows

Syntax: Describe and Modify argument:

```
"DataWindow.Message.Title { = 'titlestring' }
```

DataWindowSyntaxFromSql:

```
DataWindow(Message.Title = 'titlestring')
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>titlestring</td>
<td>A string containing the title for the title bar of the DataWindow dialog box that displays when an error occurs</td>
</tr>
</tbody>
</table>

Examples:

[Visual Basic]

```vbnet
setting = dw1.Describe("DataWindow.Message.Title")
dw1.Modify("DataWindow.Message.Title='Bad, Bad, Bad'")
```

[Visual Basic]

```vbnet
DWSyntaxGenerator.DataWindowSyntaxFromSql(myTrans, sql_syntax, "Style(...) DataWindow(Message.Title='Sales Report' ...) ...")
```

Moveable

Description: Whether the specified control in the DataWindow can be moved at runtime. Moveable controls should be in the DataWindow’s foreground.

Applies to: Button, Column, Computed Field, Graph, GroupBox, Line, OLE, Oval, Picture, Rectangle, Report, RoundRectangle, TableBlob, and Text controls

Syntax: Describe and Modify argument:

```
"controlname.Moveable { = number }
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlname</td>
<td>The control within the DataWindow for which you want to get or set the Moveable property that governs whether the user can move the control</td>
</tr>
<tr>
<td>number</td>
<td>A boolean number specifying whether the control is moveable. Values are: 0 – False, the control is not moveable. 1 – True, the control is moveable.</td>
</tr>
</tbody>
</table>
CHAPTER 3  DataWindow Object Properties

Usage  
**In the painter**  Select the control and set the value in the Properties window, Layout category.

Examples  
[Visual Basic]  
setting = dw1.Describe("bitmap_1.Moveable")  
dw1.Modify("bitmap_1.Moveable=1")

Name  
Description  The name of the control.
Applies to  Button, Column, Computed Field, Graph, GroupBox, InkPicture, Line, OLE, Oval, Picture, Rectangle, Report, RoundRectangle, TableBlob, and Text controls
Syntax  
Describe argument:

```
"controlname.Name"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlname</td>
<td>The control for which you want the name. For columns, you can specify the column number preceded by #.</td>
</tr>
</tbody>
</table>

Usage  
**In the painter**  Select the control and set the value in the Properties window, General category.

Examples  
[Visual Basic]  
setting = dw1.Describe("#4.Name")

Nest_Arguments  
Description  The values for the retrieval arguments of a nested report. The number of values in the list should match the number of retrieval arguments defined for the nested report.
Applies to  Report controls
Syntax  
Describe and Modify argument:

```
"reportname.Nest_Arguments { = list } "
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reportname</td>
<td>The name of the nested report for which you want to supply retrieval argument values.</td>
</tr>
</tbody>
</table>
Nested

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| list      | A list of values for the retrieval arguments of the nested report. The format for the list is:  
|           | ( ("arg1") {("arg2") {("arg3") { ... } } } ) |

Usage

The list is not a quoted string. It is surrounded by parentheses, and each argument value within the list is parenthesized, surrounded with double quotes, and separated by commas. If an argument is a literal string, use single quotes within the double quotes.

When changing the values for the retrieval arguments, you must supply values for all the retrieval arguments defined for the report. If you specify fewer or more arguments, an error will occur at runtime when the DataWindow retrieves its data.

To remove the report’s retrieval arguments, specify empty parentheses. If no arguments are specified, the user is prompted for the values at runtime.

In the painter Select the control and set the value in the Properties window, General category.

Examples

[Visual Basic]

```vbnet
setting = dw1.Describe("rpt_1.Nest_Arguments")
dw1.Modify("rpt_1.Nest_Arguments" = ("~cust_id~"),
(�"'Eastern'~"))
dw1.Modify("rpt_1.Nest_Arguments=()")
```

Nested

Description

Whether the DataWindow contains nested DataWindows. Values returned are Yes or No.

Applies to DataWindows

Syntax

Describe argument:

"DataWindow.Nested"

Examples

[Visual Basic]

```vbnet
setting = dw1.Describe("DataWindow.Nested")
```
CHAPTER 3 DataWindow Object Properties

NewPage (Group keywords)

Description Whether a change in the value of a group column causes a page break.

Applies to Group keywords

Syntax DataWindowSyntaxFromSql:

```
Group ( colnum1, colnum2 NewPage )
```

Examples

[Visual Basic]
```
DWSyntaxGenerator.DataWindowSyntaxFromSql(myTrans, sql_syntax, "Style(Type=Group) Group(#3 NewPage ResetPageCount)")
```

NewPage (Report controls)

Description Whether a nested report starts on a new page. NewPage applies only to reports in a composite DataWindow. Note that if the Trail_Footer property of the preceding report is set to No, the current report will be forced to begin on a new page regardless of the NewPage value.

Applies to Report controls

Syntax Describe and Modify argument:

```
"reportname.NewPage { = value } "
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reportname</td>
<td>The name of the report control for which you want to get or set the NewPage property.</td>
</tr>
<tr>
<td>value</td>
<td>Whether the report begins a new page.</td>
</tr>
<tr>
<td></td>
<td>Values are:</td>
</tr>
<tr>
<td></td>
<td>Yes – Start the report on a new page.</td>
</tr>
<tr>
<td></td>
<td>No – Do not start the report on a new page.</td>
</tr>
</tbody>
</table>

Usage In the painter Select the Report control in the Composite presentation style and set the value in the Properties window, General category.

Examples

[Visual Basic]
```
newpage_setting = dw1.Describe("rpt_1.NewPage")
dw1.Modify("rpt_1.NewPage=Yes")
```
NoUserPrompt

Description
Determines whether message boxes are displayed to the user during DataWindow processing.

Applies to
DataWindows

Syntax
Describe and Modify argument:
"DataWindow.NoUserPrompt { = ' value ' }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>A string specifying whether any message box requiring user intervention displays during DataWindow processing. Values are: Yes – No message box displays. No – (Default) Message boxes display when invoked during DataWindow processing.</td>
</tr>
</tbody>
</table>

Usage
Set the NoUserPrompt property to yes if the DataWindow is to be used in a batch process or in an EAServer environment when there is no possibility of end-user intervention. Dialog boxes you can prevent from displaying include the Error, Print, Retrieve, CrossTab, Expression, SaveAs, Import, Query, Filter, and Sort dialog boxes.

Examples
[Visual Basic]
dw1.Modify("DataWindow.NoUserPrompt=No")

Objects

Description
A list of the controls in the DataWindow object. The names are returned as a tab-separated list.

Applies to
DataWindows

Syntax
Describe argument:
"DataWindow.Objects"

Examples
[Visual Basic]
setting = dw1.Describe("DataWindow.Objects")
OLE.Client.property

**Description**
Settings that some OLE server applications use to identify the client’s information. The property values can be used to construct the title of the server window.

** Applies to**
DataWindows

**Syntax**
Describe and Modify argument:

```
"DataWindow.OLE.Client.property { = 'value' }
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>property</td>
<td>An OLE client property, as shown in the table below.</td>
</tr>
<tr>
<td>value</td>
<td>Values for the properties are shown in the table below. Value cannot be a DataWindow expression.</td>
</tr>
</tbody>
</table>

**Property for OLE.Client**

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>The client class for the DataWindow. The default is DataWindow.</td>
</tr>
<tr>
<td>Name</td>
<td>The client name for the DataWindow. The default is Untitled.</td>
</tr>
</tbody>
</table>

**Usage**

**In the painter**
Select the control and set the value in the Properties window, Definition category.

**Examples**

[Visual Basic]
```
ls_data = dw1.Describe("DataWindow.OLE.Client.Class")
dw1.Modify("DataWindow.OLE.Client.Class = 'PB'")
```

OLEClass

**Description**
The name of the OLE class for the TableBlob control.

** Applies to**
TableBlob controls

**Syntax**
Describe and Modify argument:

```
"tblobname.OLEClass { = 'oleclassname' }
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tblobname</td>
<td>The TableBlob column for which you want to get or set the class of server application.</td>
</tr>
<tr>
<td>oleclassname</td>
<td>(exp) A string specifying a class of an OLE server application installed on your system. Oleclassname is quoted and can be a DataWindow expression.</td>
</tr>
</tbody>
</table>
OverlapPercent

Usage
In the painter Select the control and set the value in the Properties window, Definition category.

Examples
[Visual Basic]
setting = dw1.Describe("blob_1.OLEClass")
dw1.Modify("blob_1.OLEClass='Word.Document'")

OverlapPercent

Description The percentage of overlap for the data markers (such as bars or columns) in different series in a graph.

Applies to Graph controls

Syntax Describe and Modify argument:

"graphname.OverlapPercent { = 'integer' }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>graphname</td>
<td>The name of the graph control in the DataWindow object for which you want to get or set the percentage of overlap.</td>
</tr>
<tr>
<td>integer</td>
<td>(exp) An integer specifying the percent of the width of the data markers that will overlap. Integer can be a quoted DataWindow expression.</td>
</tr>
</tbody>
</table>

Usage
In the painter Select the control and set the value in the Properties window, General category (applicable when a series has been specified).

Examples
[Visual Basic]
setting = dw1.Describe("graph_1.OverlapPercent")
dw1.Modify("graph_1.OverlapPercent=25")
Pen.property

Description Settings for a line or the outline of a control.

Applies to Line, Oval, Rectangle, and RoundRectangle controls

Syntax Describe and Modify argument:

"controlname.Pen.property { = value }"

Parameter Description

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlname</td>
<td>The name of the control whose Pen property you want to get or set.</td>
</tr>
<tr>
<td>property</td>
<td>A property that applies to the Pen characteristics of controlname, as listed in the table below.</td>
</tr>
<tr>
<td>value</td>
<td>The value of the property, as shown in the table below. Value can be a quoted DataWindow expression.</td>
</tr>
</tbody>
</table>

Property for Pen | Value

Color

(exp) A long specifying the color (the red, green, and blue values) to be used as the control’s line color.

Painter: Pen Color option.

Style

(exp) A number specifying the style of the line.

Values are:

- 0 – Solid
- 1 – Dash
- 2 – Dotted
- 3 – Dash-dot pattern
- 4 – Dash-dot-dot pattern
- 5 – Null (no visible line)

Painter: Pen Style option.

Width

(exp) A number specifying the width of the line in the unit of measure specified for the DataWindow.

Painter: Pen Width option (not available when Style is a value other than Solid).

Usage

In the painter Select the control and set values in the Properties window, Appearance category.

Examples

[Visual Basic]

```
setting = dw1.Describe("line_1.Pen.Width")
dw1.Modify("line_1.Pen.Width=10")
```
Perspective

Description
The distance from the front of the window at which the graph appears.

Applies to
Graph controls

Syntax
Describe and Modify argument:

```
    "graphname.Perspective { = 'integer' }
```

Parameter | Description
--- | ---

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>graphname</code></td>
<td>The name of the graph control in the DataWindow object for which you want to get or set the perspective.</td>
</tr>
<tr>
<td><code>integer</code></td>
<td><em>(exp)</em> An integer between 1 and 100 specifying how far away the graph appears. The larger the number, the greater the distance and the smaller the graph appears. Integer can be a quoted DataWindow expression.</td>
</tr>
</tbody>
</table>

Usage

**In the painter** Select the control and set the value in the Properties window, General category (available when a 3D graph type is selected).

Examples

[Visual Basic]
```
    setting = dw1.Describe("graph_1.Perspective")
    dw1.Modify("graph_1.Perspective=20")
```

Pie.DispAttr.fontproperty

See DispAttr.fontproperty.

PlotNullData

Description
Whether a continuous line is drawn between tics in a line graph when there is no data on the X and Y axes..

Applies to
Graph controls, Graph DataWindow objects

Syntax
Describe and Modify argument:

```
    "graphname.PlotNullData { = 'value' }
```

Parameter | Description
--- | ---

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>graphname</code></td>
<td>The name of the graph control in the DataWindow object for which you want to get or set the perspective.</td>
</tr>
</tbody>
</table>
CHAPTER 3  DataWindow Object Properties

### Pointer

**Description**
The image to be used for the mouse pointer when the pointer is over the specified control. If you specify a pointer for the whole DataWindow, the DataWindow server uses that pointer except when the pointer is over a control that also has a Pointer setting.

**Applies to**
DataWindow, Button, Column, Computed Field, Graph, GroupBox, Line, OLE, Oval, Picture, Rectangle, Report, RoundRectangle, TableBlob, and Text controls

**Syntax**
Describe and Modify argument:

```
"controlname(Pointer { = 'pointername' })"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlname</td>
<td>The name of the control in the DataWindow for which you want to get or set the pointer. Specify DataWindow to specify the pointer for the whole DataWindow.</td>
</tr>
<tr>
<td>pointername</td>
<td>(exp) A string specifying a value of the Pointer enumerated datatype or the name of a cursor file (.CUR) to be used for the pointer. (See the SetPointer method for a list of Pointer values.) Pointername can be a quoted DataWindow expression.</td>
</tr>
</tbody>
</table>

**Usage**

**In the painter**
Select the control and set the value in the Properties window, Appearance category. For the DataWindow object itself, select the Pointer category.
**Print.Preview.property**

**Description**
Properties that control the print preview of a DataWindow.

**Applies to**
DataWindows

**Syntax**
DataWindow .NET dot notation:

```
control.PrintProperties.property
```

Describe and Modify argument:

```
"DataWindow.Print.Preview.property { = value }
```

DataWindow Syntax From Sql:

```
DataWindow ( Print.Preview.property = value )
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>property</td>
<td>A property for print preview. Properties and their settings are listed in the table below.</td>
</tr>
<tr>
<td>value</td>
<td>The value to be assigned to the property. Value cannot be a DataWindow expression.</td>
</tr>
</tbody>
</table>

**Property for Print.Preview**

<table>
<thead>
<tr>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Buttons</td>
</tr>
<tr>
<td>Outline</td>
</tr>
</tbody>
</table>

**Buttons**
Whether buttons display in print preview.

Values are:
- Yes – Buttons are displayed.
- No – (Default) Buttons are not displayed.

Painter: Display Buttons – Print Preview.

**Outline**
Whether a blue line displays to show the location of the margins.

Values are:
- Yes – (Default) Margin outline is displayed.
- No – Margin outline is not displayed.

Painter: Print Preview Shows Outline
CHAPTER 3   DataWindow Object Properties

### Property for PrintPreview: Rulers
Whether the rulers display when the DataWindow object displays in preview mode.

Values are:
- Yes – Display the rulers.
- No – (Default) Do not display the rulers.

You can view rulers in Preview mode in the DataWindow painter. With the Preview view selected, select File>Print Preview, then File>Print Preview Rulers. However, the setting is not used at runtime. To see rulers at runtime, set PrintPreview.Rulers in code.

### Property for PrintPreview: Zoom
An integer indicating the zoom factor of the print preview. The default is 100%.

You can view different zoom percentages in Preview mode in the DataWindow painter. With the Preview view selected, select File>Print Preview, then File>Print Preview Zoom. However, the setting is not used at runtime. To change the zoom factor at runtime, set PrintPreview.Zoom in code.

### Usage
**In the painter** Select the DataWindow by deselecting all controls; then set values in the Properties window, Print Specifications category.

In DataWindow .NET, you can use the PrintProperties class to set print properties using dot notation. Note that the names of some properties of the PrintProperties class differ from the names of the DataWindow object property, and some take a different value.

See the description of the PrintProperties class in the online Help in Visual Studio .NET for a complete list of PrintPreview properties.

### Examples

```visualbasic
setting = dw1.Describe
("DataWindow.PrintPreview.Buttons")
dw1.Modify("DataWindow.PrintPreview.Buttons = 'Yes'")
dw1.PrintProperties.ShowPreviewButtons = 'Yes'
setting = dw1.Describe
("DataWindow.PrintPreview.Rulers")
dw1.Modify("DataWindow.PrintPreview.Rulers = 'Yes'")
dw1.PrintProperties.ShowPreviewRulers = 'Yes'
```

### See also
- Print.property
Print.property

Print.property

Description
Properties that control the printing of a DataWindow.

Applies to
DataWindows

Syntax
DataWindow .NET dot notation:

\[ dw\_control.PrintProperties.property \]

Describe and Modify argument:

"DataWindow.Print.property \{ = value \}"

DataWindowSyntaxFromSql:

DataWindow ( Print.property = value )

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>property</td>
<td>A property for printing. Properties and their settings are listed in the table below.</td>
</tr>
<tr>
<td>value</td>
<td>The value to be assigned to the property. Value cannot be a DataWindow expression.</td>
</tr>
</tbody>
</table>

Property for Print | Value

Buttons

Whether buttons display on the printed output.

Values are:

Yes – Buttons are displayed.
No – Buttons are not displayed.

CanUseDefault Printer

Whether a report can be printed on the default system printer if the printer specified by the PrinterName property is not valid.

ClipText

Whether the text of a static text field on a printed page is clipped to the dimensions of the text field when the text field has no visible border setting.

Values are:

Yes – The printed text does not overrun the text field.
No – (Default) The entire text can overrun the text field.

Text is automatically clipped for text fields with visible border settings even if this property is not set.

Collate

Whether printing is collated. Note that collating is usually slower since the print is repeated to produce collated sets.

Values are:

Yes – (Default) Collate the pages of the print job.
No – Do not collate.
### Property for Print

<table>
<thead>
<tr>
<th>Property for Print</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Color</strong></td>
<td>An integer indicating whether the printed output will be color or monochrome. Values are: 1 – Color 2 – Monochrome. The user can specify the value in the system’s Print dialog box if the printer driver supports it.</td>
</tr>
<tr>
<td><strong>Columns</strong></td>
<td>An integer specifying the number of newspaper-style columns the DataWindow will print on a page. For purposes of page fitting, the whole DataWindow is a single column. The default is 1.</td>
</tr>
<tr>
<td><strong>Columns.Width</strong></td>
<td>An integer specifying the width of the newspaper-style columns in the units specified for the DataWindow.</td>
</tr>
<tr>
<td><strong>Copies</strong></td>
<td>An integer indicating the number of copies to be printed. The user can also specify this value in the system’s Print Setup dialog box if the printer driver supports it. If you use both the Print.Copies property and the Print Setup dialog box to indicate that multiple copies should be printed, the total number of copies printed is the product of the two values.</td>
</tr>
<tr>
<td><strong>CustomPage.Length</strong></td>
<td>A long indicating the desired length of a custom paper size for printing. Use this property in conjunction with Print.CustomPage.Width and with Paper.Size set to 256.</td>
</tr>
<tr>
<td><strong>CustomPage.Width</strong></td>
<td>A long indicating the desired width of a custom paper size for printing. Use this property in conjunction with Print.CustomPage.Length and with Paper.Size set to 256.</td>
</tr>
<tr>
<td><strong>DocumentName</strong></td>
<td>A string containing the name that will display in the print queue when the user sends the contents of the DataWindow object to the printer.</td>
</tr>
<tr>
<td><strong>Duplex</strong></td>
<td>An integer indicating duplex or double-sided printing for printers capable of duplex printing. Values are: 0 – Default 1 – Normal (nonduplex) printing 2 – Short-edge binding (the long edge of the page is horizontal) 3 – Long-edge binding (the long edge of the page is vertical) The user can specify the value in the system’s Print dialog box if the printer driver supports it.</td>
</tr>
<tr>
<td><strong>Filename</strong></td>
<td>A string containing the name of the file to which you want to print the report. An empty string means send to the printer. Painter: Cannot be set in painter.</td>
</tr>
<tr>
<td><strong>Margin.Bottom</strong></td>
<td>An integer indicating the width of the bottom margin on the printed page in the units specified for the DataWindow. You can set Margin.Bottom when using DataWindowSyntaxFromSql to generate DataWindow syntax.</td>
</tr>
</tbody>
</table>
### Property for Print

<table>
<thead>
<tr>
<th>Property for Print</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Margin.Left</td>
<td>An integer indicating the width of the left margin on the printed page in the units specified for the DataWindow. You can set Margin.Left when using DataWindowSyntaxFromSql to generate DataWindow syntax.</td>
</tr>
<tr>
<td>Margin.Right</td>
<td>An integer indicating the width of the right margin on the printed page in the units specified for the DataWindow. You can set Margin.Right when using DataWindowSyntaxFromSql to generate DataWindow syntax.</td>
</tr>
<tr>
<td>Margin.Top</td>
<td>An integer indicating the width of the top margin on the printed page in the units specified for the DataWindow. You can set Margin.Top when using DataWindowSyntaxFromSql to generate DataWindow syntax.</td>
</tr>
<tr>
<td>Orientation</td>
<td>An integer indicating the print orientation. This property has no effect if the computer has no default printer. Values are: 0 – The default orientation for your printer 1 – Landscape 2 – Portrait</td>
</tr>
<tr>
<td>OverridePrintJob</td>
<td>Whether you want to override the print job print settings defined in the PrintOpen method with the print specifications of the DataWindow. Values are: Yes – Override the print job print settings. No – (Default) Do not override the print job print settings.</td>
</tr>
<tr>
<td>Page.Range</td>
<td>A string containing the numbers of the pages you want to print, separated by commas. You can also specify a range with a dash. For example, to print pages 1, 2, and 5 through 10, enter: “1,2, 5-10”. The empty string means print all. The user can specify the value in the system’s Print dialog box if the printer driver supports it.</td>
</tr>
<tr>
<td>Page.RangeInclude</td>
<td>An integer indicating what pages to print within the desired range. Values are: 0 – Print all. 1 – Print all even pages. 2 – Print all odd pages. The user can specify the value in the system’s Print dialog box if the printer driver supports it.</td>
</tr>
</tbody>
</table>
### Property for Print

**Paper.Size**  
An integer indicating the size of the paper used for the output:

- 0 – Default paper size for the printer
- 1 – Letter 8 1/2 x 11 in
- 2 – LetterSmall 8 1/2 x 11 in
- 3 – Tabloid 17 x 11 in
- 4 – Ledger 17 x 11 in
- 5 – Legal 8 1/2 x 14 in
- 6 – Statement 5 1/2 x 8 1/2 in
- 7 – Executive 7 1/4 x 10 1/2 in
- 8 – A3 297 x 420 mm
- 9 – A4 210 x 297 mm
- 10 – A4 Small 210 x 297 mm
- 11 – A5 148 x 210 mm
- 12 – B4 250 x 354 mm
- 13 – B5 182 x 257 mm
- 14 – Folio 8 1/2 x 13 in
- 15 – Quarto 215 x 275 mm
- 16 – 10x14 in
- 17 – 11x17 in
- 18 – Note 8 1/2 x 11 in
- 19 – Envelope #9 3 7/8 x 8 7/8
- 20 – Envelope #10 4 1/8 x 9 1/2
- 21 – Envelope #11 4 1/2 x 10 3/8
- 22 – Envelope #12 4 x 11 1/276
- 23 – Envelope #14 5 x 11 1/2
- 24 – C size sheet
- 25 – D size sheet
- 26 – E size sheet
- 27 – Envelope DL 110 x 220 mm
- 28 – Envelope C5 162 x 229 mm
- 29 – Envelope C3 324 x 458 mm
- 30 – Envelope C4 229 x 324 mm
- 31 – Envelope C6 114 x 162 mm
- 32 – Envelope C65 114 x 229 mm
- 33 – Envelope B4 250 x 353 mm
- 34 – Envelope B5 176 x 250 mm
- 35 – Envelope B6 176 x 125 mm
- 36 – Envelope 110 x 230 mm
- 37 – Envelope Monarch 3.875 x 7.5 in
- 38 – 6 3/4 Envelope 3 5/8 x 6 1/2 in
- 39 – US Std Fanfold 14 7/8 x 11 in
- 40 – German Std Fanfold 8 1/2 x 12 in
- 41 – German Legal Fanfold 8 1/2 x 13 in
- 255, 256 – User-defined paper size (see "Usage" below)
**Print.property**

<table>
<thead>
<tr>
<th>Property for Print</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Paper.Source</td>
<td>An integer indicating the bin that will be used as the paper source. The integer you use depends on the tray number used by the printer. (To determine the actual bin setting, you can query the printer with a utility that makes API calls to the printer driver.) Typical values are: 0 – Default 1 – Upper 2 – Lower 3 – Middle 4 – Manual 5 – Envelope 6 – Envelope manual 7 – Auto 8 – Tractor 9 – Smallfmt 10 – Largefmt 11 – Large capacity 14 – Cassette</td>
</tr>
<tr>
<td>Preview</td>
<td>Whether the DataWindow object is displayed in preview mode. Values are: Yes – Display in preview mode. No – (Default) Do not display in preview mode.</td>
</tr>
<tr>
<td>PrinterName</td>
<td>A string containing the name of the printer you want to use to print the DataWindow report. If the printer name is not specified or if the named printer cannot be found at runtime, print output can be directed to the default printer for the user’s machine by setting the CanUseDefaultPrinter property. Otherwise, an error is returned.</td>
</tr>
<tr>
<td>Prompt</td>
<td>Whether a Printer Setup dialog displays before a job prints so the user can change the paper or other settings for the current printer. Values are: Yes – (Default) Display a Printer Setup dialog. No – Do not display a Printer Setup dialog. Choosing Cancel in the Printer Setup dialog dismisses the Setup dialog; it does not cancel printing. To allow the user to cancel printing, see the Print method. For DataStores, this property is ignored; a dialog is never displayed.</td>
</tr>
</tbody>
</table>
In the painter  Select the DataWindow by deselecting all controls; then set values in the Properties window, Print Specifications category.

In DataWindow .NET, you can use the PrintProperties class to set print properties using dot notation. Note that the names of some properties of the PrintProperties class differ from the names of the DataWindow object property, and some take a different value. For example, the Color DataWindow object property takes an integer value, and the equivalent ColorOutput PrintProperties property takes a boolean value.

See the description of the PrintProperties class in the online Help in Visual Studio .NET for a complete list of properties.


For example:

```
// DataWindow Units set to 1/1000 inch
// 9.875 inches long
dwl.Modify("DataWindow.Print.CustomPage.Length=9875")
// 7.375 inches wide
dwl.Modify("DataWindow.Print.CustomPage.Width=7375")
```
With Paper.Size set to 256, Length and Width are in millimeters:

//25.4 centimeters long

//19.5 centimeters wide


Examples
[Visual Basic]
dw1.PrintProperties.PaperSize = 3
strData = dw1.Describe("DataWindow.Print.Scale")
dw1.Modify("DataWindow.Print.Paper.Size = 3")
dw1.Modify("DataWindow.Print.Margin.Top=500")
dw1.PrintProperties.ShowButtons = 'Yes'
setting = dw1.Describe("DataWindow.Print.Buttons")
dw1.Modify("DataWindow.Print.Buttons = 'Yes'")

See also
Print.Preview.property

Printer

Description
The name of the printer for printing the DataWindow as specified in the system’s printer selection dialog box.

Applies to
DataWindows

Syntax
Describe and Modify argument:

"DataWindow.Printer" { = printername }

Parameter | Description
-----------|-------------
printername | Name of the printer you want to use for your DataWindow

Usage
The printer you select for a DataWindow does not affect the system default printer. To specify a network-connected printer, you must use a fully specified network printer name:

dw1.SetProperty("DataWindow.Printer", "\srv\pr-6")

If you specify a DataWindow printer, but the printer is not found, the DataWindow engine does not attempt to print to a default device.
Examples

The following example changes the DataWindow printer (but does not affect the system default printer device):

[Visual Basic]
dw1.Modify ('DataWindow.Printer="My LaserJet 3" ')

You can display the DataWindow printer with the following calls:

ls_dwprinter = dw1.Describe("DataWindow.Printer")

Processing

Description

The type of processing required to display the data in the selected presentation style.

Applies to

DataWindows

Syntax

Describe argument:

"DataWindow.Processing"

Return values are:

0 – (Default) Form, group, n-up, or tabular
1 – Grid
2 – Label
3 – Graph
4 – Crosstab
5 – Composite
8 – TreeView
9 – TreeView with Grid

Examples

[Visual Basic]
setting = dw1.Describe("DataWindow.Processing")

Protect

Description

The protection setting of a column. The Protect property overrides tab order settings. When a column is protected, the user cannot edit it even if the column’s tab order is greater than 0.

Applies to

A column

Syntax

Describe and Modify argument:

"columnname.Protect { = ' integer ' }"
QueryClear

Usage
A user cannot change a column value if any one of these conditions is true:

- TabSequence is 0
- Edit.DisplayOnly is Yes when the column has the Edit edit style
- Protect is 1

Only the Protect property allows you to specify a conditional expression that protects some values in the column but not others.

In the painter
Select the control and set the value in the Properties window, Behavior category (using a conditional expression).

Examples

```
setting = dw1.Describe("emp_stat.Protect")
dw1.Modify("emp_stat.Protect=1")
dw1.Modify("emp_stat.Protect='1-tIf(IsRowNew(),0,1)'")
```

QueryClear

Description
Removes the WHERE clause from a query. Note that the only valid setting is Yes.

Applies to
DataWindows

Syntax
Modify argument:

```
"DataWindow.QueryClear { = value }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Remove the WHERE clause from a query. Yes is the only valid value.</td>
</tr>
</tbody>
</table>

Examples

```
dw1.Modify("DataWindow.QueryClear=yes")
```
CHAPTER 3   DataWindow Object Properties

**QueryMode**

**Description**

Whether the DataWindow is in query mode. In query mode, the user can specify the desired data by entering WHERE criteria in one or more columns.

---

**DataWindow presentation styles**

You cannot use QueryMode with DataWindow objects that use any of the following presentation styles: Label, Crosstab, and Graph.

---

**Applies to**

DataWindows

**Syntax**

DataWindow .NET dot notation:

\[ dw\_control.QueryMode \]

Describe and Modify argument:

"DataWindow.QueryMode \{ = value \}"

---

**Parameter** | **Description**
--- | ---
value | Whether the DataWindow is in query mode. Values are: Yes – Query mode is enabled. No – Query mode is disabled. In DataWindow .NET, set the value of the DataWindowControl property to true or false.

**Usage**

After the user specifies retrieval criteria in query mode, subsequent calls to Retrieve can use the new criteria. To retrieve data based on user selection, change the query mode back to No and use AcceptText to accept the user’s specification before the next call to Retrieve.

Setting QuerySort to Yes also puts the DataWindow into query mode, changing the QueryMode property’s value to Yes.

**Query mode and secondary DataWindows**

When you are sharing data, you cannot turn on query mode for a secondary DataWindow. Trying to set the QueryMode or QuerySort properties results in an error.

**Buffer manipulation and query mode**

A DataWindow cannot be in query mode when you call the RowsCopy method.

**Examples**

[Visual Basic]

dw1.QueryMode = true

setting = dw1.Describe("DataWindow.QueryMode")
dw1.Modify("DataWindow.QueryMode=yes")
QuerySort

Description
Whether the result set is sorted when the DataWindow retrieves the data specified in query mode. When query sort is on, the user specifies sorting criteria in the first row of the query form.

DataWindow presentation styles
You cannot use QuerySort with DataWindow objects that use any of the following presentation styles: N-Up, Label, Crosstab, and Graph.

Applies to DataWindows

Syntax
Describe and Modify argument:

"DataWindow.QuerySort { = value }"

Parameter | Description
---|---
value | Whether the data retrieved from query mode specifications is sorted.

Values are:
Yes – Sorting is enabled.
No – Sorting is disabled.

Usage
If the DataWindow is not already in query mode, setting QuerySort to Yes also sets QueryMode to Yes, putting the DataWindow in query mode.

When you set QuerySort to No, the DataWindow remains in query mode until you also set QueryMode to No.

Query mode and secondary DataWindows
When you are sharing data, you cannot turn on query mode for a secondary DataWindow. Trying to set the QueryMode or QuerySort properties results in an error.

Examples

[Visual Basic]
setting = dw1.Describe("DataWindow.QuerySort")
dw1.Modify("DataWindow.QuerySort=yes")

RadioButtons.property

Description
Properties that control the appearance and behavior of a column with the RadioButton edit style.

Applies to Column controls
CHAPTER 3   DataWindow Object Properties

Syntax

DataWindow .NET dot notation:

```csharp
(( RadioButton) colname/EditStyle).property
CType(colname/EditStyle, RadioButton).property
```

Describe and Modify argument:

```
"columnname.RadioButtons.property { = value }
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columnname</td>
<td>The name of the column that has the RadioButton edit style.</td>
</tr>
<tr>
<td>property</td>
<td>A property for the RadioButton column. Properties and their settings are listed in the table below.</td>
</tr>
<tr>
<td>value</td>
<td>The value to be assigned to the property. For RadioButton properties, value cannot be a DataWindow expression.</td>
</tr>
</tbody>
</table>

Property for Radiobuttons

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>3D or ThreeD</td>
<td>Whether the radio buttons are 3D. Values are:</td>
</tr>
<tr>
<td></td>
<td>Yes – Make the buttons 3D.</td>
</tr>
<tr>
<td></td>
<td>No – Do not make the buttons 3D.</td>
</tr>
<tr>
<td></td>
<td>When using dot notation, use the term ThreeD instead of 3D.</td>
</tr>
<tr>
<td>Columns</td>
<td>An integer constant specifying the number of columns of radio buttons.</td>
</tr>
<tr>
<td>LeftText</td>
<td>Whether the text labels for the radio buttons are on the left side.</td>
</tr>
<tr>
<td></td>
<td>Values are:</td>
</tr>
<tr>
<td></td>
<td>Yes – The text is on the left of the radio buttons.</td>
</tr>
<tr>
<td></td>
<td>No – The text is on the right of the radio buttons.</td>
</tr>
<tr>
<td>Scale</td>
<td>Whether the circle is scaled to the size of the font. Scale has an effect only when 3D is No.</td>
</tr>
<tr>
<td></td>
<td>Values are:</td>
</tr>
<tr>
<td></td>
<td>Yes – Scale the circles.</td>
</tr>
<tr>
<td></td>
<td>No – Do not scale the circles.</td>
</tr>
</tbody>
</table>

Usage

In the painter  Select the control and set the value in the Properties window, Behavior category when EditStyle is RadioButtons.

In DataWindow .NET, you can use the RadioButton class to set RadioButton Edit style properties using dot notation. See the description of the RadioButton class in the online Help in Visual Studio .NET for property names.

Examples

[Visual Basic]

```vbnet
setting = dw1.Describe("empg.RadioButtons.LeftText")
dw1.Modify("emp_gender.RadioButtons.LeftText=no")
dw1.Modify("emp_gender.RadioButtons.3D=Yes")
```

DataWindow Object Reference 315
Range

Description
The rows in the DataWindow used in the graph control. Range can be all rows, the rows on the current page, a group that you have defined for the DataWindow, or the current row (OLE Object controls only).

Applies to
Graph and OLE Object controls

Syntax
Describe argument:
"controlname.Range"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlname</td>
<td>The name of the graph control within the DataWindow that will display the graphed rows or the name of the OLE Object control that holds an OLE object to which the specified range of rows will be transferred.</td>
</tr>
</tbody>
</table>

Usage
Possible values are:
-2 – The current row (OLE Object controls only)
-1 – The rows on a single page in the DataWindow object
0 – All the rows in the DataWindow object
n – The number of a group level in the DataWindow object

GroupBy and Target also affect the data that is transferred to the OLE object.

In the painter Select the control and set the value in the Properties window, Data category. For a Graph control, set the DataRows property.

Examples
[Visual Basic]
strRange = dw1.Describe("graph_salary.Range")
strRange = dw1.Describe("ole_report.Range")
CHAPTER 3  DataWindow Object Properties

ReadOnly

Description  Whether the DataWindow is read-only.
Applies to   DataWindows
Syntax       Describe and Modify argument:
             "DataWindow.ReadOnly { = value }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Whether the DataWindow is read-only.</td>
</tr>
<tr>
<td></td>
<td>Values are:</td>
</tr>
<tr>
<td></td>
<td>Yes – Make the DataWindow read-only.</td>
</tr>
<tr>
<td></td>
<td>No – (Default) Do not make the DataWindow read-only.</td>
</tr>
</tbody>
</table>

Examples  [Visual Basic]
setting = dw1.Describe("DataWindow.ReadOnly")
dw1.Modify("DataWindow.ReadOnly=Yes")

ReplaceTabWithSpace

Description  Whether tab characters embedded in the data for a DataWindow display as square boxes when the row is not the current row.
Applies to   DataWindows
Syntax       Describe and Modify argument:
             "DataWindow.ReplaceTabWithSpace { = value }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Whether tab characters embedded in the data for a DataWindow are replaced with spaces.</td>
</tr>
<tr>
<td></td>
<td>Values are:</td>
</tr>
<tr>
<td></td>
<td>Yes – Replace each tab character with four spaces.</td>
</tr>
<tr>
<td></td>
<td>No – (Default) Do not replace tab characters.</td>
</tr>
</tbody>
</table>

Examples  [Visual Basic]
str = dw1.Describe("DataWindow.ReplaceTabWithSpace")
dw1.Modify("DataWindow.ReplaceTabWithSpace=Yes")

DataWindow Object Reference  317
Report

Description Whether the DataWindow is a read-only report.
Applies to Style keywords
Syntax DataWindowSyntaxFromSql:
  Style ( Report = value )

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| value     | Whether the DataWindow is a read-only report, similar to a DataWindow created in the Report painter. Values are:
  Yes – The DataWindow is a read-only report.
  No – The DataWindow is not read-only. |

Examples [Visual Basic]
DWSyntaxGenerator.DataWindowSyntaxFromSql(myTrans, sqlstring, 'Style(...Report = yes ...)')

ResetPageCount

Description Specifies that a change in the value of the group column causes the page count to begin again at 0.
Applies to Group keywords
Syntax DataWindowSyntaxFromSql:
  Group (col1 {col2 ...} ... ResetPageCount )

Examples [Visual Basic]
DWSyntaxGenerator.DataWindowSyntaxFromSql(myTrans, sqlstring, "Style(Type=Group) Group(#3 NewPage ResetPageCount)")

Resizeable

Description Whether the user can resize the specified control.
Applies to Button, Column, Computed Field, Graph, GroupBox, Line, OLE, Oval, Picture, Rectangle, Report, RoundRectangle, TableBlob, and Text controls
Syntax Describe and Modify argument:
"controlname.Resizeable { = value }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlname</td>
<td>The control within the DataWindow whose Resizeable setting you want to get or set.</td>
</tr>
</tbody>
</table>
| value         | A boolean number indicating whether controlname can be resized.  
                Values are:  
                0 – (False) The control cannot be resized.  
                1 – (True) The control can be resized. |

**Usage**  
**In the painter**  
Select the control and set the value in the Properties window, Layout category.  
When you make the control resizable, set the Border property to the resizable border so the user knows it is resizable.

**Examples**  
[Visual Basic]  
setting = dw1.Describe("graph_1.Resizeable")  
dw1.Modify("graph_1.Resizeable=1")  
dw1.Modify("bitmap_1.Resizeable=0")

**Retrieve**  
**Description**  
The SQL statement for the DataWindow.
Retrieve is set in DataWindow syntax only for the Create method.

**Applies to**  
Table keywords

**Syntax**  
Table ( ... Retrieve = selectstatement ... )

**Retrieve.AsNeeded**  
**Description**  
Whether rows will be retrieved only as needed from the database. After the application calls the Retrieve method to get enough rows to fill the visible portion of the DataWindow, additional rows are “needed” when the user scrolls down to view rows that have not been viewed yet.

**Applies to**  
DataWindows

**Syntax**  
Describe and Modify argument:  
"DataWindow.Retrieve.AsNeeded { = 'value' }"
**RichText.property**

**Parameter** | **Description**
--- | ---
`value` | Whether rows will be retrieved only as needed from the database. Values are:
- Yes – Rows will be retrieved only as needed.
- No – All rows will be retrieved when the Retrieve method is called.

**Usage**

**In the painter** Set the value using Rows>Retrieve Options>Rows As Needed.

**Examples**

[Visual Basic]
```
setting = dw1.Describe("DataWindow.Retrieve.AsNeeded")
dw1.Modify("DataWindow.Retrieve.AsNeeded=Yes")
```

**RichText.property**

Description Properties for the DataWindow RichText presentation style (not in DataWindow .NET).

Applies to DataWindows

Syntax Describe and Modify argument:
```
"DataWindow.RichText.property{ = value }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>property</code></td>
<td>A property for the DataWindow RichText presentation style. Properties and appropriate values are listed in the table below.</td>
</tr>
<tr>
<td><code>value</code></td>
<td>A value to be assigned to the property.</td>
</tr>
</tbody>
</table>

**Rotation**

Description The degree of left-to-right rotation for the graph control within the DataWindow when the graph has a 3D type.

Applies to Graph controls

Syntax Describe and Modify argument:
```
"graphname.Rotation = { int" |
```

320 DataWindow .NET
### Parameter: `graphname`  
**Description:** The name of the Graph control for which you want to get or set the rotation.

### Parameter: `integer`  
**Description:** The degree of rotation for the graph. Effective values range from -90 to 90. Integer can be a quoted DataWindow expression.

#### Usage
**In the painter** Select the control and set the value in the Properties window, General category (enabled when a 3D graph type is selected).

#### Examples
```
[Visual Basic]
setting = dw1.Describe("graph_1.Rotation")
dw1.Modify("graph_1.Rotation=25")
dw1.Modify("graph_1.Rotation='1-tHour(Now())'")
```

### Row.Resize
**Description:** Whether the user can use the mouse to change the height of the rows in the detail area of the DataWindow.

**Applies to:** DataWindows

**Syntax** Describe and Modify argument:
```
"DataWindow.Row.Resize { = value }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| `value`   | Whether the user can resize the rows in the detail area. Values are:  
|           | • 1 – Yes, the user can resize the rows.  
|           | • 0 – No, the user cannot resize the rows. |

#### Usage
**In the painter** Select the DataWindow by deselecting all controls; then set the value in the Properties window, General category (available when the presentation style is Grid or Crosstab).

#### Examples
```
[Visual Basic]
setting = dw1.Describe("DataWindow.Row.Resize")
dw1.Modify("DataWindow.Row.Resize=0")
```

### Rows_Per_Detail
**Description:** The number of rows in the detail area of an n-up DataWindow object. This property should be 1 unless the Type property for the Style keyword is Tabular.

**Applies to:** DataWindows
Selected

Syntax

Describe argument:

"DataWindow.Rows_Per_Detail"

DataWindow_SyntaxFromSql:

DataWindow ( ... Rows_Per_Detail = \text{n} ... )

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>\text{n}</td>
<td>A long specifying the number of rows in each column</td>
</tr>
</tbody>
</table>

Examples

[Visual Basic]

DWSyntaxGenerator.DataWindowSyntaxFromSql(myTrans, sqlstring, 'DataWindow(...Rows_Per_Detail = 12 ...)')

Selected

Description

A list of selected controls within the DataWindow.

Applies to

DataWindows

Syntax

Describe and Modify argument:

"DataWindow.Selected = 'list'"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| list      | A list of the controls you want to select. In the list you designate a group of controls by specifying a range of row numbers and a range of controls in the format: \text{startrow}/\text{endrow}/\text{startcontrol}/\text{endcontrol}

To specify more than one group, separate each group with a semicolon:

\text{startrow1}/\text{endrow1}/\text{startobj1}/\text{endobj1};\text{startrow2}/\text{endrow2}/\text{startobj2}/\text{endobj2};...

Do not include spaces in the string. You must use column names, not column numbers. |

Examples

[Visual Basic]

setting = dw1.Describe("DataWindow.Selected")
dw1.Modify("DataWindow.Selected=" _
"'1/10/emp_id/emp_name;12/23/salary/status'")

DataWindow .NET
**Selected.Data**

Description: A list describing the selected data in the DataWindow. Each column’s data is separated by a tab and each row is on a separate line.

Applies to: DataWindows (Crosstab and Grid presentation styles only)

Syntax: Describe argument:

```
"DataWindow.Selected.Data"
```

Examples:

```
[Visual Basic]
setting = dw1.Describe("DataWindow.Selected.Data")
```

**Selected.Mouse**

Description: Whether the user can use the mouse to select columns.

Applies to: DataWindows

Syntax: Describe and Modify argument:

```
"DataWindow.Selected.Mouse { = value }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>Whether the user can use the mouse to select columns. Values are:</td>
</tr>
<tr>
<td></td>
<td>• Yes – The mouse can be used.</td>
</tr>
<tr>
<td></td>
<td>• No – The mouse cannot be used.</td>
</tr>
</tbody>
</table>

Usage: **In the painter** Select the DataWindow by deselecting all controls; then set the value in the Properties window, General category (available when the presentation style is Grid or Crosstab).

Examples:

```
[Visual Basic]
setting = dw1.Describe("DataWindow.Selected.Mouse")
dw1.Modify("DataWindow.Selected.Mouse = Yes")
```

**Series**

See Axis, Axis.property, and DispAttr.fontproperty.
ShadeColor

Description
The color used for shading the back edge of the series markers when the
graph's type is 3D. ShadeColor has no effect unless Series.ShadeBackEdge is
1 (Yes). If ShadeBackEdge is 0, the axis plane is the same color as the
background color of the graph.

Applies to
Graph controls

Syntax
Describe and Modify argument:

"graphname.ShadeColor { = 'long' }

Parameter | Description
---|---
graphname | The Graph control in the DataWindow for which you want to
shade color.
long | (exp) A long number converted to a string specifying the color
of the shading for axes of a 3D graph.

You can use the RGB function in a DataWindow expression to
calculate the desired color value.

Long can be a quoted DataWindow expression.

Usage
To set the shade color for individual series markers, such as bars or pie slices,
use the method SetDataStyle.

In the painter
Select the control and set the value in the Properties window,
General category.

Examples

[Visual Basic]
setting = dw1.Describe("graph_1.ShadeColor")
dw1.Modify("graph_1.ShadeColor=16600000")
dw1.Modify("graph_1.ShadeColor=String(RGB(90,90,90))")
dw1.Modify("graph_1.ShadeColor='0~t If(salary>50000,
String(RGB(100,90,90)), String(RGB(90,90,100)) )' ")"

ShowBackColorOnXP

Description
Whether the background color that you select for a button displays on
Windows XP.

Applies to
DataWindow objects

Syntax
Describe and Modify argument:

"DataWindow.ShowBackColorOnXP{ = value }"
The Background.Color property is not supported for buttons on Windows XP by default because the current XP theme controls the appearance of the button. In the painter, set the Show Backcolor on XP property in the General category of the Properties window for the DataWindow object. The background color you selected will display in Preview mode.

Examples

dw1.Modify("DataWindow.ShowBackColorOnXP = yes")

ShowDefinition

Description
Whether the DataWindow definition will display. The DataWindow will display the column names instead of data.

Applies to
DataWindows

Syntax
Describe and Modify argument:

"DataWindow.ShowDefinition { = 'value' }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>(exp) Whether the column names will display. Values are: - Yes – Display the column names. - No – Do not display the data, if any. Value can be a quoted DataWindow expression.</td>
</tr>
</tbody>
</table>

Examples

[Visual Basic]

setting = dw1.Describe("DataWindow.ShowDefinition")
dw1.Modify("DataWindow.ShowDefinition=Yes")
SizeToDisplay

Description
Whether the graph should be sized automatically to the display area.

Applies to
Graph controls

Syntax
Describe and Modify argument:

"graphname.SizeToDisplay { = ' value ' }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>graphname</td>
<td>The graph control in the DataWindow for which you want to get or set adjustability.</td>
</tr>
<tr>
<td>value</td>
<td>(exp) A boolean number specifying whether to adjust the size of the graph to the display. Values are:</td>
</tr>
<tr>
<td></td>
<td>• 0 – False, do not adjust the size of the graph.</td>
</tr>
<tr>
<td></td>
<td>• 1 – True, adjust the size of the graph.</td>
</tr>
<tr>
<td></td>
<td>Value can be a quoted DataWindow expression.</td>
</tr>
</tbody>
</table>

Usage
In the painter  Select the control and set the value in the Properties window, Appearance category.

Examples

[Visual Basic]
setting = dw1.Describe("graph_1.SizeToDisplay")
dw1.Modify("graph_1.SizeToDisplay=0")

SlideLeft

Description
Whether the control moves to the left when other controls to the left leave empty space available. This property is for use with read-only controls and printed reports. It should not be used with data entry fields or controls.

Applies to
Button, Column, Computed Field, Graph, GroupBox, Line, Oval, Picture, Rectangle, Report, RoundRectangle, TableBlob, and Text controls

Syntax
Describe and Modify argument:

"controlname.SlideLeft { = ' value ' }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlname</td>
<td>The name of the control for which you want to get or set the Slide setting.</td>
</tr>
</tbody>
</table>
**CHAPTER 3   DataWindow Object Properties**

### SlideLeft

**Description**
Whether the control slides left when there is empty space to its left. Values are:
- Yes – The control will slide left into available space.
- No – The control will remain in position.

**Value** can be a quoted DataWindow expression.

**Usage**
In the painter Select the control and set the value in the Properties window, Layout category. This property is not supported in Web DataWindows.

**Examples**

```
[Visual Basic]
setting = dw1.Describe("graph_1.SlideLeft")
dw1.Modify("emp_lname.SlideLeft=yes")
```

### SlideUp

**Description**
Whether the control moves up when other controls above it leave empty space available. This property is for use with read-only controls and printed reports. It should not be used with data entry fields or controls.

**Applies to**
Button, Column, Computed Field, Graph, GroupBox, Line, Oval, Picture, Rectangle, Report, RoundRectangle, TableBlob, and Text controls

**Syntax**
Describe and Modify argument:

```
"controlname.SlideUp { = 'value' }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlname</td>
<td>The name of the control for which you want to get or set the Slide setting.</td>
</tr>
</tbody>
</table>
| value      | (exp) How the control slides up when there is empty space above it. Values are:  
  - AllAbove – Slide the control up if all the controls in the row above it are empty.  
  - DirectlyAbove – Slide the column or control up if the controls directly above it are empty.  
  - No – The control will not slide up.  
  Value can be a quoted DataWindow expression. |

**Usage**
In the painter Select the control and set the value in the Properties window, Layout category. This property is not supported in Web DataWindows.
Sort

Examples

[Visual Basic]
setting = dw1.Describe("graph_1.SlideUp")
dw1.Modify("emp_lname.SlideUp=no")

Sort
Description
Sort criteria for a newly created DataWindow. To specify sorting for existing DataWindows, see the SetSort and Sort methods.
Applies to
Table keywords in DataWindow syntax
Syntax
DataWindow syntax for Create method:

Table ( ... Sort = stringexpression ... )

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>stringexpression</td>
<td>A string whose value represents valid sort criteria. See the SetSort method for the format for sort criteria. If the criteria string is null, the DataWindow server prompts for a sort specification when it displays the DataWindow.</td>
</tr>
</tbody>
</table>

Spacing
Description
The gap between categories in a graph.
Applies to
Graph controls
Syntax
Describe and Modify argument:

"graphname.Spacing { = 'integer' }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>graphname</td>
<td>The name of the graph control in the DataWindow for which you want to get or set the spacing.</td>
</tr>
<tr>
<td>integer</td>
<td>(exp) An integer specifying the gap between categories in the graph. You specify the value as a percentage of the width of the data marker. For example, in a bar graph, 100 is the width of one bar, 50 is half a bar, and so on. Integer can be a DataWindow expression.</td>
</tr>
</tbody>
</table>

Usage
In the painter
Select the control and set the value in the Properties window, General category.
Examples

[Visual Basic]
setting = dw1.Describe("graph_1.Spacing")
dw1.Modify("graph_1.Spacing=120")

DataWindow .NET
**Sparse**

Description: The names of repeating columns that will be suppressed in the DataWindow.

Applies to: DataWindows

Syntax: Describe and Modify argument:

```
"DataWindow.Sparse \{ = ' list ' \}"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>list</code></td>
<td><code>(exp)</code> A tab-separated list of column names to be suppressed. <code>List</code> can be a quoted DataWindow expression.</td>
</tr>
</tbody>
</table>

Create method (include at the end of the DataWindow syntax):

```syntax
Sparse ( names = "col1~tcol2~tcol3 ...")
```

Usage: **In the painter** Set the value using Rows>Suppress Repeating Values. This property is not supported in Web DataWindows.

Examples

```visualbasic
setting = dw1.Describe("DataWindow.Sparse")
dw1.Modify("DataWindow.Sparse='col1~tcol2'")
```

**Storage**

Description: The amount of virtual storage in bytes that has been allocated for the DataWindow object.

Applies to: DataWindows

Syntax: Describe argument:

```
"DataWindow.Storage"
```

Usage: **Canceling a query that uses too much storage** You can check this property in the script for the RetrieveRow event in the DataWindow control and cancel a query if it is consuming too much storage.

Examples

```visualbasic
setting = dw1.Describe("DataWindow.Storage")
```

**StoragePageSize**

Description: The default page size for DataWindow storage.

Applies to: DataWindows
### Summary.property

**Syntax**

Describe and Modify argument:

```
"DataWindow.StoragePageSize { = 'size' }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| size       | Two values are provided to enable the DataWindow to use the available virtual memory most efficiently in the current environment:  
            |  
            |  
            | LARGE (Recommended)  
            | MEDIUM               |

**Usage**

Set this property to avoid out of memory errors when performing large retrieve, import, or RowsCopy operations. The property must be set **before** the operation is invoked.

**Examples**

```visualbasic
[Visual Basic]
dw1.Modify("datawindow.storagepagesize='LARGE'")
```

### Summary.property

See Bandname.property.

---

### SuppressEventProcessing

**Description**

Whether the ButtonClicked or ButtonClicking event is fired for this particular button.

**Applies to**

Button controls

**Syntax**

Describe and Modify argument:

```
"buttonname.SuppressEventProcessing { = 'value' }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>buttonname</td>
<td>The name of the button control for which you want to suppress event processing.</td>
</tr>
</tbody>
</table>
| value      | Whether event processing is to occur.  
            | Values are:  
            | Yes – The event should not be fired.  
            | No – The event should be fired (default). |

**Usage**

In the painter  
Select the control and set the value in the Properties window, General category.
Examples

[Visual Basic]
setting = dw1.Describe("b_name.SuppressEventProcessing")
        dw1.Modify("b_name.SuppressEventProcessing = 'No'")

Syntax
Description
The complete syntax for the DataWindow.
Applies to
DataWindows
Syntax
DataWindow .NET dot notation:

\[\text{dw\_control}\text{.Syntax}\]

Describe argument:

"DataWindow.Syntax"

Examples

[C#]
String dwSyntax;
dwSyntax = dw1.Syntax;

[Visual Basic]
setting = dw1.Describe("DataWindow.Syntax")

Syntax.Data
Description
The data in the DataWindow object described in parse format (the format required by the DataWindow parser).
Applies to
DataWindows
Syntax
Describe argument:

"DataWindow.Syntax.Data"
Usage
Use this property with the Syntax property to obtain the description of the DataWindow object and the data. Using this information, you can create a syntax file that represents both the structure and data of a DataWindow at an instant in time. You can then use the syntax file as a DropDownDataWindow containing redefined data at a single location or to mail this as a text object.
**Syntax.Modified**

**Description**
Whether the DataWindow syntax has been modified by a function call or user intervention. Calling the **Modify**, **SetSort**, or **SetFilter** method or changing the size of the DataWindow grid automatically sets **Syntax.Modified** to Yes.

**Applies to**
DataWindows

**Syntax**
Describe and Modify argument:
```
"DataWindow.Syntax.Modified { = value }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| value     | Whether the DataWindow syntax has been modified. Values are:  
  - Yes – DataWindow syntax has been modified.  
  - No – DataWindow has not been modified. |

**Usage**
Use this property in **Modify** to set **Syntax.Modified** to No after you cause a change in the syntax that does not affect the user (such as setting preview on).

**Examples**
```
[Visual Basic]
setting = dw1.Describe("DataWindow.Syntax.Modified")
dw1.Modify("DataWindow.Syntax.Modified=No")
```

---

**Table (for Create)**

**Description**
The section of the DataWindow syntax that specifies information about the DataWindow’s database table, including the name of the update table.

Use Table in DataWindow syntax for the Create method.

**Syntax**
Does not apply.

**Usage**
Use this property to redefine a DataWindow result set. You can add a column, change the datatype of a column, or make other changes to the table section of your DataWindow involving properties that are not accessible through **Modify** calls.

**Caution**
When you use this property to redefine the result set, you must redefine the table section in its entirety.
You can call the `GetItem` and `SetItem` methods to access columns added using this property, but the columns do not display in the DataWindow unless you call `Modify("create column(...)")` to add them.

To redefine your table section:

1. Export your DataWindow object to a DOS file.
2. Copy only the table section into your script.
3. Modify the table section to meet your needs.
4. Put the new table definition into a string variable. Change existing double quotation marks ("') in the string to single quotation marks (') and change the tilde quotation marks to tilde tilde single quotation marks (~~').
5. Call `Modify`. Modifying the table section of your DataWindow causes the DataWindow to be reset.
6. (Optionally) Call `Modify` to add the column to the DataWindow display.

### Table (for InkPicture and TableBlobs)

**Description**
The name of the database table that contains the blob(s).

**Applies to**
InkPicture and TableBlob controls

**Syntax**
Describe and Modify argument:

```
"controlname.Table { = 'tablename' }
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlname</td>
<td>The name of the control in the DataWindow.</td>
</tr>
<tr>
<td>tablename</td>
<td>(exp) A string specifying the name of the table that contains the blob data. Tablename can be a quoted DataWindow expression.</td>
</tr>
</tbody>
</table>

**Usage**

**In the painter**
Select the control and set the value in the Properties window, Definition category. For InkPicture controls, the table contains a large binary column to store ink overlay data and a large binary column to hold a background image for the InkPicture control. For TableBlob controls, the table contains the large binary database object you want to insert into the DataWindow.

**Examples**

[Visual Basic]

```VisualBasic
setting = dw1.Describe("blob_1.Table")
dw1.Modify("blob_1.Table='emp_pictures'")
```
Table.property

Properties for the DataWindow’s DBMS connection.

You can also specify stored procedures for update activities. For information, see Table.sqlaction.property.

Applies to DataWindows

Syntax Describe and Modify argument:

"DataWindow.Table.property { = value }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>property</td>
<td>A property for the DataWindow’s DBMS connection. Properties and appropriate values are listed in the table below.</td>
</tr>
<tr>
<td>value</td>
<td>The value to be assigned to the property.</td>
</tr>
</tbody>
</table>

Property for Table | Value |
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>Arguments</td>
<td>(Read only) A string containing retrieval argument names and types for the DataWindow.</td>
</tr>
<tr>
<td>CrosstabData</td>
<td>A string containing a tab-separated list of the expressions used to calculate the values of columns in a crosstab DataWindow.</td>
</tr>
<tr>
<td>Data.Storage</td>
<td>A string indicating whether table data is to be kept in memory or offloaded to disk. Values are:</td>
</tr>
<tr>
<td></td>
<td>• Memory (Default) – Table data is to be kept in memory.</td>
</tr>
<tr>
<td></td>
<td>• Disk – Table data is to be offloaded to disk.</td>
</tr>
<tr>
<td>Delete.Argument</td>
<td>(Internal use only) A string containing arguments to pass to the delete method.</td>
</tr>
<tr>
<td>Delete.Method</td>
<td>(Internal use only) The name of the method.</td>
</tr>
<tr>
<td>Delete.Type</td>
<td>(Internal use only) Currently stored procedure is the only type implemented.</td>
</tr>
<tr>
<td>Filter</td>
<td><em>(exp)</em> A string containing the filter for the DataWindow. Filters are expressions that can evaluate to true or false. The Table.Filter property filters the data before it is retrieved. To filter data already in the DataWindow’s buffers, use the Filter property or the SetFilter and Filter methods.</td>
</tr>
<tr>
<td></td>
<td>The filter string can be a quoted DataWindow expression.</td>
</tr>
<tr>
<td></td>
<td>Painter: Rows&gt;Filter.</td>
</tr>
<tr>
<td>GridColumns</td>
<td>(Read-only) The grid columns of a DataWindow.</td>
</tr>
<tr>
<td>Insert.Argument</td>
<td>(Internal use only) A string containing arguments to pass to the insert method.</td>
</tr>
<tr>
<td>Insert.Method</td>
<td>(Internal use only) The name of the method.</td>
</tr>
<tr>
<td>Insert.Type</td>
<td>(Internal use only) Currently stored procedure is the only type implemented.</td>
</tr>
</tbody>
</table>
### CHAPTER 3  DataWindow Object Properties

<table>
<thead>
<tr>
<th>Property for Table</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Procedure</td>
<td>A string that contains the number of the result set returned by the stored procedure to populate the DataWindow object. You can use this property only if your DBMS supports stored procedures. Use this property to change the stored procedure or to change the data source from a SELECT statement or script to a stored procedure (see the example). Painter: Set when Stored Procedure is selected as a data source.</td>
</tr>
</tbody>
</table>
| Select             | A string containing the SQL SELECT statement that is the data source for the DataWindow. Use this property to specify a new SELECT statement or change the data source from a stored procedure or Script to a SELECT statement. Table.Select has several advantages over the SetSqlSelect method:  
  - It is faster. PowerBuilder does not validate the statement until retrieval.  
  - You can change data source for the DataWindow. For example, you can change from a SELECT to a Stored Procedure.  
  - You can use none or any of the arguments defined for the DataWindow object in the SELECT. You cannot use arguments that were not previously defined for the DataWindow object.  
  - Describe always tries to return a SQL SELECT statement. If the database is not connected and the property’s value is a PBSELECT statement, Describe will convert it to a SQL SELECT statement if a SetTransaction method has been executed. If you are using describeless retrieval (the StaticBind database parameter is set to 1), you cannot use the Select property. Painter: Set when Select or Quick Select is selected as a data source. |
| Select.Attribute    | (Read-only) A string containing the PBSELECT statement for the DataWindow. |
| Sort               | (exp) A string containing the sort criteria for the DataWindow, for example, “1A,2D” (column 1 ascending, column 2 descending). The Table.Sort property sorts the data before it is retrieved. To sort data already in the DataWindow’s buffers, use the SetSort and Sort methods. The value for Sort is quoted and can be a DataWindow expression. Painter: Rows>Sort. |
| SQLSelect          | The most recently executed SELECT statement. Setting this has no effect. See Select in this table. |
| Update.Argument    | (Internal use only) A string containing arguments to pass to the update method. |
| Update.Method      | (Internal use only) The name of the method. |
| Update.Type        | (Internal use only) Currently stored procedure is the only type implemented. |
### Table.property

<table>
<thead>
<tr>
<th>Property for Table</th>
<th>Value</th>
</tr>
</thead>
</table>
| UpdateKey InPlace     | Whether the key column can be updated in place or the row has to be deleted and reinserted. This value determines the syntax the DataWindow server generates when a user modifies a key field:  
  - Yes – Use the **UPDATE** statement when the key is changed so that the key is updated in place.  
  - No – Use a **DELETE** and an **INSERT** statement when the key is changed.  |
| Caution               | **Caution**  
  When there are multiple rows in a DataWindow object and the user switches keys or rows, updating in place might fail due to DBMS duplicate restrictions.  
  Painter: Rows>Update Properties, Key Modification. |
| UpdateTable           | A string specifying the name of the database table used to build the Update syntax.  
  Painter: Rows>Update Properties, Table to Update. |
| UpdateWhere           | An integer indicating which columns will be included in the **WHERE** clause of the **Update** statement. The value of UpdateWhere can impact performance or cause lost data when more than one user accesses the same tables at the same time.  
  Values are:  
  - 0 – Key columns only (risk of overwriting another user’s changes, but fast).  
  - 1 – Key columns and all updatable columns (risk of preventing valid updates; slow because **SELECT** statement is longer).  
  - 2 – Key and modified columns (allows more valid updates than 1 and is faster, but not as fast as 0).  
  For more about the effects of this setting, see the discussion of the Specify Update Characteristics dialog box in the *User’s Guide*.  

**Examples**

```vbnet
[Visual Basic]
setting = dw1.Describe("DataWindow.Table.Sort")
dw1.Modify("DataWindow.Table.Filter='salary>50000'")
dw1.Modify("DataWindow.Table.Procedure= 
  '1 Execute MyOwner MyProcName;1 
  @NameOfProcArg=:NameOfDWArg,  
  @NameOfProcArg=:NameOfDWArg...' ")
sqlvar = 'SELECT ... WHERE ...
```

```vbnet
dw1.Modify("DataWindow.Table.Select='' + sqlvar + ''")
```
**Table.sqlaction.property**

**Description**
The way data is updated in the database. When the Update method is executed, it can send UPDATE, INSERT, and DELETE SQL statements to the DBMS. You can specify that a stored procedure be used instead of the default SQL statement for each type of data modification.

**Applies to**
DataWindows

**Syntax**
Describe and Modify argument:

"DataWindow.Table.sqlaction.property \{ = value \}"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>sqlaction</td>
<td>The SQL statement that would ordinarily be executed as part of a database update. Values are: • UPDATE • INSERT • DELETE</td>
</tr>
<tr>
<td>property</td>
<td>A property for sqlaction. Properties and appropriate values are listed in the table below.</td>
</tr>
<tr>
<td>value</td>
<td>The value to be assigned to the property.</td>
</tr>
</tbody>
</table>

**Property for Table**

<table>
<thead>
<tr>
<th>Value</th>
<th></th>
</tr>
</thead>
</table>
| Arguments      | A string specifying the arguments used in the stored procedure. The string takes this format:  
  \( \text{"argname"}, \text{valuetype} \{ =\text{"valuesrc"}, \text{datasrc}, \text{paramtype} \} \)  
  \text{Argname} is the name of the stored procedure parameter.  
  \text{Valuetype} is one of the keywords described below. \text{Datasrc} and \text{paramtype} apply to the COLUMN keyword.  
  \text{Valuesrc} is the column, computed field, or expression that produces the value to be passed to the stored procedure. |
| Method         | A string specifying the name of the stored procedure. The stored procedure is used only if the value of Type is SP. |
| Type           | Specifies whether the database update is performed using a stored procedure. Values are: • SP – The update is performed using a stored procedure. • SQL – The update is performed using standard SQL syntax (default). |
In the painter Set the values using Rows>Stored Procedure Update. Select the tab page for the SQL command you want to associate with a stored procedure.

In code If you enable a DataWindow object to use stored procedures to update the database when it is not already using stored procedures, you must change Type to SP first. Setting Type ensures that internal structures are built before you set Method and Arguments. If you do not change Type to SP, then setting Method or Arguments will fail.

<table>
<thead>
<tr>
<th>Keyword for valuetype</th>
<th>Description</th>
</tr>
</thead>
</table>
|COLUMN               | The argument value will be taken from the table and column named in valuesrc. Valuesrc has the form:
   "tablename.column"
   For COLUMN, you must also specify whether the data is the new or original column value. Values for datasrc are:
   - NEW The new column value that is being sent to the database.
   - ORIG The value that the DataWindow originally read from the database.
   You can also specify the type of stored procedure parameter. Values for paramtype are:
   - IN (Default) An input parameter for the procedure.
   - OUT An output parameter for the procedure. The DataWindow will assign the resulting value to the current row and column (usually used for identity and timestamp columns).
   - INOUT An input and output parameter.
   A sample string for providing a column argument is:
   
   (*empid*, COLUMN=("employee.empid", ORIG, IN))

|COMPUTE             | The computed field named in valuesrc is the source of the value passed to the stored procedure. A sample string for providing a computed field argument is:
   
   (*newsalary*, COMPUTE="salary_calc")

|EXPRESSION          | The expression specified in valuesrc is evaluated and passed to the stored procedure. A sample string for providing an expression argument is:
   
   (*dept_name*,
   EXPRESSION="LookUpDisplay(dept_id)")

|UNUSED              | No value is passed to the stored procedure.
When the values you specify in code are nested in a longer string, you must use
the appropriate escape characters for quotation marks.

**Examples**

Each is all on one line:

```vbnet
[Visual Basic]
dw_x.Describe("DataWindow.Table.Delete.Method")
dw_x.Describe("DataWindow.Table.Delete.Arguments")
dw_x.Modify("DataWindow.Table.Delete.Type=SP")
dw_x.Modify("DataWindow.Table.Delete.Arguments=  
    ((~"id~", COLUMN=(~"department.dept_id!~", 
    ORIG)))")

dw_x.Modify("DataWindow.Table.Delete.Method=  
    "spname~")
```

---

**TabSequence**

**Description**
The number assigned to the specified control in the DataWindow’s tab order.

**Applies to**
Column controls

**Syntax**
Describe and Modify argument:

```
"columnname.TabSequence { = number }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columnname</td>
<td>The name of the column whose tab order you want to get or set.</td>
</tr>
<tr>
<td>number</td>
<td>A number from 0 to 32000 specifying the position of the column in the tab order. A value of 0 takes the column out of the tab order and makes it read-only.</td>
</tr>
</tbody>
</table>

**Usage**

**In the painter**
Set the value using Format>Tab Order.

---

**Tab order changes have no effect in grid DataWindow objects**

In a grid DataWindow object, the tab sequence is always left to right (except on right-to-left operating systems). Changing the tab value to any number other than 0 has no effect.

**Examples**

```vbnet
[Visual Basic]
setting = dw1.Describe("emp_name.TabSequence")
dw1.Modify("emp_name.TabSequence = 10")
```
**Tag**

**Description**

The tag value of the specified control. The tag value can be any text you see fit to use in your application.

**Applies to**

Button, Column, Computed Field, Graph, GroupBox, Oval, Picture, Rectangle, Report, RoundRectangle, TableBlob, and Text controls

**Syntax**

Describe and Modify argument:

"controlname.Tag { = 'string' }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlname</td>
<td>The name of a control in the DataWindow.</td>
</tr>
<tr>
<td>string</td>
<td>(exp) A string specifying the tag for controlname.String is quoted and can be a DataWindow expression.</td>
</tr>
</tbody>
</table>

**Usage**

In the painter Select the control and set the value in the Properties window, General category.

**Examples**

[Visual Basic]

```vbnet
setting = dw1.Describe("blob_1.Tag")
dw1.Modify("graph_1.Tag = 'Graph of results'")
```

---

**Target**

**Description**

The columns and expressions whose data is transferred from the DataWindow to the OLE object.

**Applies to**

OLE Object controls

**Syntax**

Describe and Modify argument:

" oleobjectname.Target { = 'columnlist' }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>oleobjectname</td>
<td>The name of the OLE Object control for which you want to get or set the data to be transferred.</td>
</tr>
<tr>
<td>columnlist</td>
<td>(exp) A list of the columns or expressions whose data is transferred to the OLE object. If there is more than one, separate them with commas. Columnlist can be a quoted DataWindow expression.</td>
</tr>
</tbody>
</table>

**Usage**

GroupBy and Range also affect the data that is transferred to the OLE object.

In the painter Select the control and set the value in the Properties window, Data category.
Examples

[Visual Basic]
setting = dw1.Describe("ole_1.Target")
dw1.Modify("ole_1.Target = 'lname, Len(companyname)'")

**Template**

**Description**
The name of a file that will be used to start the application in OLE.

**Applies to**
TableBlob controls

**Syntax**
Describe and Modify argument:
"\[tblobname\].Template \{ = 'string' \}"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>tblobname</td>
<td>The name of a TableBlob control in the DataWindow.</td>
</tr>
<tr>
<td>string</td>
<td>( (exp) ) A string whose value is the file name of an application that</td>
</tr>
<tr>
<td></td>
<td>is to be the OLE template. ( String ) is quoted and can be a</td>
</tr>
<tr>
<td></td>
<td>DataWindow expression.</td>
</tr>
</tbody>
</table>

**Usage**
**In the painter** Select the control and set the value in the Properties window, Definition category.

**Examples**

[Visual Basic]
setting = dw1.Describe("blob_1.Template")
dw1.Modify("blob_1.Template='Excel.xls'")

**Text**

**Description**
The text of the specified control.

**Applies to**
Button, GroupBox, and Text controls

**Syntax**
Describe and Modify argument:
"\[textname\].Text \{ = 'string' \}"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>textname</td>
<td>The name of a control in the DataWindow.</td>
</tr>
<tr>
<td>string</td>
<td>( (exp) ) A string specifying the text for textname. To specify an</td>
</tr>
<tr>
<td></td>
<td>accelerator key in the text, include an ampersand before the desired</td>
</tr>
<tr>
<td></td>
<td>letter. The letter will display underlined. ( String ) is quoted and can</td>
</tr>
<tr>
<td></td>
<td>be a DataWindow expression.</td>
</tr>
</tbody>
</table>

**Usage**
**In the painter** Select the control and set the value in the Properties window, General category.
Timer_Interval

Examples

[Visual Basic]
setting = dw1.Describe("text_1.Text")
dw1.Modify("text_1.Text='Employee &Name'")

Timer_Interval

Description
The number of milliseconds between the internal timer events. When you use
time in a DataWindow, an internal timer event is triggered at the interval
specified by Timer_Interval. This determines how often time fields are
updated.

Applies to
DataWindows

Syntax

Describe and Modify argument:
"DataWindow.Timer_Interval { = number }"

DataWindowSyntaxFromSql:

DataWindow ( Timer_Interval = number )

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| number    | An integer specifying the interval between timer events in
milliseconds. The default is 60,000 milliseconds or one minute.
The maximum value is 65,535 milliseconds. |

Usage
When a computed field uses Now as its expression value, it refreshes the
displayed value every time the timer interval period elapses.

In the painter Select the DataWindow by deselecting all controls; then set
the value in the Properties window, General category.

Examples

[Visual Basic]
setting = dw1.Describe("DataWindow.Timer_Interval")
dw1.Modify("DataWindow.Timer_Interval=10000")
Title
Description
The title of the graph.
Applies to
Graph controls
Syntax
Describe and Modify argument:

"graphname.Title { = 'titlestring' }"

Usage
In the painter Select the control and set the value in the Properties window, General category.

The default expression for the Title.DispAttr.DisplayExpression property is "title", which refers to the value of the Title property. The display expression can combine the fixed text of the Title property with other text, functions, and operators. If the expression for Title.DispAttr.DisplayExpression does not include the Title property, then the value of the Title property will be ignored.

For an example, see DispAttr.fontproperty.

Examples

[Visual Basic]
setting = dwl.Describe("gr_1.Title")
dwl.Modify("gr_1.Title = 'Sales Graph'")

Title.DispAttr.fontproperty

See DispAttr.fontproperty.

Trail_Footer
Description
Whether the footer of a nested report is displayed at the end of the report or at the bottom of the page. Trail_Footer applies only to reports in a composite DataWindow. Setting Trail_Footer to No forces controls following the report onto a new page.
Applies to
Report controls
Syntax
Describe and Modify argument:

"reportname.Trail_Footer { = value }"
Trailer.#.property

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>reportname</td>
<td>The name of the report control for which you want to get or set Trail_Footer.</td>
</tr>
<tr>
<td>value</td>
<td>Whether the report’s footer trails the last line of the report or appears at the bottom of the page. Values are: Yes – The footer appears right after the last line of data in the report. No – The footer appears at the bottom of the page, forcing any data following the report onto the following page.</td>
</tr>
</tbody>
</table>

Examples

[Visual Basic]
setting = dw1.Describe("rpt_1.Trail_Footer")
dw1.Modify("rpt_1.Trail_Footer = Yes")

Trailer.#.property

See Bandname.property.

Tree.property

Description Settings for a TreeView DataWindow.
Applies to TreeView DataWindows
Syntax Describe and Modify argument:
  "DataWindow.Tree.property { = value } "
DataWindow .NET dot notation:
  dw_control.TreeViewProperties.property

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>property</td>
<td>A property that controls the appearance or behavior of the TreeView DataWindow. Properties and their settings are listed in the table below.</td>
</tr>
<tr>
<td>value</td>
<td>(exp) A string value for the file name of the tree node icon in the detail band. Value can be a quoted DataWindow expression.</td>
</tr>
</tbody>
</table>
### Property for Tree

<table>
<thead>
<tr>
<th>Property</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>DefaultExpandToLevel</td>
<td>A long value that is the default level of expansion for the TreeView DataWindow. For example, if the default level is 2, only data with a level less than or equal to 2 is expanded by default. The value must represent a valid level.</td>
</tr>
<tr>
<td>Indent</td>
<td>A long value in the units specified for the DataWindow that defines the position of the state icon. The state icon is a plus (+) or minus (-) sign that indicates whether the tree node is in a collapsed or expanded state. The icon’s indent indicates the level of the node in the tree. The X position of the state icon is the X position of its parent plus value.</td>
</tr>
</tbody>
</table>
| SelectNodeByMouse         | A boolean value that indicates whether you can select a tree node by clicking the node with the mouse.  
Values are:  
Yes – You can select a tree node with a mouse-click (default).  
No – You cannot select a tree node with a mouse-click. |
| ShowConnectLines          | A boolean value that indicates whether lines connecting parents and children display in the DataWindow object. This property is not supported by the Web DataWindow. If you want to show lines connecting rows in the detail band to their parent, you must also set ShowLeafNodeConnectLines.  
Values are:  
Yes – Display connecting lines (default).  
No – Do not display connecting lines. |
| ShowLeafNodeConnectLines  | A boolean value that indicates whether lines connecting rows in the detail band to their parent display in the DataWindow object. This property is disabled if Show Lines box is not set. This property is not supported by the Web DataWindow.  
Values are:  
Yes – Display connecting lines (default).  
No – Do not display connecting lines. |
| ShowTreeNodeIcon          | A boolean value that indicates whether tree node icons for level and detail bands display. If this property is not set, the Expanded and Collapsed Tree Node Icon Name properties in the General category for each TreeView level are disabled.  
Values are:  
No – Do not display tree node icons (default).  
Yes – Display tree node icons. |
| StateIconAlignMode        | A long value that indicates how the state icon is aligned vertically with respect to the TreeView level band.  
Values are:  
0 – Middle (default).  
1 – Top.  
2 – Bottom. |
Usage

In the painter  Select the control and set values in the Properties window, General category.

Examples

The following code gets and sets the Indent value:

```csharp
indentVal = dw1.Object.DataWindow.Tree.indent
dw1.Object.DataWindow.Tree.indent = 80
```

The following examples manipulate the SelectNodeByMouse property:

```csharp
if cbx_selectnodebymouse.checked then
  ls_selectnodebymouse='yes'
else
  ls_selectnodebymouse='no'
end if
ls_ret=dw1.modify("datawindow.tree.selectnodebymouse='","+ls_selectnodebymouselesen='+")
ls_selectnodebymouse=dw1.Describe("datawindow.tree.selectnodebymouse")
if lower(ls_selectnodebymouse)="no" then
  cbx_selectnodebymouse.checked=false
else
  cbx_selectnodebymouse.checked=true
end if

dw1.modify("datawindow.tree.selectnodebymouse='yes'")
dw1.Describe("datawindow.tree.selectnodebymouse")
```

The following examples manipulate the show connecting lines properties:

```csharp
bShowLines = dw1.TreeViewProperties.ShowConnectLines
bw.ShowConnectLines=true;
bShowLeafLines = dw1.TreeViewProperties.ShowConnectLines;
ShowLeafNodeConnectLines;
dw1.TreeViewProperties.ShowLeafNodeConnectLines=true;
```

The following example gets the current value of the StateIconAlignMode property and sets it to be aligned at the top:

```csharp
lAlign = dw1.TreeViewProperties.StateIconAlignMode;
//Align Top
dw1.TreeViewProperties.StateIconAlignMode = 1;
```
**Tree.LeafTREE НовоIconName**

**Description**
The file name of the tree node icon in the detail band.

**Applies to**
TreeView DataWindows

**Syntax**
Describe and Modify argument:

```
"DataWindow.Tree.Leaf.TreeNodeIconName { = value } "
```

DataWindow .NET dot notation:

```
dw_control.TreeViewProperties.TreeNodeIconName
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>(exp) A string value for the file name of the tree node icon in the detail band. Value can be a quoted DataWindow expression.</td>
</tr>
</tbody>
</table>

**Usage**

**In the painter**
Select the detail band by clicking the gray divider for the band. Specify a file name and location in the TreeLeafTreeNodeIconName property in the General category in the Properties window. This property is disabled if Use Tree Node Icon is not set in the General category in the Properties window for the DataWindow.

In DataView .NET, this property cannot be set in code using dot notation.

For the TreeView Web DataView, the image file must be deployed to the Web site.

**Tree.Level.#.property**

**Description**
The file name of the icon for a TreeView node in a TreeView level band when the icon is in either the expanded or collapsed state. You set the icon file name separately for each TreeView level band.

**Applies to**
TreeView DataWindows

**Syntax**
Describe and Modify argument:

```
"DataWindow.Tree.Level.#.property { = value } "
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>The number of the level for which you want to specify an icon. The level number must exist.</td>
</tr>
</tbody>
</table>
Tree.Level.#.property

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>property</td>
<td>A property that indicates whether the icon specified is for the expanded or collapsed state. Values are:</td>
</tr>
<tr>
<td></td>
<td>• CollapsedTreeNodeIconName</td>
</tr>
<tr>
<td></td>
<td>• ExpandedTreeNodeIconName</td>
</tr>
<tr>
<td>value</td>
<td>(exp) A string value that is the file name of the tree node icon in the selected TreeView level band. Value can be a quoted DataWindow expression.</td>
</tr>
</tbody>
</table>

Usage

**In the painter**  Select the level by clicking the gray divider for the band.
Specify a file name and location in the TreeLevelCollapsedTreeNodeIconName and TreeLevelExpandedTreeNodeIconName properties in the General category in the Properties window for the band. These properties are disabled if is not selected in the General category in the Properties window for the DataWindow.

You cannot set these properties using dot notation.

For a TreeView Web DataWindow, the image files must be deployed to the Web site.

Examples

The following example gets the name of the icon used when a level 1 node is collapsed:

```csharp
ls_ico = dw_tview.Describe &
("DataWindow.Tree.Level.1.CollapsedTreeNodeIconName")
```
CHAPTER 3  DataWindow Object Properties

Type
Description
The type of the control (for Describe) or the type of presentation style (for DataWindowSyntaxFromSql).

Syntax
Describe argument:
"controlname.Type"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| controlname     | The name of the control for which you want the type. Valid values are:  
datawindow  
bitmap (for Picture)  
button  
column  
compute (for Computed Field)  
graph  
groupbox  
line  
ellipse (for Oval)  
rectangle  
report  
roundrectangle  
tableblob  
text  

DataWindowSyntaxFromSql:
Style ( Type = value )

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| value     | A keyword specifying the presentation style for the DataWindow object. Keywords are:  
( Default) Tabular  
Grid  
Form (for the Freeform style)  
Crosstab  
Graph  
Group  
Label  
Nested  

Examples
[Visual Basic]
setting = dw1.Describe("emp_name.Type")

DWSyntaxGenerator.DataWindowSyntaxFromSql(myTrans, sqlstring,'Style(... Type=grid ...)')
**Units**

**Description**

The unit of measure used to specify measurements in the DataWindow object. You set this in the DataWindow Style dialog box when you define the DataWindow object.

**Applies to**

DataWindows

**Syntax**

Describe argument:

"DataWindow.Units"

DataWindowSyntaxFromSql:

```csharp
DataWindow ( Units = value )
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
</table>
| value     | The type of units for measurements in the DataWindow. Values are:
|           | 0 – Normalized units
|           | 1 – Display pixels
|           | 2 – 1/1000 of a logical inch
|           | 3 – 1/1000 of a logical centimeter |

**Usage**

Normalized units and display pixels are adjusted for printing.

**In the painter**  Select the DataWindow by deselecting all controls; then set the value in the Properties window, General category.

**Examples**

[Visual Basic]

```vbnet
setting = dw1.Describe("DataWindow.Units")
```

---

**Update**

**Description**

Whether the specified column is updatable. Each updatable column is included in the SQL statement that the Update method sends to the database. All updatable columns should be in the same database table.

**Applies to**

Column controls

**Syntax**

Describe and Modify argument:

"columnname.Update ( = value )"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columnname</td>
<td>The column for which you want to get or set the updatable status</td>
</tr>
</tbody>
</table>
CHAPTER 3  DataWindow Object Properties

### Usage

**In the painter** Set the value using Rows>Update Properties, Updateable Columns option.

**Examples**

[Visual Basic]
```vbnet
setting = dw1.Describe("emp_name.Update")
dw1.Modify("emp_name.Update=No")
```

### Validation

**Description**

The validation expression for the specified column. Validation expressions are expressions that evaluate to true or false. They provide checking of data that the user enters in the DataWindow.

To set the validation expression, you can also use the SetValidate method. To check the current validation expression, use the GetValidate method.

**Applies to** Column controls

**Syntax** Describe and Modify argument:

```text
"columnname.Validation { = 'validationstring' }
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>columnname</td>
<td>The column for which you want to get or set the validation rule.</td>
</tr>
<tr>
<td>validationstring</td>
<td>(exp) A string containing the rule that will be used to validate data entered in the column. Validation rules are expressions that evaluate to true or false. Validationstring is quoted and can be a DataWindow expression.</td>
</tr>
</tbody>
</table>

**Usage**

**In the painter** Set the value using Column Specifications view, Validation Expression option.

Use operators, functions, and columns to build an expression. Use Verify to test it.

**Examples**

[Visual Basic]
```vbnet
setting = dw1.Describe("emp_status.Validation")
```
ValidationMsg

Description
The message that the DataWindow server displays instead of the default message when an ItemError event occurs in the column.

Applies to
Column controls

Syntax
Describe and Modify argument:
"columnname.ValiationMsg { = 'string' }"

Parameter | Description
--- | ---
columnname | The column for which you want to get or set the error message displayed when validation fails.
string | (exp) A string specifying the error message you want to set. String is quoted and can be a DataWindow expression.

Usage
In the painter Set the value using the Column Specifications view, Validation Message option.

Examples

[Visual Basic]
setting = dw1.Describe("emp_salary.ValidationMsg")
dw1.Modify("emp_salary.ValidationMsg = " _ "'Salary must be between 10,000 and 100,000'")

Values (for columns)

Description
The values in the code table for the column.

Applies to
Column controls

Syntax
Describe and Modify argument:
"columnname.Values { = 'string' }"

Parameter | Description
--- | ---
columnname | The column for which you want to specify the contents of the code table.
string | (exp) A string containing the code table values for the column. In the string, separate the display values and the actual values with a tab character, and separate multiple pairs of values with a slash using this format:
"displayval-tabactualval/displayval-tabactualval/ ..."
For example:
"red-t1/white-t2"
String is quoted and can be a DataWindow expression.
Usage

In the painter  Select the control and set the value in the Properties window, Behavior category.

When EditStyle is DropDownListBox, click the Values browse button to fill in the Display Value and Data Value columns for the code table.

When Style is Edit or EditMask, set the CodeTable property and click the Values browse button to fill in the Display Value and Data Value columns for the code table.

Examples

[Visual Basic]

```vbnet
setting = dw1.Describe("emp_status.Values")
dw1.Modify("emp_status.Values = 'Active-tA/Part Time-tP/Terminated-tT'")
```

Values (for graphs)

See Axis, Axis.property, and DispAttr.fontproperty.

Vertical_Size

Description

The height of the columns in the detail area of the DataWindow object.

Vertical_Size is meaningful only when Type is Form (meaning the Freeform style). When a column reaches the specified height, the DataWindow server starts a new column to the right of the current column. The space between columns is specified in the Vertical_Spread property.

Applies to

Style keywords

Syntax

DataWindowSyntaxFromSql:

```
Style ( Vertical_Size = value )
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>An integer specifying the height of the columns in the detail area of the DataWindow object area in the units specified for the DataWindow</td>
</tr>
</tbody>
</table>

Examples

[Visual Basic]

```
DWSyntaxGenerator.DataWindowSyntaxFromSql(myTrans, sqlstring, 'Style(... Vertical_Size=1225...)')
```
Vertical_Spread

Description
The vertical space between columns in the detail area of the DataWindow object. Vertical_Spread is meaningful only when Type is Form (meaning the Freeform style). The Vertical_Size property determines when to start a new column.

Applies to
Style keywords

Syntax
DataWindowSyntaxFromSql:

Style ( Vertical_Spread = value )

Parameter | Description
---|---
value | An integer specifying the vertical space between columns in the detail area of the DataWindow object area in the units specified for the DataWindow

Examples
[Visual Basic]
DWSyntaxGenerator.DataWindowSyntaxFromSql(myTrans, sqlstring,'Style(... Vertical_Spread=25...)')

VerticalScrollMaximum

Description
The maximum height of the scroll box of the DataWindow’s vertical scroll bar. This value is set by the DataWindow server based on the content of the DataWindow. Use VerticalScrollMaximum with VerticalScrollPosition to synchronize vertical scrolling in multiple DataWindow objects. The value is a long.

Applies to
DataWindows

Syntax
Describe argument:

"DataWindow.VerticalScrollMaximum"

Examples
[Visual Basic]
setting = dw1.Describe("DataWindow.VerticalScrollMaximum")
CHAPTER 3  DataWindow Object Properties

**VerticalScrollPosition**

**Description**
The position of the scroll box in the vertical scroll bar. Use VerticalScrollMaximum with VerticalScrollPosition to synchronize vertical scrolling in multiple DataWindow objects.

** Applies to**
DataWindows

**Syntax**
Describe and Modify argument:

```
"DataWindow.VerticalScrollPosition { = scrollvalue }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>scrollvalue</td>
<td>A long specifying the position of the scroll box in the vertical scroll bar of the DataWindow</td>
</tr>
</tbody>
</table>

**Visible**

**Description**
Whether the specified control in the DataWindow is visible.

** Applies to**
Button, Column, Computed Field, Graph, GroupBox, Line, OLE, Oval, Picture, Rectangle, Report, RoundRectangle, TableBlob, and Text controls

**Syntax**
Describe and Modify argument:

```
"controlname.Visible { = 'value' }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>controlname</td>
<td>The name of the control for which you want to get or set the Visible property.</td>
</tr>
</tbody>
</table>
| value | *(exp)* Whether the specified control is visible. Values are:

<table>
<thead>
<tr>
<th>Value</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>False; the control is not visible.</td>
</tr>
<tr>
<td>1</td>
<td>True; the control is visible.</td>
</tr>
</tbody>
</table>

Value can be a quoted DataWindow expression.

**Usage**

**In the painter**  Select the control and set the value in the Properties window, Visible category. The Visible property is not supported for column controls in DataWindow objects with the Label presentation style.

**Examples**

[Visual Basic]

```vbnet
setting = dw1.Describe("emp_status.Visible")
dw1.Modify("emp_status.Visible=0")
dw1.Modify("emp_stat.Visible='0~tIf(emp_cls=1,0,1)'")
```
**VTextAlign**

Description: The way text in a button is vertically aligned.

Applies to: Button controls

Syntax: Describe and Modify argument:

```
"buttonname.VTextAlign { = 'value' }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>buttonname</code></td>
<td>The name of the button for which you want to align text.</td>
</tr>
<tr>
<td><code>value</code></td>
<td>An integer indicating how the button text is horizontally aligned. Values are: 0 – Center, 1 – Top, 2 – Bottom, 3 – Multiline</td>
</tr>
</tbody>
</table>

Usage: **In the painter** Select the control and set the value in the Properties window, Appearance category.

Examples:

```Visual Basic
setting = dw1.Describe("b_name.VTextAlign")
dw1.Modify("b_name.VTextAlign = '0'")
```

**Width**

Description: The width of the specified control.

Applies to: Button, Column, Computed Field, Graph, GroupBox, OLE, Oval, Picture, Rectangle, Report, RoundRectangle, TableBlob, and Text controls

Syntax: Describe and Modify argument:

```
"controlname.Width { = 'value' }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>controlname</code></td>
<td>The name of the control for which you want to get or set the width.</td>
</tr>
<tr>
<td><code>value</code></td>
<td><code>(exp)</code> The width of the <code>controlname</code> in the units specified for the DataWindow. Value can be a quoted DataWindow expression.</td>
</tr>
</tbody>
</table>

Usage: **In the painter** Select the control and set the value in the Properties window, Layout category.
CHAPTER 3  DataWindow Object Properties

Examples

[Visual Basic]
setting = dw1.Describe("emp_name.Width")
dw1.Modify("emp_name.Width=250")

X

Description
The distance of the specified control from the left edge of the DataWindow object.

Applies to
Button, Column, Computed Field, Graph, GroupBox, OLE, Oval, Picture, Rectangle, Report, RoundRectangle, TableBlob, and Text controls

Syntax
Describe and Modify argument:
"controlname.X \{ = 'value' \}"

Parameter | Description
--- | ---
controlname | The name of the control for which you want to get or set the x coordinate.
value | (exp) An integer specifying the x coordinate of the control in the unit of measure specified for the DataWindow object. Value can be a quoted DataWindow expression.

Usage
In the painter  Select the control and set the value in the Properties window, Layout category.

Examples

[Visual Basic]
setting = dw1.Describe("emp_name.X")
dw1.Modify("emp_name.X=10")

X1, X2

Description
The distance of each end of the specified line from the left edge of the line’s band.

Applies to
Line controls

Syntax
Describe and Modify argument:
"controlname.X1 \{ = 'value' \}"
"controlname.X2 \{ = 'value' \}"

DataWindow Object Reference 357
**XHTMLGen.Browser**

**Parameter** | **Description**
--- | ---
`controlname` | The name of the line for which you want to get or set one of the x coordinates.

`value` | (exp) An integer specifying the x coordinate of the line in the unit of measure specified for the DataWindow object. *Value* can be a quoted DataWindow expression.

**Usage**

**In the painter** Select the control and set the value in the Properties window, Layout category.

**Examples**

[Visual Basic]

```vbscript
setting = dw1.Describe("line_1.X1")
dw1.Modify("line_1.X1=10")
dw1.Modify("line_1.X2=1000")
```

**XHTMLGen.Browser**

**Description**
A string that identifies the browser in which XHTML generated within an XSLT style sheet is displayed.

**Applies to**
DataWindow objects

**Syntax**
Describe and Modify argument:

```
"DataWindow.XHTMLGen.Browser { = 'value' }"
```

**Parameter** | **Description**
--- | ---
`value` | (exp) A string identifying the browser in which you want to display the generated XHTML. The value should match the browser identifier part of the text string that the browser specifies in the HTTP header it sends to the server. This property is usually set dynamically on the server according to the HTTP header returned from the client. Recognized strings are listed in the Usage section below.

**Usage**
If the string specifies a browser that the DataWindow engine supports, the DataWindow generates an XSLT style sheet and JavaScript for XHTML transformation optimized for that browser. Browser-specific XSLT and JavaScript are generated only for Microsoft Internet Explorer 5.0 and later and Netscape 6.0 and later.

Browser identification strings are sent by the client to the server in the HTTP header. The server component can assign the HTTP_USER_AGENT value from the HTTP header to the Browser property.
CHAPTER 3   DataWindow Object Properties

The XML Web DataWindow generator recognizes these browsers:

<table>
<thead>
<tr>
<th>Browser</th>
<th>HTTP header string</th>
</tr>
</thead>
<tbody>
<tr>
<td>Microsoft Internet Explorer</td>
<td>Mozilla/4.0 (compatible; MSIE 5.0;</td>
</tr>
<tr>
<td></td>
<td>Mozilla/4.0 (compatible; MSIE 5.5;</td>
</tr>
<tr>
<td></td>
<td>Mozilla/4.0 (compatible; MSIE 6.x;</td>
</tr>
<tr>
<td>Netscape</td>
<td>Mozilla/5.0(</td>
</tr>
</tbody>
</table>

In the painter  In the Web Generation category in the Properties window for the DataWindow object, select XHTML from the WebDW list and select a browser from the list.

**XMLGen.property**

**Description**

Settings that specify how XML is generated, whether client-side, postback, or callback paging is used, the physical path to which XML is published, and the URL referenced by the JavaScript that transforms the XML to XHTML.

**Applies to**

DataWindow objects

**Syntax**

Describe and Modify argument:

"DataWindow.XMLGen.property { = value }"

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>property</td>
<td>One of the following:</td>
</tr>
<tr>
<td></td>
<td>• Inline</td>
</tr>
<tr>
<td></td>
<td>• PublishPath</td>
</tr>
<tr>
<td></td>
<td>• ResourceBase</td>
</tr>
<tr>
<td>value</td>
<td><em>(exp)</em> Inline – A boolean that specifies whether the XML generated for the XML Web DataWindow is generated inline to the XSLT transformation script. Values are:</td>
</tr>
<tr>
<td></td>
<td>true – The XML is generated within the XSLT transformation script.</td>
</tr>
<tr>
<td></td>
<td>false – (default) The XML is published to a separate document.</td>
</tr>
<tr>
<td></td>
<td><em>(exp)</em> PublishPath – A string that specifies the physical path of the Web site folder to which the DataWindow server publishes the generated XML document that contains the XML Web DataWindow content.</td>
</tr>
<tr>
<td></td>
<td><em>(exp)</em> ResourceBase – A string that specifies the URL of the generated XML document that contains the XML Web DataWindow content.</td>
</tr>
</tbody>
</table>
**Usage**

**Inline**  The XML published on the Internet in your XML Web DataWindow could contain sensitive data, and this data might be exposed to Internet users when published to a separate document. For increased security, if the Inline property is set to true, the XML is generated “inline” to the XSLT transformation script in the page that renders the control. If only authenticated users have access to this script, the security of the XML is ensured. Setting this property should have no adverse side effects on the caching efficiency of the control.

**PublishPath and ResourceBase**  The PublishPath folder must correspond to the URL specified in the ResourceBase property. At runtime, after the DataWindow server generates XML content to the PublishPath folder, client-side JavaScript in a generated page downloads it using a reference to the ResourceBase property. The JavaScript transforms the XML content to XHTML using the generated XSLT style sheet.

**In the painter**  In the Web Generation category in the Properties window for the DataWindow object, select XML from the WebDW list and select the options you require.

**In DataWindow .NET**  In DataWindow .NET, you can specify the physical directory in which dynamically created files, such as .css, .js, .xml, and .xslt files, and URL references are stored, using the XmlConfigurations.UrlPath property. If you specify a value for XmlConfigurations.UrlPath, it overrides the values set for PublishPath and ResourceBase set in the DataWindow painter.

If you want to specify a full path or different paths for each of the different file types, set the properties in the DataWindow painter and leave the XmlConfigurations.UrlPath property empty in DataWindow .NET.

If you do not set these properties in the DataWindow painter or in DataWindow .NET, the files are saved in the current Web application’s path.

**Examples**

These statements set the XMLGen.ResourceBase and XMLGen.PublishPath properties:

```visualbasic
    dw1.Modify("DataWindow.XMLGen.PublishPath='C:\Inetpub\wwwroot\MyWebApp\generatedfiles'")
    dw1.Modify("DataWindow.XMLGen.ResourceBase='\MyWebApp\generatedfiles'")
```

This statement sets the XMLGen.Inline property so that XML is generated inline:

```visualbasic
    dw1.Modify("DataWindow.XMLGen.Inline='1'")
```
**XSLTGen.property**

**Description**
Settings that specify the physical path to which the generated XSLT style sheet is published and the URL referenced by the JavaScript that transforms the XML to XHTML.

**Applies to**
DataWindow objects

**Syntax**
Describe and Modify argument:

```
"DataWindow.XSLTGen.property { = 'value' }
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>property</td>
<td>One of the following:</td>
</tr>
<tr>
<td></td>
<td>• PublishPath</td>
</tr>
<tr>
<td></td>
<td>• ResourceBase</td>
</tr>
<tr>
<td>value</td>
<td>(exp) PublishPath – A string that specifies the physical path of the Web site folder to which the DataWindow server publishes the generated XSLT style sheet</td>
</tr>
<tr>
<td></td>
<td>(exp) ResourceBase – A string that specifies the URL of the generated XSLT style sheet</td>
</tr>
</tbody>
</table>

**Usage**
The PublishPath folder must correspond to the URL specified in the ResourceBase property. At runtime, after the DataWindow server generates the XSLT style sheet to the PublishPath folder, client-side JavaScript in a generated page downloads it using a reference to the ResourceBase property. The JavaScript transforms the XML content to XHTML using the generated XSLT style sheet.

**In the painter**
In the Web Generation category in the Properties window for the DataWindow object, select XSLT from the WebDW list and specify the ResourceBase and Publish Path locations.

**In DataWindow .NET**
In DataWindow .NET, you can specify the physical directory in which dynamically created files, such as .css, .js, .xml, and .xslt files, and URL references are stored, using the XmlConfigurations.UrlPath property. If you specify a value for XmlConfigurations.UrlPath, it overrides the values set for PublishPath and ResourceBase set in the DataWindow painter.

If you want to specify a full path or different paths for each of the different file types, set the properties in the DataWindow painter and leave the XmlConfigurations.UrlPath property empty in DataWindow .NET.

If you do not set these properties in the DataWindow painter or in DataWindow .NET, the files are saved in the current Web application’s path.

**Examples**
These statements set the XSLTGen.ResourceBase and XSLTGen.PublishPath properties:
Y

Description
The distance of the specified control from the top of the control’s band.

Applies to
Button, Column, Computed Field, Graph, GroupBox, OLE, Oval, Picture, Rectangle, Report, RoundRectangle, TableBlob, and Text controls

Syntax
Describe and Modify argument:

"controlname.Y { = 'value' }"

Usage
In the painter
Select the control and set the value in the Properties window, Layout category.

Examples

[Visual Basic]
setting = dw1.Describe("emp_name.Y")
dw1.Modify("emp_name.Y=100")

Y1, Y2

Description
The distance of each end of the specified line from the top of the line’s band.

Applies to
Line controls

Syntax
Describe and Modify argument:

"controlname.Y1 { = 'value' }"
"controlname.Y2 { = 'value' }"

Parameter Description

<table>
<thead>
<tr>
<th>controlname</th>
<th>The name of the line for which you want to get or set one of the y coordinates.</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>(exp) An integer specifying the y coordinate of the control in the unit of measure specified for the DataWindow object. Value can be a quoted DataWindow expression.</td>
</tr>
</tbody>
</table>
**CHAPTER 3   DataWindow Object Properties**

**DataWindow Object Reference**

### Usage

**In the painter**  Select the control and set the value in the Properties window, Layout category.

**Examples**

[Visual Basic]

```vb
setting = dw1.Describe("line_1.Y1")
dw1.Modify("line_1.Y1=50")
dw1.Modify("line_1.Y2=50")
```

### Zoom

**Description**  The scaling percentage of the DataWindow object.

**Applies to**  DataWindows

**Syntax**  Describe and Modify argument:

```
"DataWindow.Zoom { = value }"
```

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value</td>
<td>An integer specifying the scaling percentage of the DataWindow object. The default is 100%.</td>
</tr>
</tbody>
</table>

**Usage**  In the painter  To see the effect of different zoom factors in Preview mode, use Design>Zoom. The zoom factor you set in the painter is not used at runtime.

**Limitation**  The zoom property is not supported for the Graph DataWindow style.

**Examples**

[Visual Basic]

```vb
setting = dw1.Describe("DataWindow.Zoom")
dw1.Modify("DataWindow.Zoom=50")
```

---

Parameter | Description
---|---
value | *(exp)* An integer specifying the y coordinate of the line in the unit of measure specified for the DataWindow object. *Value* can be a quoted DataWindow expression.

**Value** *(exp)* An integer specifying the y coordinate of the line in the unit of measure specified for the DataWindow object. *Value* can be a quoted DataWindow expression.
Zoom
Index

Symbols
= (relational) 6

Numerics
3D (Checkbox.property) 192
3D (RadioButtons.property) 314

A
Abs function 24
absolute value 24
Accelerator property 168
AccessibleDescription property 169
AccessibleName property 169
AccessibleRole property 170
ACos function 24
Action property 171
Activation property 173
addition operator 5
aggregate functions
  Avg 27
  Count 34
  CrosstabMax 43
  CrosstabMaxDec 44
  CrosstabMin 45
  CrosstabMinDec 47
  CrosstabSum 48
  CrosstabSumDec 50
  CumulativePercent 51
  CumulativeSum 53
  First 63
  Large 74
  Last 76
  Max 87
  Median 89
  Min 92
  Mode 95
  Percent 103
  restrictions 14, 16
  Small 120
  StDev 124
  StDevP 126
  Sum 131
  Var 136
  VarP 139
Alignment property 174
AllowEdit (dddw.property) 212
AllowEdit (ddlby.property) 216
AND operator 9
angle
calculating arc cosine 24
calculating arc sine 26
calculating arc tangent 27
calculating cosine 34
calculating sine 120
calculating tangent 133
AntiAliased (Ink.property) 278
AppendedHTML (HTML.property) 263
appending a string 112, 113
arc cosine 24
arc sine 26
arc tangent 27
Arguments (Table.property) 334
Arguments (Table.sqlaction.property) 337
Arguments property 175
arithmetic operators 5
Asc function 25
AscA function 25
ASCII values, converting characters to 25
ASin function 26
asterisks (*), in text patterns 85
ATan function 27
Attributes property 175
AutoErase (InkPic.property) 282
AutoHScroll (dddw.property) 212
AutoHScroll (ddlby.property) 216
AutoHScroll (Edit.property) 224
Index

AutoHScroll (InkEdit.property) 279
AutoRetrieve (dddw.property) 212
AutoScale (Axis.property) 177
AutoSelect (Edit.property) 224
AutoSelect (InkEdit.property) 279
AutoSize Height property for bands 186
AutoSkip (EditMask.property) 228
AutoVScroll (Edit.property) 224
AutoVScroll (InkEdit.property) 279
average value
  columns 27
  crosstabs 36, 40
Avg function 27
Axis properties 177
Axis property 176

B
BackColor (InkPic.property) 282
BackColor property 180
Background properties 181
BackColor property 183
backslash character, in text patterns 85
Band property 184
Bandname properties 185
Bandname.Text property (RichText only) 188
Bands property 188
BETWEEN operator 6, 7
BinaryIndex property 189
Bitmap function 30
BitmapName property 189
Border (HTMLTable.property) 273
Border property (DataWindow object), about 189
brackets in text patterns 85
Browser (HTMLGen.property) 266
Brush properties 191
Buttons (Print.Preview.property) 302
Buttons (Print.property) 304

C
CanUseDefaultPrinter (Print.property) 304
capitalization
  first letter 141
  lowercase 84
  uppercase 136
caret in text patterns 85
Case (dddw.property) 212
Case (ddlb.property) 216
Case (Edit.property) 224
Case function 31
Category property. See Axis properties
Ceiling function 32
CellPadding (HTMLTable.property) 273
CellSpacing (HTMLTable.property) 273
century 142
Char function 33
CharA function 33
characters
  case of 25
  changing capitalization 84, 136, 141
  converting to ASCII values 25
  extracting 91, 92
  matching 84
  returning leftmost 78, 79
  returning rightmost 115
CheckBox property 192
ClientComputedFields (HTMLGen.property) 266
ClientEvents (HTMLGen.property) 266
ClientFormatting (HTMLGen.property) 266
ClientName property 194
ClientScriptable (HTMLGen.property) 266
ClientValidation (HTMLGen.property) 266
ClipText (Print.property) 304
CodeTable (Edit.property) 224
CodeTable (EditMask.property) 228
CollapseTreeNodeIconName (Tree.Level property) 347
Collate (Print.property) 304
CollectionMode (InkPic.property) 282
Color (Background.property) 181
Color (Bandname.property) 185
Color (Brush.property) 191
Color (Ink.property) 278
Color (Pen.property) 299
Color (Print.property) 304
Color property 195
colors
  red, green, and blue components of 113
  table of standard colors 114
ColType property 196

366
Column.Count property 197
columns
  average value 27
  checking for null value 70
  counting null values, example 15
  cumulative percent 51
  cumulative sum 53
  display value 83
  first value 63
  large value 74
  last value 76
  maximum value 87
  median value 89
  minimum value 92
  most frequently occurring value 95
  number of rows 34
  percent of range 103
  small value 120
  standard deviation 124, 126
  total of values 131
  total of values, example 15, 17
  value in code table 83
  variance 136, 139
Columns (Crosstab.property) 200
Columns (Print.property) 304
Columns (RadioButtons.property) 314
Columns.Width (Print.property) 304
CommonJSFile (HTMLGen.property) 266
comparing strings 8
computed fields, expressions 13
concatenation operator 10
conditional expressions
  with Evaluate 12
  conditional expressions, IF function 67
configuration settings, reading 107, 109
ContentsAllowed property 198
Copies (Print.property) 304
Cos function 34
cosine 34
Count function 34
count of values
  columns 34
crosstabs 41
  example 14
Criteria properties 199
Criteria property 198
Crosstab properties 200
CrosstabAvg function 36
CrosstabAvgDec function 40
CrosstabCount function 41
CrosstabData (Table.property) 334
CrosstabMax function 43
CrosstabMaxDec function 44
CrosstabMin function 45
CrosstabMinDec function 47
CrosstabSum function 48
CrosstabSumDec function 50
CSS generation properties 202
CSSGen.PublishPath 202
CSSGen.ResourceBase 202
CSSGen.SessionSpecific 202
CumulativePercent function 51
CumulativeSum function 53
currency, and rows 65
CustomPage.Length (Print.property) 304
CustomPage.Width (Print.property) 304
D
data
  converting to type long 82
  counting nulls 15
Data property 203
datatype checking and conversion functions
  Asc 25
  AscA 25
  Char 33
  CharA 33
  Date 55
  DateTime 56
  Dec 59
  Integer 68
  IsDate 69
  IsNull 70
  IsNumber 71
  IsTime 73
  Long 82
  Number 99
  Real 110
  String 129
  Time 133

Index
Index

datatatypes
  real  110
  string  129
  time  133
Data.HTML property  204
Data.HTMLTable property  205
Data.Storage (Table.property)  334
Data.XHTML property  206
Data.XML property  207
Data.XMLDTD property  208
Data/XMLSchema property  208
Data/XMLSchema property  208
Data.XMLOWeb property  208
Data.XSLFO property  209
Database painter, validation rules  3
DataColumn (dddw.property)  212
DataObject property  209
DataWindow expression functions
  Describe in painter expressions  60
  Fill in painter expressions  63
  Left in painter expressions  79
  Len in painter expressions  81
  Mid in painter expressions  92
  Pi in painter expressions  105
  Pos in painter expressions  107
  Replace in painter expressions  113
  Right in painter expressions  115
DataWindow object properties  145
  for controls in a DataWindow  145
overview  143
DataWindow object properties, table  145
date columns, and different DBMSs  197
Date function  55
date, day, and time functions
  Day  57
  DayName  57
  DayNumber  58
  DaysAfter  59
  Hour  66
  Minute  94
  Month  98
  Now  98
  RelativeDate  111
  RelativeTime  111
  Second  118
  SecondsAfter  119
  Today  134
  Year  142
  DateJSFile (HTMLGen.property)  266
dates
  checking string  69
  converting to  55
  DateTime data type  56
  day of week  57, 58
  determining interval  59
  obtaining current  134
  obtaining day of month  57
  DateTime function  56
  Day function  57
  DayName function  57
  DayNumber function  58
  DaysAfter function  59
  dbAlias property  210
  dbName property  211
DDCal_AlignRight (EditMask.property)  228
DDCal_BackColor (EditMask.property)  228
DDCal_TextColor (EditMask.property)  228
DDCal_TitleBackColor (EditMask.property)  228
DDCal_TitleTextColor (EditMask.property)  228
DDCal_TrailingTextColor (EditMask.property)  228
DDCalendar (EditMask.property)  228
ddlw properties  212
ddlb properties  216
Dec function  59
decimal, converting to  59
DefaultExpandToLevel (Tree.property)  344
DefaultPicture property  218
Delete (Table.property)  334
Depth property  219
Describe function
  evaluating expressions  11
Describe function, in DataWindow expressions  60
Detail properties.  See Bandname properties
Detail_Bottom_Margin property  220
Detail_Top_Margin property  220
Dialog (Criteria.property)  199
DispAttr (Axis.property)  177
DispAttr font properties  221
display formats
  applying to strings  129
DisplayColumn (dddw.property)  212
displayed value from code table  83
DisplayEveryNLabels (Axis.property)  177
DisplayOnly (Edit.property) 224
DisplayOnly (InkEdit.property) 279
DisplayType property 223
division 95
division operator 5
DocumentName (Print.property) 304
dollar sign in text patterns 85
drawing controls, setting color of 114
DropLines (Axis.property) 177
Duplex (Print.property) 304
DynamicRendering (InkPic.property) 282

E
Edit properties 224
EditMask properties 228
EditMode (InkPic.property) 282
Elevation property 232
EllipseHeight property 233
EllipseWidth property 234
Enabled property 234
EncodeSelfLinkArgs (HTMLGen.property) 266
EraserMode (InkPic.property) 282
EraserWidth (InkPic.property) 282
escape keyword 7
Evaluate function 11
Exp function 61
ExpandTreeNodeIconName (Tree.Level property) 347
exponent 61
exponentiation operator 5
Export.PDF.Distill.CustomPostScript property 235
Export.PDF.XSLFOP.Print property 236
Export.XHTML.UseTemplate property 239
Export.XML.IncludeWhitespace property 240
Export.XML.IncludeWhitespace property 240
Export.XML.SaveMetaDataType property 236, 241
Export.XML.SaveMetaDataType property 242
Export.XML.TemplateCount property 237, 238, 244
Export.XML.UseTemplate property 245
Expression property 246
expressions
  checking for null 70
  conditional evaluation 67
  conditional for DataWindow properties 12

F
Fact function 61
Factoid (InkEdit.property) 279
Factoid property 281
Filename (Print.property) 304
Fill function 62
FillA function 63
Filter (Table.property) 334
filters
  functions in expressions for 13
First function 63
FirstRowOnPage property 247
FocusRectangle (Edit.property) 224
FocusRectangle (EditMask.property) 228
FocusRectangle (InkEdit.property) 279
Font properties 248
Font.Bias property 248
Footer properties. See Bandname properties
Format (Edit.property) 224
Format property 250
Frame (Axis.property) 177
functions
  aggregate 14, 16
  example, counting data 16
  example, counting NULLs 14
  example, displaying data 21
  example, row indicator 19

G
Generate Securely Inline (XMLGen.property) 359
GenerateCSS (HTMLTable.property) 273
GenerateDDDWFrames (HTMLGen.property) 266
GenerateJavaScript (HTMLGen.property) 266
GetRow function 65
GraphType property 251
greater than operator 6
greater than or equal to operator 6
Grid.ColumnMove property 252
Grid.Lines property 252
GridColumn (Table.property) 334
Group keyword, table of DataWindow object properties 160
GroupBy property 253
Index

H
Hatch (Brush.property) 191
Header properties.  See Bandname properties
Header.# properties.  See Bandname properties
Header_Bottom_Margin property 254
Header_Top_Margin property 254
Height (Bandname.property) 185
Height property 255
Height.AutoScale (Bandname.property) 185
Height.AutoSize property 255
Height.AutoScale property for bands 186
Help properties 256
HideGrayLine property 258
HideSnaked property 258
HighContrastInk (InkPic.property) 282
Horizontal_Spread property 259
HorizontalScrollMaximum property 260
HorizontalScrollMaximum2 property 260
HorizontalScrollPosition property 260
HorizontalScrollPosition2 property 261
HorizontalScrollSplit property 262
Hour function 66
HSscrollBar (dddw.property) 212
HSscrollBar (Edit.property) 224
HSscrollBar (InkEdit.property) 279
HTextAlign property 262
HTML generation properties 266, 358
HTML link generation properties 263
HTMLDW property 265
HTMLGen properties 266
HTMLTable properties 273
HTMLVersion (HTMLGen.property) 266

I
ID property 274
Identity property 274
If function 67
IgnorePressure (Ink.property) 278
image, in computed field 30
Import.XML.Trace property 275
Import.XML.TraceFile property 276
Import.XML.UseTemplate property 276
IN operator 6

Indent (Tree.property) 344
InfoMaker functions
  Describe 60
  Len 81
  Mid 92
  Pos 107
  Right 115
Initial property 277
initialization files, reading 107, 109
Ink properties 278
InkEdit properties 279
InkEnabled (InkPic.property) 282
InkMode (InkEdit.property) 279
InkPic properties 282
InkPicture properties 282
Inline (XMLGen.property) 359
Insert (Table.property) 334
inserting strings 112, 113
Int function 68
integer
  converting to 68
  converting to char 33
Integer function 68
Invert property 284
IsDate function 69
IsExpanded function 70
IsNull function 70
IsNumber function 71
IsRowModified function 71
IsSelected function 72
IsTime function 73

K
Key property 286
KeyClause property 286

L
Label (Axis.property) 177
Label properties 287
LabelDispAttr (Axis.property) 177
LabelDispAttr font properties.  See DispAttr font properties
Large function 74
Index

Last function 76
LastRowOnPage property 289
Left function 78
Left_Margin property 289
LeftA function 79
LeftText (Checkbox.property) 192
LeftText (RadioButtons.property) 314
LeftTrim function 80
Legend property 290
Legend.DispAttr font properties. See DispAttr font properties
Len function 80
LenA function 81
length, string 80, 81
less than operator 6
less than or equal to operator 6
Level property 291
LIKE operator 6
limit 32
Limit (dddw.property) 212
Limit (ddlb.property) 216
Limit (Edit.property) 224
Limit (InkEdit.property) 279
Lines (dddw.property) 212
Link (HTML.property) 263
LinkArgs (HTML.property) 263
LinkTarget (HTML.property) 263
LinkUpdateOptions property 291
Log function 81
logarithms 81, 82
logical expressions, truth table
logical operators 9
LogTen function 82
Long function 82
longs, converting to 82
LookUpDisplay function 83
Lower function 84
lowercase 84

M
MajorDivisions (Axis.property) 177
MajorGridLine (Axis.property) 177
MajorTic (Axis.property) 177
Margin (Print.property) 304
Mask (EditMask.property) 228
masks, matching 84
Match function 84
Max function 87
maximum value
below a limit 68
columns 87
crosstabs 43, 44
MaximumValue (Axis.property) 177
Median function 89
Message.Title property 292
metacharacters 84
Method (Table.sqlaction.property) 337
Mid function 91
MidA function 92
Min function 92
minimum value
above a limit 32
columns 92
crosstabs 45, 47
MinimumValue (Axis.property) 177
MinorDivisions (Axis.property) 177
MinorGridLine (Axis.property) 177
MinorTic (Axis.property) 177
Minute function 94
Mod function 95
Mode (Background.property) 181
Mode function 95
modulus 95
Month function 98
month, obtaining the day of 57
Moveable property 292
multiplication operator 5

N
Name (dddw.property) 212
Name (Edit.property) 224
Name property 293
negative numbers 119
Nest_Arguments property 293
Nested property 294
NetscapeLayers (HTMLGen.property) 266
NewPage property 295
NilIsNull (dddw.property) 212
NilIsNull (ddlb.property) 216
NilIsNull (Edit.property) 224

DataWindow Object Reference 371
Index

NilIsNull (InkEdit.property) 279
NOT BETWEEN operator 6, 7
not equal operator 6
NOT IN operator 6, 8
NOT LIKE operator 6
NOT operator 6, 9
NoUserPrompt property 296
Now function 98
NoWrap (HTMLTable.property) 273
null
   checking 70
   ignored in aggregate 28, 35, 51, 88, 90, 94, 96
Number function 99
NumberJSFile (HTMLGen.property) 266
numbers
   checking string 71
   determining maximum 32
   determining sign of 119
   logarithm of 81, 82
   multiplying by pi 105
   of day of week 58
   random 110
   returning remainder 95
   rounding 116
   truncating 135
   U.S. format 14
numeric functions
   Abs 24
   ACos 24
   ASin 26
   ATan 27
   Ceiling 32
   Cos 34
   Exp 61
   Fact 61
   Int 68
   Log 81
   Mod 95
   Pi 105
   Rand 110
   Round 116
   Sign 119
   Sin 120
   Sqrt 123
   Tan 133
   Truncate 135

O
Object (HTML Gen.property) 266
ObjectName (HTMLGen.property) 266
Objects property 296
Off (Checkbox.property) 192
OLE.Client properties 297
OLEClass property 297
On (Checkbox.property) 192
operators
   arithmetic 5
   concatenation 10
   logical 9
   precedence 11
   relational 5
OR operator 9
Orientation (Print.property) 304
OriginLine (Axis.property) 177
Other (Checkbox.property) 192
Outline (Print.Preview.property) 302
OverlapPercent property 298
Override>Edit (Criteria.property) 199
OverridePrintJob (Print.property) 304

P
page
   absolute 100
   current 100
   current horizontal 101
   total 101
   total across 102
Page (Print.property) 304
Page function 100
PageAbs function 100
PageAcross function 101
PageCount function 101
PageCountAcross function 102
PageSize (HTMLGen.property) 266
paging, client-side 269
PagingMethod (HTMLGen.property) 266
Paper (Print.property) 304
parsing strings 78, 79, 106, 107
Password (Edit.property) 224
pattern matching 84
Pen properties 299
Pentip (Ink.property) 278
Percent function 103
PercentWidth (dddw.property) 212
period in text patterns 85
Perspective property 300
Pi function 105
pictures in computed fields 20, 30
PictureSizeMode (InkPic.property) 282
Pie.DispAttr font properties. See DispAttr font properties
PlotNullData property 300
plus sign in text patterns 85
Pointer (Bandname.property) 185
Pointer property 301
Pos function 106
PosA function 107
positive numbers 119
precedence of operators 11
Preview (Print.property) 304
primary buffer 117
PrimaryLine (Axis.property) 177
Print properties 304
PrintPreview properties 302
Printer property 310
PrintName (Print.property) 304
Procedure (Table.property) 334
Processing property 311
profile files, reading 107, 109
ProfileInt function 107
ProfileString function 109
Prompt (Print.property) 304
properties, in expressions 60
property expressions conditional 12
Protect property 311
PublishPath (CSSGen.property) 200
PublishPath (JSGen.property) 285
PublishPath (XMLGen.property) 359
PublishPath (XSLTGen.property) 361

Q
Quality (Print.property) 304
QueryClear property 312
QueryMode property 313
QuerySort property 314
question mark in text patterns 85

R
RadioButton properties 314
Rand function 110
random numbers, obtaining 110
Range property 316
ReadOnly (EditMask.property) 228
Real function 110
RecognitionTimer (InkEdit.property) 279
relational operators 5
RelativeDate function 111
RelativeTime function 111
remainder 95
Replace function 112
ReplaceA function 113
ReplaceTabWithSpace property 317
Report property 318
Required (Criteria.property) 199
Required (dddw.property) 212
Required (ddlb.property) 216
Required (Edit.property) 224
Required (EditMask.property) 228
Required (InkEdit.property) 279
ResetPageCount property 318
Resizable property 318
ResourceBase (CSSGen.property) 200
ResourceBase (HTMLGen.property) 266
ResourceBase (JSGen.property) 285
ResourceBase (XMLGen.property) 359
ResourceBase (XSLTGen.property) 361
Retrieve property 319
Retrieve.AsNeeded property 319
RGB function 113
RichText properties 320
Right function 115
RightA function 115
RightTrim function 116
Rotation property 320
Round function 116
RoundTo (Axis.property) 177
RoundToUnit (Axis.property) 177
Row.Resize property 321
RowCount function 117
RowHeight function 117
rows and bands 65
checking if modified 71
Index

getting current 20, 65
height 117
in primary buffer 117
modification status 71
selecting 72
Rows (Crosstab.property) 200
Rows Per Page (HTMLGen.PageSize) 266
Rows_Per_Detail property 321
Rulers (Print.Preview.property) 302

S
Scale (Checkbox.property) 192
Scale (Print.property) 304
Scale (RadioButtons.property) 314
ScaleType (Axis.property) 177
ScaleValue (Axis.property) 177
Second function 118
SecondaryLine (Axis.property) 177
SecondsAfter function 119
Select (Table.property) 334
Selected property 322
Selected.Data property 323
Selected.Mouse property 323
selection, of rows 72
SelectNodeByMouse (Tree.property) 344
SelfLink (HTMLGen.property) 266
SelfLinkArgs (HTMLGen.property) 266
Series property. See Axis properties
SessionSpecific (CSSGen.property) 200
ShadeBackEdge (Axis.property) 177
ShadeColor property 324
ShowBackColorOnXP property 324
ShowConnectLines (Tree.property) 344
ShowDefinition property 325
ShowLeafNodeConnectLines (Tree.property) 344
ShowList (dddw.property) 212
ShowList (ddlb.property) 216
ShowTreeNodeIcon (Tree.property) 344
Sign function 119
Sin function 120
sine 120
size of string 80, 81
SizeToDisplay property 326
SlideLeft property 326
SlideUp property 327
Small function 120
Sort (Axis.property) 177
Sort (Table.property) 334
Sort property 328
Sorted (ddlb.property) 216
SourceNames (Crosstab.property) 200
Space function 122
spaces
deleting leading 80
deleting trailing 116
inserting in a string 122
removing from strings 134
Spacing property 328
Sparse property 329
Spin (EditMask.property) 228
SpinIncr (EditMask.property) 228
SpinRange (EditMask.property) 228
SQLSelect (Table.property) 334
Sqrt function 123
square root 123
standard deviation 124, 126
StateIconAlignMode (Tree.property) 344
StaticMode (Crosstab.property) 200
StDev function 124
StDevP function 126
Storage property 329
String function 129
string functions
Asc 25
AscA 25
Char 33
CharA 33
Fill 62
FillA 63
Left 78
LeftA 79
LeftTrim 80
Len 80
LenA 81
Match 84
Mid 91
MidA 92
Pos 106
PosA 107
Replace 112
ReplaceA 113
Right 115
RightA 115
RightTrim 116
Space 122
Trim 134
Upper 136
WordCap 141
StringJSFile (HTMLGen.property) 266
strings
comparing 8
concatenating 10
converting 55, 82, 99, 110
deleting leading spaces 80
detecting contents 69, 71, 73
extracting 91, 92
finding substrings 106, 107
lowercase 84
uppercase 136
Style (Edit.property) 224
Style (Pen.property) 299
Style keyword, table of DataWindow object properties 165
StyleSheet (HTMLTable.property) 273
substring
extracting 91, 92
finding 106, 107
replacing 112, 113
subtraction operator 5
Sum function 131
Summary properties. See Bandname properties
Suppress (Bandname.property) 185
SuppressEventProcessing property 330
Syntax property 331
Syntax.Data property 331
Syntax.Modified property 332
system and environment functions
ProfileInt 107
ProfileString 109
system date 134
system time 98

T
TabIndexBase (HTMLGen.property) 266
Table properties 334
Table property
Create function 332
InkPicture objects 333
TableBlob objects 333
Table SQLAction properties 337
TabSequence property 339
Tag property 340
Tan function 133
tangent 133
Target property 340
Template property 341
text
finding substrings 106, 107
metacharacters 84
setting color of 114
Text (Checkbox.property) 192
Text property 341
time
checking string 73
converting to data type 133
DateTime data type 56
minutes 94
now 98
relative 111
seconds 118, 119
Time function 133
Timer_Interval property 342
Title keyword, table of DataWindow object properties 167
Title property 343
Title.DispAttr font properties. See DispAttr font properties
Today function 134
total of values
columns 131
crosstabs 48, 50
running 53
Trail_Footer property 343
Trailer.# properties. See Bandname properties
Transparency (Ink.property) 278
Tree properties 344
Tree.Level properties 347
TreeNodeIconName (Tree.Leaf property) 347
Trim function 134
Truncate function 135
truth table for boolean expressions 9
Index

Type (Table.sqlaction.property) 337
Type property 349

VScrollBar (InkEdit.property) 279
VTextAlign property 356

W
week, day of 57, 58
Width (HTMLTable.property) 273
Width (Ink.property) 278
Width (Pen.property) 299
Width property 356
WordCap function 141

X
X property 357
X1, X2 properties 357
XHTMLGen.Browser 358
XHTMLGen.PublishPath 285, 359, 361
XHTMLGen.ResourceBase 285, 359, 361
XML generation properties 285, 359, 361

Y
Y property 362
Y1, Y2 properties 362
Year function 142

Z
zero, determining 119
Zoom (Print.Preview.property) 302
Zoom property 363

Values (Crosstab.property) 200
Values properties, graphs. See Axis property
Values property, columns 352
Var function 136
variance 136, 139
VarP function 139
Vertical_Size property 353
Vertical_Spread property 354
VerticalScrollMaximum property 354
VerticalScrollPosition property 355
Visible property
about 355
VScrollBar (dddw.property) 212
VScrollBar (ddlb.property) 216
VScrollBar (Edit.property) 224

Type property 349