Powersoft®

Open Tools from Sybase, Inc.

PowerBuilder

DataWindow Reference

Version 6

PowerBuilder®

AA0520

October 1997

Contents

About This Book

Subject

This book provides reference information for the DataWindow object. It lists the DataWindow functions and properties and includes the syntax for accessing properties and data.

Audience

This book is for anyone defining DataWindow objects and writing scripts that deal with DataWindow objects. It assumes that:

- You are familiar with the DataWindow painter. If not, see the *PowerBuilder User's Guide*.
- You have a basic familiarity with PowerScript. If not, see the *PowerScript Reference*.

CHAPTER 1

Operators and Expressions

About this chapter

You use an expression to request that a report or DataWindow object perform a computational operation. You create expressions with DataWindow operators and functions. This chapter explains how expressions work and how to write them.

Contents

Where you use expressions

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

In the DataWindow and Report painters

DataWindow painter expressions are associated with DataWindow objects and reports. You specify them in both the DataWindow and Report painters. You can also specify expressions in the Database painter, although these expressions have a slightly different format.

FOR INFO For information about DataWindow painter functions that you can use in expressions, see "Using DataWindow painter and InfoMaker functions" on page 16, or look up the function you want in online Help.

In a DataWindow object or report, you use expressions in these ways:

In this painter	Expressions are used in
DataWindow painter and Report painter	Computed fields Conditional expressions for property values Validation rules Filters Sorting Series and values in graphs Columns, rows, and values in crosstabs
Database painter	Validation rules

Other places where you create expressions

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 painter operations and functions, to create expressions.

In scripts, you use operators and expressions in the PowerScript code you write. PowerScript operators and functions are documented in the *PowerScript Reference*.

You can change the value of DataWindow properties in scripts. The format for expressions you specify in scripts is different from the same expression specified in the painter. These differences are described in Chapter 4, "Accessing DataWindow Properties in Scripts".

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

In computed fields

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

Expression	Description
Today ( )	Displays the date using the Today function
Salary/12	Computes the monthly salary
Sum (Salary for group 1)	Computes the salary for the first group using the Sum aggregate function
Price*Quantity	Computes the total cost

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:

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

In validation rules for table columns

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

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

If you include the column name in the expression, you will 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 data type 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 data type conversion function.

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:

Expression in Database painter	Expression in DataWindow painter	Description
@column >= 10000	Integer(GetText())>= 10000	If a user enters a salary below $10,000, an error message displays
@column IN (100, 200, 300)	Integer(GetText()) IN (100, 200, 300)	If a user does not enter a department ID of 100, 200, or 300, an error message displays
@salary > 0	Long(GetText()) > 0	If a user doesn't enter a positive number, an error message displays
Match(@disc_price, "^[0-9]+$") and @disc_price < Full_Price	Match(GetText( ), "^[0-9]+$") and Real(GetText()) < Full_Price	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

Operators used in 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.

Reports and DataWindow objects support the following types of operators:

- **Arithmetic** for numeric data types
- **Relational** for all data types
- **Logical** for all data types
- **Concatenation** for string data types

Arithmetic operators

When you write a DataWindow painter expression, you can use the following arithmetic operators:

Operator	Meaning	Example
+	Addition	SubTotal + Tax
-	Subtraction	Price - Discount
*	Multiplication	Quantity * Price
/	Division	Discount / Price
^	Exponentiation	Rating ^ 2.5

Multiplication and division

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

Expression	Value
20.0/3	6.666666666666667
3*(20.0/3)	20
Truncate(20.0/3,4)	6.6666

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.

Relational operators

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

When you write a DataWindow painter expression, you can use the following relational operators (for more information about LIKE, IN, and BETWEEN, see after the table):

Operator	Meaning	Example
=	Is equal to	Price = 100
>	Is greater than	Price > 100
<	Is less than	Price < 100
<>	Is not equal to	Price <> 100
>=	Greater than or equal to	Price >= 100
<=	Less than or equal to	Price <= 100
NOT =	Is not equal to	Price NOT= 100
LIKE	Matches this specified pattern	Emp_lname LIKE 'C%' OR Emp_lname LIKE 'G%'
IN	Is in this set of values	Dept_id IN (100, 200, 500)
BETWEEN	Is within this range of values. The range includes the first and last values	Price BETWEEN 1000 AND 3000
NOT LIKE	Does not match this specified pattern	Emp_lname NOT LIKE 'C%' AND Emp_lname NOT LIKE 'G%'
NOT IN	Is not in this set of values	Dept_id NOT IN (100, 200, 500)

Operator	Meaning	Example
NOT BETWEEN	Is outside this range of values. The range includes the first and last values	Price NOT BETWEEN 1000 AND 2000

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 special characters to match unknown characters in a pattern:

Special character	Meaning	Example
% (percent)	Matches any group of characters	Good% matches all names that begin with Good
_ (underscore)	Matches any single character	Good _ _ _ matches all 7-letter names that begin with Good

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:

```
If(emp_lname LIKE'F%',RGB(255,0,0),RGB(255,255,255))
```

BETWEEN and NOT BETWEEN operators

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:

```
If(salary BETWEEN 50000 AND 100000, RGB(255,0,0),
RGB(255,255,255))
```

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:

```
If(dept_id IN (300,400) and salary BETWEEN 50000 AND
100000, RGB(255,0,0), RGB(255,255,255))
```

Logical operators

You use logical operators to combine boolean expressions into a larger boolean expression. The result is always TRUE or FALSE:

Operator	Meaning	Example
NOT	Logical negation If A is true, NOT A is false If A is false, NOT A is true	NOT Price = 100
AND	Logical and A AND B is true if both are true A AND B is false if either is false	Tax > 3 AND Ship < 5
OR	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	Tax > 3 OR Ship < 5

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:

If one expression has this value	And the logical operator is	And if another expression has this value	The resulting expression has this value
TRUE	AND	TRUE	TRUE
TRUE	AND	FALSE	FALSE
FALSE	AND	TRUE	FALSE
FALSE	AND	FALSE	FALSE
TRUE	OR	TRUE	TRUE
TRUE	OR	FALSE	TRUE
FALSE	OR	TRUE	TRUE
FALSE	OR	FALSE	FALSE
NOT TRUE	AND	TRUE	FALSE

If one expression has this value	And the logical operator is	And if another expression has this value	The resulting expression has this value
NOT TRUE	AND	FALSE	FALSE
NOT FALSE	AND	TRUE	TRUE
NOT FALSE	AND	FALSE	FALSE
NOT TRUE	OR	TRUE	TRUE
NOT TRUE	OR	FALSE	FALSE
NOT FALSE	OR	TRUE	TRUE
NOT FALSE	OR	FALSE	TRUE

Comparing strings

When you compare strings, the comparison is case sensitive. Trailing blanks are significant.

Case-sensitivity examples

Assume City1 is "Austin" and City2 is "AUSTIN". Then:

```
City1=City2
```

returns FALSE.

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

```
Upper(City1)=Upper(City2)
```

returns TRUE.

FOR INFO For information about these functions, see "Using DataWindow painter and InfoMaker functions" on page 16.

Trailing blanks examples

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

```
City1=City2
```

returns FALSE.

To prevent blanks from affecting a comparison, remove them with one of the Trim functions: Trim, LeftTrim, or RightTrim.

For example:

```
Trim(City1)=Trim(City2)
```

returns TRUE.

FOR INFO For information about these functions, see "Using DataWindow painter and InfoMaker functions" on page 16.

Concatenation operator

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.

String expression	Value
"over" + "stock"	overstock
Lname + ', ' + Fname	If Lname is Hill and Fname is Craig, then "Hill, Craig"

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 expressions

To ensure predictable results, all operators in an expression are evaluated in a specific order of precedence. When the operators have the same precedence, the DataWindow object evaluates them left to right.

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

Operator	Purpose
( )	Grouping
^	Exponentiation
*, /	Multiplication and division
+, -	Addition and subtraction; string concatenation
IN,LIKE,BETWEEN	SQL SELECT statement conditions
NOT	Logical negation
=, >, <, <=, >=, <>	Relational operators
AND,OR	Logical and and logical or

Overriding the precedence order

Since expressions in parentheses are evaluated first, to override the precedence order, enclose expressions in parentheses. Within each set of parentheses, precedence order applies.

In the expression **x+y*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*(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 painter expressions in scripts

In a script, you use functions 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.

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

```
ll_rownum = dw_1.GetRow()
```

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

```
ls_first = dw_1.Object.DataWindow.FirstRowOnPage
ls_last = dw_1.Object.DataWindow.LastRowOnPage
dw_1.Title = "Rows " + ls_first + " to " + ls_last
```

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

DataWindow painter 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 painter and InfoMaker functions" on page 16.

Using Evaluate within the Describe function

The Describe function provides a way to evaluate these expressions outside the context their usual context. The Evaluate function, which is only used within Describe, allows you to evaluate DataWindow expressions within a script using data in the DataWindow.

Evaluate has the following syntax:

dwcontrol .Describe ("Evaluate (*'expression'* , *rownumber*) ")

Expression is the expression you want to evaluate and rownumber is the number of the row for which you want to evaluate the expression. The expression can include DataWindow painter functions that cannot be called in a script.

This example displays in the title of the DataWindow control the current page for the current row in the DataWindow:

```
string ls_modstring, ls_rownum
ls_rownum = String(dw_1.GetRow())

ls_modstring = "Evaluate('Page()'," + ls_rownum +")"
// The resulting string, for row 99, would be:
// Evaluate('Page()', 99)

Parent.Title = &
"Current page: "+ dw_1.Describe(ls_modstring)
```

This example returns the display value for the dept_id column for row 5:

```
dw_1.Describe("Evaluate('LookUpDisplay(dept_id)', 5)")
```

Expressions that apply to all rows

To evaluate an expression that applies to all rows, specify 0 for the *rownumber* argument. This example calculates the sum of the salary column in the current DataWindow. It will return the expression's result or "!" if the expression is not valid:

```
dw_1.Describe("Evaluate('Sum(Salary)', 0)")
```

Evaluating user-specified expressions

In some types of applications, you might use Evaluate to get the result of an expression the user specifies. For example, users might specify the type of aggregation they want to see. This example evaluates an expression specified in a SingleLineEdit. It applies to all rows:

```
dw_1.Describe("Evaluate('" + sle_expr.Text + "', 0)")
```

Using Evaluate with conditional property expressions

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 the Emp_Id for any of the rows. If you query the property value for this column during execution, 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. That property will be set on a row-by-row basis during execution.

For example, you may 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:

```
If(IsRowNew(), 0, 1)
```

When you query the Protect property during execution, 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)"
```

Getting a property value for a particular row

To obtain the actual value of the Protect property for a particular row, you need to strip off the default value and the tab and evaluate the returned expression for the desired row. After stripping off the extra information, you can construct an expression for Describe that uses the Evaluate function.

This example checks whether the value of the Protect property for emp_id is a constant or a conditional expression. If it is a conditional expression, the script builds a string for the Describe function that uses Evaluate to get the value for of Protect for the current row:

```
string ls_protect, ls_eval
long ll_row


ll_row = dw_1.GetRow()
ls_protect = dw_1.Object.id.Protect

IF NOT IsNumber(ls_protect) THEN

   // Get the expression following the tab (~t)
   ls_protect = Right(ls_protect, &
      Len(ls_protect) - Pos(ls_protect, "~t"))

   // Build string for Describe. Include a leading
   // quote to match the trailing quote that remains
   ls_eval = "Evaluate(~"" + ls_protect + ", " &
      + String(ll_row) + ")"

   ls_protect = dw_1.Describe(ls_eval)

END IF

// Display result
st_result.Text = ls_protect
```

CHAPTER 2

DataWindow Painter and InfoMaker Functions

About this chapter

This chapter provides syntax, descriptions, and examples of the functions you can use in expressions in PowerBuilder's DataWindow painter and in InfoMaker's Report painter and Form painter.

Contents

After a short introduction and several examples, the functions are listed alphabetically.

Using DataWindow painter and InfoMaker functions

PowerBuilder DataWindow functions and InfoMaker functions are the same functions. In PowerBuilder's DataWindow painter and InfoMaker's Report and Form painters, you can use these functions in expressions for computed fields, filters, validation rules, and graphed data, with some exceptions.

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

FOR INFO For information about expressions, see Chapter 1, "Operators and Expressions".

Return values for functions and expressions

DataWindow painter and InfoMaker expressions can return the following data types:

- Double
- String
- DateTime
- Time

Within an expression, a function can return other data types (such as boolean, date, or integer) but the final value of an expression is converted to one of the four data types.

Restrictions for aggregate functions

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.

User-defined functions in PowerBuilder

In PowerBuilder, you can include user-defined functions in DataWindow painter expressions. The data type of the function's return value can be any of the following: double, string, boolean, date, DateTime, or time. The function must be defined as a global function so that it is available to the DataWindow object.

Formatting for the locally correct display of numbers

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 painter and InfoMaker 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 will display appropriately in whatever countries you deploy applications in. During execution, the locally correct symbols for numbers will display (because PowerBuilder and InfoMaker use the international Control Panel settings) when numbers are interpreted. For example, in countries where comma represents the decimal place and period represents thousands, users will see numbers in those formats during execution.

FOR INFO For information about the locally correct display of dates and day names, see String on page 154 and DayName on page 60.

Four examples

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 may not be applicable, or it may 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's useful to know how many NULL values there are in a particular column.

What you want to do

You are working with the Fin_code table in the Powersoft Demo Database. The Fin_code table has three columns:

Column	What the column is	Allows NULL values?
Code	Unique financial identifier (primary key)	No
Type	Code type: expense or revenue	No
Description	Code description: the department incurring the expense or getting the revenue	Yes

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):

```
Count(description for all)
```

which counts the number of descriptions (that are not NULL);

```
Sum(If(IsNull(description), 1, 0))
```

which returns a 1 if the description column is NULL, a 0 if the description column is NOT NULL, and then adds the total;

```
Count(id for all)
```

which counts the number of IDs (which is also the number of rows);

```
Sum(If(IsNull(description), 1, 1))
```

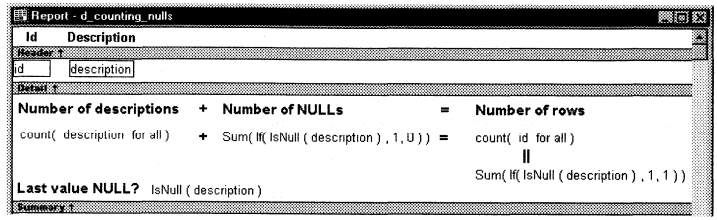
which adds the number of NULLs and NOT NULLs in the description column (which is the total number of rows) and should match the result of the Count(id for all) function; and

```
IsNull(description)
```

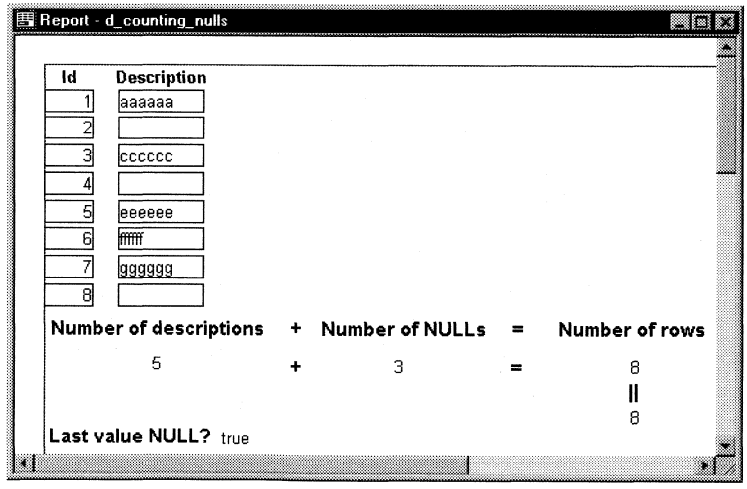
which evaluates whether the last row in the table has a description that is NULL. The return value of the IsNull function is TRUE or FALSE.

What you get

Here's the design for the DataWindow object.



Here's the DataWindow object. The DataWindow object or report shows there are 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:

Avg	Large	Mode	Sum
Count	Last	Percent	Var
CumulativePercent	Max	Small	VarP
CumulativeSum	Median	StDev	
First	Min	StDevP	

About crosstab functions

Although the crosstab functions (CrosstabAvg, CrosstabCount, CrosstabMax, CrosstabMin, and CrosstabSum) 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 can't 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 Powersoft Demo Database 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:

```
Sum(If(sex = "M", 1, 0))
```

which counts the number of males in your company;

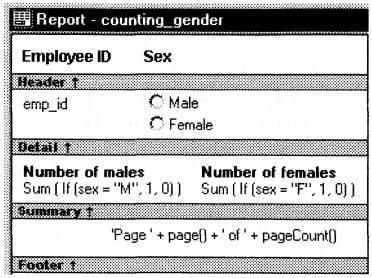
```
Sum(If(sex = "F", 1, 0))
```

which counts the number of females in your company.

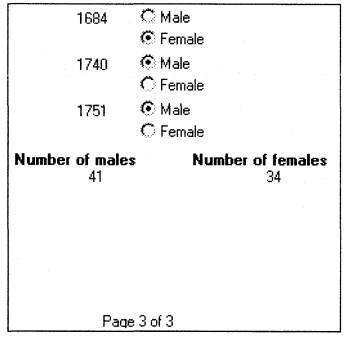
You can also add a Page computed field (by clicking the Page computed field button) in the footer band to display the page number and total pages at the bottom of each page of the DataWindow object.

What you get

Here's what the design of the DataWindow object looks like.



Here's the last page of the DataWindow object in preview, with the total number of males and females in the company displayed.



If you now want more information

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 Click the SQL button so you can edit the data source.

2 Open the Department table in the SQL 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 in the workspace, add two additional computed fields to your DataWindow object:

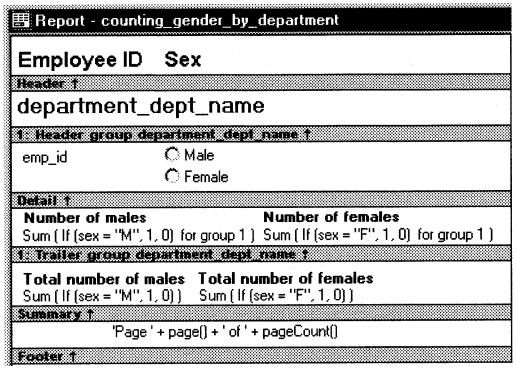
```
Sum(If(sex = "M", 1, 0) for group 1)
```

which counts the number of males in each department;

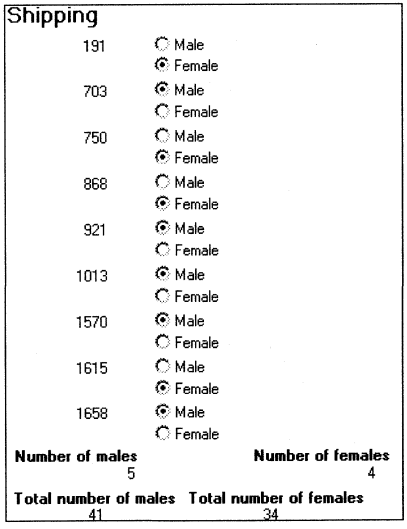
```
Sum(If(sex = "F", 1, 0) for group 1)
```

which counts the number of females in each department.

Here's what the design of the grouped DataWindow object looks like.



Here's 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.



Example 3: creating a row indicator

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

The example is presented in PowerBuilder's DataWindow painter, which is the same as InfoMaker's Report painter. You can use all the functions shown in the example in the Report painter. However, because you can change the current row and change data in a DataWindow object (which you can't do in a report), the example is more interesting to consider in a DataWindow object.

What you want to do

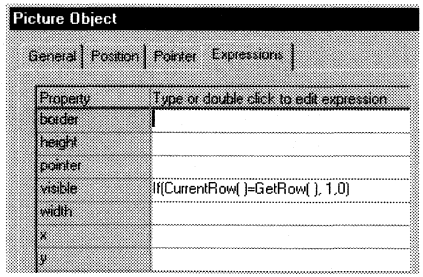
Using the Employee table in the Powersoft Demo Database, you create a DataWindow object using the Emp_id, Emp_fname, Emp_lname, and Salary columns.

In the DataWindow 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.

How to do it

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 (in the Expressions property page of the Picture Object property sheet) an expression for the arrow's visible property:



  The expression results in an arrow displaying in the current row and no arrow displaying in other rows.

- A computed field using the If, CurrentRow, and GetRow functions:

```
If(CurrentRow() = GetRow(),"Current","Not current")
```

which displays the word "Current" when the row is the current row and "Not current" for all other rows

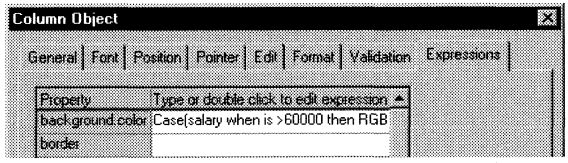
- A computed field (typed on one line) using the Bitmap, CurrentRow, and GetRow functions:

```
Bitmap(If(CurrentRow()=GetRow(),
"c:\pb5i32\ex\code\indicatr.bmp", " "))
```

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

- An expression (on one line in the Expressions property page of the Column Object property sheet) for the Background.Color property of the salary column:

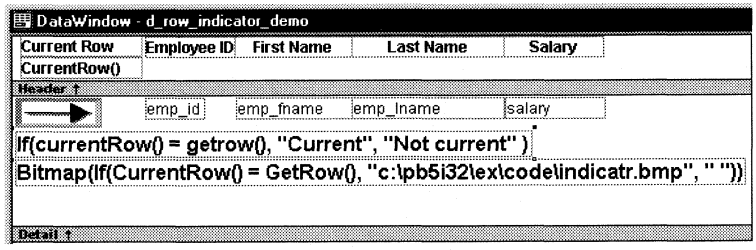
```
Case(salary WHEN IS >60000 THEN RGB(192,192,192)
WHEN IS >40000 THEN RGB(0,255,0) ELSE
RGB(255,255,255))
```



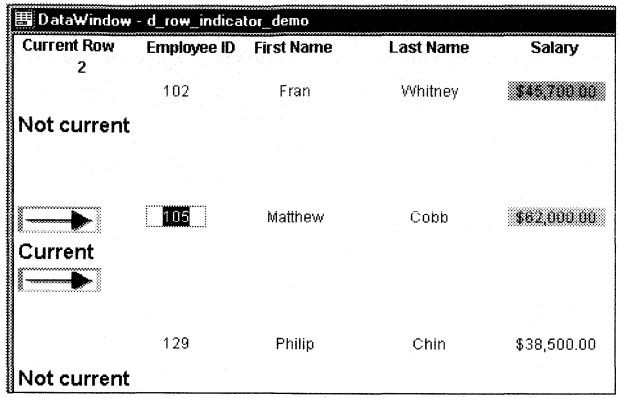
The expression results in a salary above $40,000 displaying in green, a salary above $60,000 displaying in gray, and all other salaries displaying in white.

What you get

Here's what the design of the DataWindow object looks like:



After previewing the DataWindow object, here's what the data looks like with the second row current.



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's more than $40,000; the salary in the second row has a gray background because it's 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 NULLs. 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's 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 we assume that Address2 will only have data if Address1 has data for a particular row.

How to do it

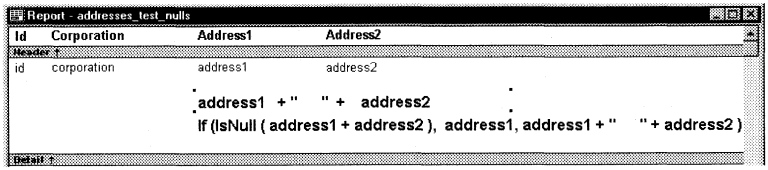
In the detail band, create a computed field that uses the If and IsNull functions:

```
If(IsNull(address1 + address2), address1, address1 +
", " + address2)
```

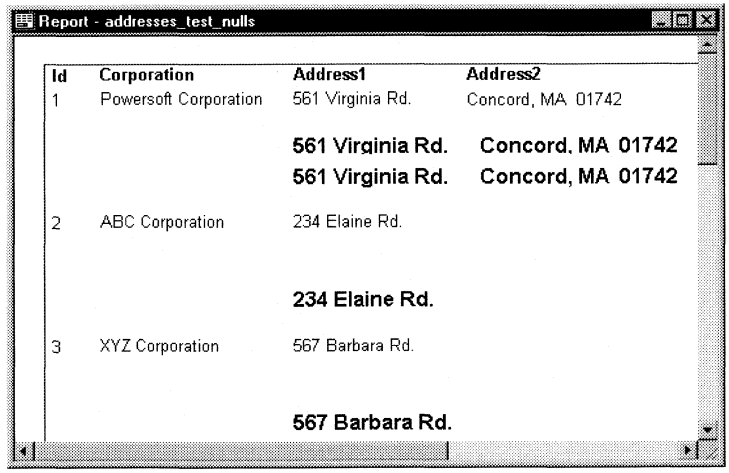
The computed field says this: if the concatenation of the addresses is NULL (because address2 is NULL), then display address1, and if it's not NULL, display both parts of the address separated by a comma.

What you get

Here's what the design of the DataWindow object looks like. It includes both the computed field that doesn't 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.



Abs

Description

Calculates the absolute value of a number.

Syntax

Abs (*n*)

Argument	Description
n	The number for which you want the absolute value

Return value

The data type 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
Abs in the *PowerScript Reference*

Asc

Description

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

Syntax

Asc (*string*)

Argument	Description
string	The string for which you want the ASCII value of the first character

Return value

Integer. Returns the ASCII value of the first character in *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 ASCII value is within the appropriate range.

Examples

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

```
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:

```
IF (Asc(lname) > 64 AND Asc(lname) < 91, lname,
WordCap(lname))
```

See also

Char
WordCap
Asc in the *PowerScript Reference*

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 data type of *column* must be numeric
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 the type of object for *range*. The values to be aggregated are determined by the range specified in the object definition. (See Usage for more information.) Values are: ♦ GRAPH — (Graphs only) The average of values in *column* in the range specified for the Rows option of the graph ♦ OBJECT — (OLE objects only) The average of values in *column* in the range specified for the Rows option of the OLE object
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 data type 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 for the object 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 or OLE presentation style, Rows is always All.
- For Graph objects, Rows can be All, Page, or Group.
- For OLE objects, Rows can be All, Current Row, Page, or Group. The available choices depend on the layer the object 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 DataWindow painter and Report painter. To do the aggregation, a DataWindow object or report always retrieves all rows.

Examples

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

```
Avg(salary)
```

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

```
Avg(salary for group 1)
```

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

```
Avg(#5 for page)
```

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

```
If(Avg(salary for page) > Avg(salary), "Above
Average", " ")
```

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

```
Avg(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:

```
Avg(sale_price for graph)
```

Assuming a DataWindow object, report, or form 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:

```
Avg(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, RLE, or WMF file), or an expression that evaluates to a string containing the name of an image file

Return value

The special data type 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"))
```

Case

Description

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

Syntax

Case (*column* WHEN *value1* THEN *result1* { WHEN *value2* THEN *result2* { ... } } { ELSE *resultelse* })

Argument	Description
column	The column or expression whose values you want to test. *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. *Column* is compared to each *valuen*
WHEN (optional)	Introduces a value-result pair. At least one WHEN is required
valuen	One or more values that you want to compare to values of *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)
THEN	Introduces the result to be returned when *column* matches the corresponding *valuen*
resultn	An expression whose value is returned by Case for the corresponding *valuen*. All *resultn* values must have the same data type
ELSE (optional)	Specifies that for any values of *column* that don't match the values of *valuen* already specified, Case returns *resultelse*
resultelse	An expression whose value is returned by Case when the value of *column* doesn't match any WHEN *valuen* expression

Return value

The data type of *resultn*. Returns the result you specify in *resultn*.

Usage

If more than one WHEN clause matches *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 then $50,000, and blue otherwise:

```
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:

```
Case(emp_id WHEN 101 THEN 255 WHEN 102 THEN
RGB(100,100,100) 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:

```
Case(marital_status WHEN 'S'THEN 'Single' WHEN 'M'
THEN 'Married' ELSE 'Unknown')
```

See also

If

Ceiling

Description

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

Syntax

Ceiling (*n*)

Argument	Description
n	The number for which you want the smallest whole number that is greater than or equal to it

Return value

The data type of *n*. Returns the smallest whole number that is greater than or equal to *n*.

Examples

These expressions both return - 4:

```
Ceiling(-4.2)
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:

```
If(discount_amt <= Ceiling(discount_factor * price),
String(discount_amt), "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:

```
discount_amt <= Ceiling(discount_factor * price)
```

See also

Int
Round
Truncate
Ceiling in the *PowerScript Reference*

Char

Description

Converts an integer to a character.

Syntax

Char (*n*)

Argument	Description
n	The integer you want to convert to a character

Return value

String. Returns the character whose ASCII value is *n*.

Examples

This expression returns the escape character:

```
Char(27)
```

See also

Asc
Char in the *PowerScript Reference*

Cos

Description

Calculates the cosine of an angle.

Syntax

Cos (*n*)

Argument	Description
n	The angle (in radians) for which you want the cosine

Return value

Double. Returns the cosine of *n*.

Examples

This expression returns 1:

```
Cos(0)
```

This expression returns .540302:

```
Cos(1)
```

This expression returns - 1:

```
Cos(Pi(1))
```

See also

Pi
Sin
Tan
Cos in the *PowerScript Reference*

Count

Description

Calculates the total number of rows in the specified column.

Syntax

Count (*column* { FOR *range* { DISTINCT { *expres1* {, *expres2* {, ... } } } } })

Argument	Description
column	The column for which you want the number of 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
FOR *range* (optional)	The data that will be included in the count. For most presentation styles, values for *range* are: ♦ ALL — (Default) The count of all rows in *column* ♦ GROUP *n* — The count of rows in *column* in the specified group. Specify the keyword GROUP followed by the group number: for example, GROUP 1 ♦ PAGE — The count of the rows in *column* on a page For Crosstabs, specify CROSSTAB for *range*: ♦ CROSSTAB — (Crosstabs only) The count of all rows in *column* in the crosstab For Graph and OLE objects, specify the type of object for *range*. The values to be aggregated are determined by the range specified in the object definition. (See Usage for more information.) Values are: ♦ GRAPH — (Graphs only) The count of values in *column* in the range specified for the Rows option of the graph ♦ OBJECT — (OLE objects only) The count of values in *column* in the range specified for the Rows option of the OLE object
DISTINCT (optional)	Causes Count to consider only the distinct values in *column* when counting the rows. 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

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

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

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 DataWindow painter and Report painter. To do the aggregation, a DataWindow object or a report always retrieves all rows.

Examples

This expression returns the number of rows in the column named emp_id that are not NULL:

```
Count(emp_id)
```

This expression returns the number of rows in the column named emp_id of group 1 that are not NULL:

```
Count(emp_id for group 1)
```

This expression returns the number of dept_ids that are distinct:

```
Count(dept_id for all DISTINCT)
```

This expression returns the number of regions with distinct products:

```
Count(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:

```
Count(#3 for page)
```

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 or report.

Syntax

CrosstabAvg (*n* {, *column*, *groupvalue* })

Argument	Description
n	The number of the crosstab-values expression for which you want the average of the returned values. The crosstab expression 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

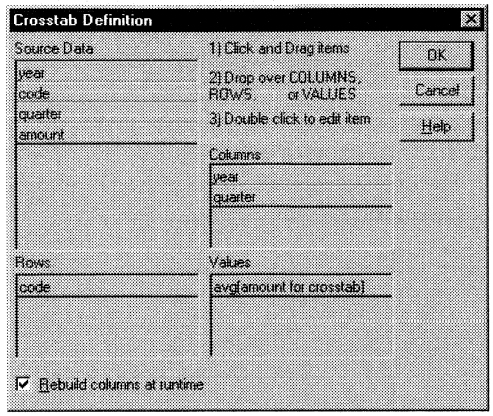
Double. Returns the average of the crosstab values returned by expression *n* for all the column values or, optionally, for a subset of column values.

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 PowerBuilder and InfoMaker use the functions in a crosstab When PowerBuilder and InfoMaker generate a crosstab from your definition, they automatically create the appropriate computed fields using the Crosstab functions. To understand the functions, consider a crosstab with two columns (year and quarter) a row (product), and the values expression Avg(amount for crosstab). The Crosstab Definition dialog box looks like this.



When you define the crosstab described above, the DataWindow painter or Report 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:

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

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

```
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 DataWindow painter or Report painter looks like this.

	Year	Quarter	
Header [1] ↑			
	@year	@year Avg	
Header [2] ↑			
Code	@quarter		Grand Avg
Header [3] ↑			
code	amount	crosstabavg(1, 2, "@year")	crosstabavg(1)
Detail ↑			
"Grand Avg"	avg(amount for all)	avg(avg_amount for all)	avg(grand_avg_amount for all)
Summary ↑			
Footer ↑			

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.

	Year	Quarter								
	1992				1992 Avg	1993				1993 Avg
Code	Q1	Q2	Q3	Q4		Q1	Q2	Q3	Q4	
e1	101	93	129	145	**117**	153	149	157	163	**156**
e2	403	459	609	632	**526**	643	687	898	923	**788**
e3	1,437	2,033	2,184	2,145	**1,950**	2,478	2,998	3,702	3,600	**3,195**
e4	623	784	856	1,043	**827**	1,051	1,158	1,459	1,439	**1,277**
e5	381	402	412	467	**416**	523	749	723	748	**686**
r1	1,023	2,033	2,998	3,014	**2,267**	3,114	3,998	6,523	7,267	**5,226**
r2	234	459	601	944	**560**	992	1,195	1,704	1,823	**1,429**
Grand Avg	**600**	**895**	**1,113**	**1,199**	**952**	**1,279**	**1,562**	**2,167**	**2,280**	**1,822**

1994				1994 Avg	
Q1	Q2	Q3	Q4		Grand Avg
198	204	214	231	**212**	**161**
921	975	984	982	**966**	**760**
4,139	4,500	4,532	5,298	**4,617**	**3,254**
1,462	1,472	1,439	1,498	**1,468**	**1,190**
798	983	956	963	**925**	**675**
9,144	10,988	13,567	15,199	**12,225**	**6,572**
1,839	2,011	2,897	4,129	**2,719**	**1,569**
2,643	**3,019**	**3,513**	**4,043**	**3,304**	**2,026**

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:

```
CrosstabAvg(1, 2, "@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 th ' PowerBuilder and InfoMaker group the columns. To let PowerBuilder a. oMaker define 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 objects or report presentation styles and define computed fields in the group header or foote sing noncrosstab aggregation functions, such as Avg, Sum, or Max.

Reviewing the expressions
To review the expressions defined for the crosstab values, open the Crosstab Definition dialog box (display the popup menu in an unused area of the crosstab and then select Crosstab).

Examples

The first two examples use the crosstab expressions shown below:

```
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):

```
CrosstabAvg(1)
```

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

```
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

CrosstabCount
CrosstabMax
CrosstabMin
CrosstabSum

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, 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 or report.

Syntax

CrosstabCount (*n* {, *column*, *groupvalue* })

Argument	Description
n	The number of the crosstab-values expression for which you want the total number of returned values
column (optional)	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
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

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.

FOR INFO 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 (display the popup menu in an unused area of the crosstab and then select Crosstab).

Examples

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

```
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):

```
CrosstabCount(1)
```

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

```
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:

```
CrosstabCount(1, 2, "@year")
```

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

```
CrosstabCount(1)
```

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

See also

CrosstabAvg
CrosstabMax
CrosstabMin
CrosstabSum

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 or report.

Syntax

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

Argument	Description
n	The number of the crosstab-values expression for which you want the maximum returned value. The expression's data type 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

Double. Returns the maximum value 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 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 INFO 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 (display the popup menu in an unused area of the crosstab and then select Crosstab).

Examples

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

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

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

```
CrosstabMax(1)
```

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

```
CrosstabMax(2)
```

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

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

```
CrosstabMax(1, 2, "@year")
```

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

```
CrosstabMax(1)
```

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

See also

CrosstabAvg
CrosstabCount
CrosstabMin
CrosstabSum

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 or report.

Syntax

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

Argument	Description
n	The number of the crosstab-values expression for which you want the minimum return value. The expression's data type 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

Double. Returns the minimum value 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 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 INFO 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 (display the popup menu in an unused area of the crosstab and then select Crosstab).

Examples

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

```
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):

```
CrosstabMin(1)
```

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

```
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:

```
CrosstabMin(1, 2, "@year")
```

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

```
CrosstabMin(1)
```

For an example illustrating how the DataWindow painter and Report painter automatically define a crosstab by creating computed fields using the crosstab functions, see CrosstabAvg.

See also

CrosstabAvg
CrosstabCount
CrosstabMax
CrosstabSum

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 or report.

Syntax

CrosstabSum (*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 data type 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

Double. Returns the total of the 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 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 INFO 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 (display the popup menu in an unused area of the crosstab and then select Crosstab).

Examples

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

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

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

```
CrosstabSum(1)
```

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

```
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:

```
CrosstabSum(1, 2, "@year")
```

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

```
CrosstabSum(1)
```

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

See also

CrosstabAvg
CrosstabCount
CrosstabMax
CrosstabMin

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 data type 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 the type of object for *range*. The values to be aggregated are determined by the range specified in the object definition. (See Usage for more information.) Values are: ♦ GRAPH — (Graphs only) The cumulative percentage of values in *column* in the range specified for the Rows option of the graph ♦ OBJECT — (OLE objects only) The cumulative percentage of values in *column* in the range specified for the Rows option of the OLE object

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 for the object 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 or OLE presentation style, Rows is always All.
- For Graph objects, Rows can be All, Page, or Group.
- For OLE objects, Rows can be All, Current Row, Page, or Group. The available choices depend on the layer the object 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 DataWindow painter and Report painter. To do the aggregation, a DataWindow object or a report 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 in the Graph Object property sheet 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

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 data type 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 For Graph and OLE objects, specify the type of object for *range*. The values to be aggregated are determined by the range specified in the object definition. (See Usage for more information.) Values are: ♦ GRAPH — (Graphs only) The cumulative sum of values in *column* in the range specified for the Rows option of the graph ♦ OBJECT — (OLE objects only) The cumulative sum of values in *column* in the range specified for the Rows option of the OLE object

Return value

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

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:

```
CumulativeSum(salary)
```

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

```
CumulativeSum(salary for group 1)
```

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

```
CumulativeSum(salary for group 1)
```

This expression entered in the Value box on the Data property page in the Graph Object property sheet returns the running total for the salary column for the values in the graph that are not NULL:

```
CumulativeSum(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:

```
CumulativeSum(salary for crosstab)
```

See also

CumulativePercent

CurrentRow

Description

Reports the number of the current row (the row with focus) in a DataWindow object or form.

Syntax

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:

```
CurrentRow()
```

This expression for a computed object 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 in the DataWindow object or form, an arrow marks where the user is:

```
Bitmap(If(CurrentRow() = GetRow(),"arrow.bmp",""))
```

Alternatively, this expression for the Visible property of an arrow picture object 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 in the DataWindow object or form, an arrow marks where the user is:

```
If(CurrentRow() = GetRow(), 1, 0)
```

See also

GetRow
GetRow in the *PowerScript Reference*

Date

Description

Converts a string whose value is a valid date to a value of data type date.

Syntax

Date (*string*)

Argument	Description
string	A string containing a valid date (such as Jan 1, 1998, or 12-31-99) that you want returned as a date

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.

An expression has a more limited set of data types 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 data type. To display the date without the time, choose an appropriate display format. (See"Using DataWindow painter and InfoMaker functions" on page 16.)

Examples

These expressions all return the date data type for July 4, 1994 (1994-07-04), when the default position of month is center:

```
Date("1994/07/04")
Date("1994 July 4")
Date("July 4, 1994")
```

See also

IsDate
Date in the *PowerScript Reference*

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, 1998, 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 INFO 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)
```

See also

Date
Time
DateTime in the *PowerScript Reference*

Day

Description

Obtains the day of the month in a date value.

Syntax

Day (*date*)

Argument	Description
date	The date from 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(1994-01-31)
```

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

```
Day(start_date)
```

See also

Date
IsDate
Month
Year
Day in the *PowerScript Reference*

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*.

Usage

DayName returns a name in the language of the deployment kit (DDDK) available on the machine where the application is run. If you have installed a localized deployment kit (DDDK) in the development environment or on a user's machine, then on that machine the name returned by DayName will be in the language of the localized DDDK.

FOR INFO For information about localized deployment kits, which are available in French, German, Italian, Spanish, Dutch, Danish, Norwegian, and Swedish, see the section on deploying your application in the *PowerBuilder User's Guide*.

Examples

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

```
If(DayName(date_signed) <> "Sunday", "Okay", "Invalid
Date")
```

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

```
DayName(date_signed) <> "Sunday"
```

See also

Date
Day
DayNumber
IsDate
DayName in the *PowerScript Reference*

DayNumber

Description

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

Syntax

DayNumber (*date*)

Argument	Description
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 *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:

```
If(DayNumber(start_date) > 2, "Okay", "Wrong Day")
```

This expression for a computed field returns Wrong Day if the date in end_date is a Saturday or a Sunday:

```
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:

```
DayNumber(end_date) >1 and DayNumber(end_date) < 7
```

See also

Date
Day
DayName
IsDate
DayNumber in the *PowerScript Reference*

DaysAfter

Description

Gets the number of days one date occurs after another.

Syntax

DaysAfter (*date1*, *date2*)

Argument	Description
date1	A date value that is the start date of the interval being measured
date2	A date value that is the end date of the interval

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(1995-12-20, 1995-12-24)
```

This expression returns -4:

```
DaysAfter(1995-12-24, 1995-12-20)
```

This expression returns 0:

```
DaysAfter(1995-12-24, 1995-12-24)
```

This expression returns 5:

```
DaysAfter(1994-12-29, 1995-01-03)
```

See also

Date
SecondsAfter
DaysAfter in the *PowerScript Reference*

Describe

Description

Reports the values of properties of a DataWindow object or a report or form object and objects within them. Each column and graphic object in the DataWindow object, report, or form has a set of properties, which are listed in "Objects in a DataWindow and their properties" on page 223. You specify one or more properties as a string and Describe returns the values of the properties.

Syntax

Describe (*propertylist*)

Argument	Description
propertylist	A string whose value is a blank-separated list of properties or Evaluate functions. For a list of valid properties, see "Objects in a DataWindow and their properties" on page 223.

Return value

String. Returns a string that includes a value for each property or Evaluate function. A newline 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

The values for *propertylist* can be complex. For complete information and examples, see the Describe function in the *PowerScript Reference*.

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")
```

See also

Describe in the *PowerScript Reference*

Exp

Description

Raises *e* to the specified power.

Syntax

Exp (*n*)

Argument	Description
n	The power to which you want to raise *e* (2.71828)

Return value

Double. Returns *e* raised to the power *n*.

Examples

This expression returns 7.38905609893065:

```
Exp(2)
```

See also

Log
LogTen
Exp in the *PowerScript Reference*

Fact

Description

Gets the factorial of a number.

Syntax

Fact (*n*)

Argument	Description
n	The number for which you want the factorial

Return value

Double. Returns the factorial of *n*.

Examples

This expression returns 24:

```
Fact(4)
```

Both these expressions return 1:

```
Fact(1)
Fact(0)
```

See also

Fact in the *PowerScript Reference*

Fill

Description

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

Syntax

Fill (*chars*, *n*)

Argument	Description
chars	A string whose value will be repeated to fill the return string
n	A long whose value is the length of the string you want returned

Return value

String. Returns a string *n* characters long filled with repetitions of the characters in the argument *chars*. If the argument *chars* has more than *n* characters, the first *n* characters of *chars* are used to fill the return string. If the argument *chars* has fewer than *n* characters, the characters in *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 stars:

```
Fill("*", 35)
```

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

```
Fill("-+", 7)
```

This expression returns 10 tildes (~):

```
Fill("~", 10)
```

See also

Space
Fill in the *PowerScript Reference*

First

Description

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

Syntax

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

Argument	Description
column	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
FOR *range* (optional)	The data that will be included when the value in the first row is found. 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*: ♦ CROSSTAB — (Crosstabs only) The value in the first of all rows in *column* in the crosstab For Graph and OLE objects, specify the type of object for *range*. The values to be aggregated are determined by the range specified in the object definition. (See Usage for more information.) Values are: ♦ GRAPH — (Graphs only) The value in the first row in *column* in the range specified for the Rows option of the graph ♦ OBJECT — (OLE objects only) The value in the first row in *column* in the range specified for the Rows option of the OLE object
DISTINCT (optional)	Causes First to consider only the distinct values in *column* when determining the first 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 data type of the column. Returns the value in the first row of *column*. If you specify *range*, First returns the value of the first row in *column* in *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 graphs and OLE objects, you do not select the range when you call the function. The range for the object 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 or OLE presentation style, Rows is always All.
- For Graph objects, Rows can be All, Page, or Group.
- For OLE objects, Rows can be All, Current Row, Page, or Group. The available choices depend on the layer the object 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 DataWindow painter and Report painter. To do the aggregation, a DataWindow object or a report always retrieves all rows.

Examples

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

```
First(#3 for page)
```

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

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

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

```
First(dept_id for group 2)
```

See also

Last

GetRow

Description

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

Syntax

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:

If the object in the DataWindow object or report is in this band	GetRow returns
Header	First row on the page
Group header	First row in the group
Detail	The row in which the expression occurs
Group trailer	Last row in the group
Summary	Last row in the report or DataWindow object
Footer	Last row on the page

Examples

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

```
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:

```
If(GetRow()= 0, "No Data", String(GetRow()))
```

See also

CurrentRow
GetRow in the *PowerScript Reference*

GetText

Description	Obtains the text that a user has entered in a column in a DataWindow object or form.
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:

```
Integer(GetText()) < 100
```

See also GetText in the *PowerScript Reference*

Hour

Description

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

Syntax

Hour (*time*)

Argument	Description
time	The time value from which you want the hour

Return value

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

Examples

This expression returns the current hour:

```
Hour(Now())
```

This expression returns 19:

```
Hour(19:01:31)
```

See also

Minute
Now
Second
Hour in the *PowerScript Reference*

If

Description

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

Syntax

If (*boolean, truevalue, falsevalue*)

Argument	Description
boolean	A boolean expression that evaluates to TRUE or FALSE
truevalue	A string containing the value you want returned if the boolean expression is TRUE
falsevalue	A string containing the value you want returned if the boolean expression is FALSE

Return value

The data type 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:

```
If(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:

```
If(salary > 100000, "Boss", If(salary > 12000,
"Supervisor", "Clerk"))
```

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:

```
(Number(GetText()) >= If(price >=1000, .10, .04)) AND
(Number(GetText()) <= If(price >= 1000, .20, .09))
```

The accompanying error message expression might be:

```
"Price is " + If(price >= 1000, "greater than or
equal to", "less than") + " 1000. Commission must be
between " + If(price >= 1000, ".10", ".04") + " and
" + If(price >= 1000, ".20.", ".09.")
```

See also

Case

Int

Description

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

Syntax

Int (*n*)

Argument	Description
n	The number for which you want the largest whole number that is less than or equal to it

Return value

The data type 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
Int in the *PowerScript Reference*

Integer

Description

Converts the value of a string to an integer.

Syntax

Integer (*string*)

Argument	Description
string	The string you want returned as an integer

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:

```
If (Integer(age) <> 0, age, "Not a valid age")
```

This expression returns 0:

```
Integer("3ABC")     // 3ABC is not a number
```

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

```
Integer(GetText()) < 100
```

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

```
Integer(age) <> 0
```

See also

IsNumber
Integer in the *PowerScript Reference*

IsDate

Description

Tests whether a string value is a valid date.

Syntax

IsDate (*datevalue*)

Argument	Description
datevalue	A string whose value you want to test to determine whether it is a valid date

Return value

Boolean. Returns TRUE if *datevalue* is a valid date and FALSE if it is not.

Examples

This expression returns TRUE:

```
IsDate("Jan 1, 95")
```

This expression returns FALSE:

```
IsDate("Jan 32, 1997")
```

This expression for a computed field is entered in the Computed Object property sheet. It returns the number of the day in date_received in the computed field if the column contains a valid date, and otherwise returns 0:

```
If(IsDate(String(date_received)),
DayNumber(date_received), 0)
```

See also

IsDate in the *PowerScript Reference*

IsNull

Description

Reports whether the value of a column or expression is NULL.

Syntax

IsNull (*any*)

Argument	Description
any	A column or expression that you want to test to determine whether its value is NULL

Return value

Boolean. Returns TRUE if *any* is NULL and FALSE if it is not.

Usage

Use 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:

```
IsNull(a + b)
```

This expression returns TRUE if the value in the salary column is NULL:

```
IsNull(salary)
```

This expression returns TRUE if the value the user has entered is NULL:

```
IsNull(GetText())
```

See also

IsNull in the *PowerScript Reference*

IsNumber

Description

Reports whether the value of a string is a number.

Syntax

IsNumber (*string*)

Argument	Description
string	A string whose value you want to test to determine whether it is a valid number

Return value

Boolean. Returns TRUE if *string* is a valid number and FALSE if it is not.

Examples

This expression returns TRUE:

```
IsNumber("32.65")
```

This expression returns FALSE:

```
IsNumber("A16")
```

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

```
If(IsNumber(age), age, "Not a valid age")
```

To pass this validation rule, Age_nbr must be a number:

```
IsNumber(Age_nbr) = TRUE
```

See also

Integer
IsNumber in the *PowerScript Reference*

IsRowModified

Description

Reports whether the row has been modified in the DataWindow object or form.

Syntax

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 IsRowModified 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 in a computed field in the detail area displays TRUE or FALSE to indicate whether each row has been modified:

```
IsRowModified()
```

This expression for the Color property on the Expressions property page in the Column Object property sheet displays the associated column text in red if the user has modified any value in the row:

```
If(IsRowModified(), 255, 0)
```

See also

GetRow

IsRowNew

Description

Reports whether the row has been newly inserted in the DataWindow object or form.

Syntax

IsRowNew ( )

Return value

Boolean. Returns TRUE if the row is new and FALSE if it was retrieved from the database.

Usage

In a DataWindow object, when you call IsRowNew 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 the Protect property on the Expressions property page in the Column Object property sheet prevents the user from modifying the associated column unless the row has been newly inserted:

```
If(IsRowNew(), 0, 1)
```

See also

GetRow
GetItemStatus in the *PowerScript Reference*

IsSelected

Description

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

Syntax

IsSelected ( )

Return value

Boolean. Returns TRUE if the row is selected and 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:

```
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:

```
If(IsSelected(), salary, 0)
```

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

```
Sum(cf_selected_salary for all)
```

See also

GetRow
IsSelected in the *PowerScript Reference*

IsTime

Description

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

Syntax

IsTime (*timevalue*)

Argument	Description
timevalue	A string whose value you want to test to determine whether it is a valid time

Return value

Boolean. Returns TRUE if *timevalue* is a valid time and FALSE if it is not.

Examples

This expression returns TRUE:

```
IsTime("8:00:00 am")
```

This expression returns FALSE:

```
IsTime("25:00")
```

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

```
IsTime(start_time)
```

See also

IsTime in the *PowerScript Reference*

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* {, ... } } } } })

Argument	Description
returnexp	The value you want returned when the large value is found. *Returnexp* 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
column	The column that contains the large 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 data type of *column* must be numeric
ntop	The ranking of the large value in relation to the column's largest value. For example, when *ntop* is 2, Large finds the second-largest value
FOR *range* (optional)	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 and OLE objects, specify the type of object for *range*. The values to be aggregated are determined by the range specified in the object definition. (See Usage for more information.) Values are: ♦ GRAPH — (Graphs only) The largest of values in *column* in the range specified for the Rows option of the graph ♦ OBJECT — (OLE objects only) The largest of values in *column* in the range specified for the Rows option of the OLE object

Argument	Description
DISTINCT (optional)	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 (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 data type of *returnexp*. Returns the *ntop*-largest value if it succeeds and -1 if an error occurs.

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 and OLE objects, you do not select the range when you call the function. The range for the object 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 or OLE presentation style, Rows is always All
- For Graph objects, Rows can be All, Page, or Group
- For OLE objects, Rows can be All, Current Row, Page, or Group. The available choices depend on the layer the object occupies

Max may be faster

If you don't need a return value from another column and you want to find the largest value (*ntop* = 1), use Max; 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 DataWindow painter and Report painter. To do the aggregation, a DataWindow object or report 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* {, ... } } } } })

Argument	Description
column	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.
FOR *range* (optional)	The data that will be included when the value in the last row is found. For most presentation styles, values for *range* are: ♦ ALL — (Default) The value in the last of all rows in *column* ♦ 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 ♦ PAGE — The value in the last row in *column* on a page For Crosstabs, specify CROSSTAB for *range*: ♦ CROSSTAB — (Crosstabs only) The value in the last row in *column* in the crosstab For Graph and OLE objects, specify the type of object for *range*. The values to be aggregated are determined by the range specified in the object definition. (See Usage for more information.) Values are: ♦ GRAPH — (Graphs only) The value in the last row in *column* in the range specified for the Rows option of the graph ♦ OBJECT — (OLE objects only) The value in the last row in *column* in the range specified for the Rows option of the OLE object
DISTINCT (optional)	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
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 data type 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 and OLE objects, you do not select the range when you call the function. The range for the object 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 or OLE presentation style, Rows is always All.
- For Graph objects, Rows can be All, Page, or Group.
- For OLE objects, Rows can be All, Current Row, Page, or Group. The available choices depend on the layer the object 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 DataWindow painter and Report painter. To do the aggregation, a DataWindow object or report 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

Left

Description

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

Syntax

Left (*string, n*)

Argument	Description
string	The string containing the characters you want
n	A long specifying the number of characters you want

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:

```
Left("BABE RUTH", 4)
```

This expression returns BABE RUTH:

```
Left("BABE RUTH", 40)
```

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

```
Left(home_address, 40)
```

See also

Mid
Pos
Right
Left in the *PowerScript Reference*

LeftTrim

Description

Removes spaces from the beginning of a string.

Syntax

LeftTrim (*string*)

Argument	Description
string	The string you want returned with leading spaces deleted

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
LeftTrim in the *PowerScript Reference*

Len

Description

Reports the length of a string.

Syntax

Len (*string*)

Argument	Description
string	The string for which you want the length

Return value

Long. Returns a long containing the length of *string* 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

Len in the *PowerScript Reference*

Log

Description

Gets the natural logarithm of a number.

Syntax

Log (*n*)

Argument	Description
n	The number for which you want the natural logarithm (base e). The value of *n* must be greater than 0

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 during execution:

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

See also

Exp
LogTen
Log in the *PowerScript Reference*

LogTen

Description

Gets the base 10 logarithm of a number.

Syntax

LogTen (*n*)

Argument	Description
n	The number for which you want the base 10 logarithm. The value of *n* must not be negative

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 = 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
LogTen in the *PowerScript Reference*

Long

Description

Converts the value of a string to a long.

Syntax

Long (*string*)

Argument	Description
string	The string you want returned as a long

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")
```

See also

Long in the *PowerScript Reference*

LookUpDisplay

Description

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

Syntax

LookUpDisplay (*column*)

Argument	Description
column	The column for which you want the code table display value

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 Object property sheet:

```
LookUpDisplay(product_type)
```

Lower

Description

Converts all the characters in a string to lowercase.

Syntax

Lower (*string*)

Argument	Description
string	The string you want to convert to lowercase letters

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 babe ruth:

```
Lower("Babe Ruth")
```

See also

Upper
Lower in the *PowerScript Reference*

Match

Description

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

Syntax

Match (*string, textpattern*)

Argument	Description
string	The string in which you want to look for a pattern of characters
textpattern	A string whose value is the text pattern

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:

Metacharacter	Meaning	Example
Caret (^)	Matches the beginning of a string	^C matches C at the beginning of a string
Dollar sign ($)	Matches the end of a string	s$ matches s at the end of a string
Period (.)	Matches any character	. . . matches three consecutive characters
Backslash (\)	Removes the following metacharacter's special characteristics so that it matches itself	\$ matches $

Metacharacter	Meaning	Example
Character class (a group of characters enclosed in square brackets [ ])	Matches any of the enclosed characters	[AEIOU] matches A, E, I, O, or U You can use hyphens to abbreviate ranges of characters in a character class. For example, [A-Za-z] matches any letter
Complemented character class (first character inside the brackets is a caret)	Matches any character *not* in the group following the caret	[^0-9] matches any character except a digit, and [^A-Za-z] matches any character except a letter

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

Metacharacter	Meaning	Example
* (asterisk)	Indicates zero or more occurrences	A* matches zero or more As (no As, A, AA, AAA, and so on)
+ (plus)	Indicates one or more occurrences	A+ matches one A or more than one A (A, AAA, and so on)
? (question mark)	Indicates zero or one occurrence	A? matches an empty string ("") or A

Sample patterns The following table shows various text patterns and sample text that matches each pattern:

This pattern	Matches
AB	Any string that contains AB; such as ABA, DEABC, graphAB_one
B*	Any string that contains 0 or more Bs; such as AC, B, BB, BBB, ABBBC, and so on
AB*C	Any string containing the pattern AC or ABC or ABBC, and so on (0 or more Bs)
AB+C	Any string containing the pattern ABC or ABBC or ABBBC, and so on (1 or more Bs)
ABB*C	Any string containing the pattern ABC or ABBC or ABBBC, and so on (1 B plus 0 or more Bs)

This pattern	Matches
^AB	Any string starting with AB
AB?C	Any string containing the pattern AC or ABC (0 or 1 B)
^[ABC]	Any string starting with A, B, or C
[^ABC]	A string containing any characters other than A, B, or C
^[^abc]	A string that begins with any character except a, b, or c
^[^a-z]$	Any single-character string that is not a lowercase letter (^ and $ indicate the beginning and end of the string)
[A-Z]+	Any string with one or more uppercase letters
^[0-9]+$	Any string consisting only of digits
^[0-9][0-9][0-9]$	Any string consisting of exactly three digits
^([0-9][0-9][0-9])$	Any consisting of exactly three digits enclosed in parentheses

Examples

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

```
Match(GetText(), "^[A-Z]")
```

See also

Pos
Match in the *PowerScript Reference*

Max

Description

Gets the maximum value in the specified column.

Syntax

Max (*column* { FOR *range* { DISTINCT { *expres1* {, *expres2* {, ... } } } } })

Argument	Description
column	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 data type of *column* must be numeric
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 For Graph and OLE objects, specify the type of object for *range*. The values to be aggregated are determined by the range specified in the object definition. (See Usage for more information.) Values are: ♦ GRAPH — (Graphs only) The maximum value in *column* in the range specified for the Rows option of the graph ♦ OBJECT — (OLE objects only) The maximum value in *column* in the range specified for the Rows option of the OLE object
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 data type 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 and OLE objects, you do not select the range when you call the function. The range for the object 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 or OLE presentation style, Rows is always All.
- For Graph objects, Rows can be All, Page, or Group.
- For OLE objects, Rows can be All, Current Row, Page, or Group. The available choices depend on the layer the object occupies.

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 DataWindow painter and Report painter. To do the aggregation, a DataWindow object or report 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, report, or form 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)
```

See also

Min
Max in the *PowerScript Reference*

Median

Description

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 data type 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 and OLE objects, specify the type of object for *range*. The values to be aggregated are determined by the range specified in the object definition. (See Usage for more information.) Values are: ♦ GRAPH — (Graphs only) The median of values in *column* in the range specified for the Rows option of the graph ♦ OBJECT — (OLE objects only) The median of values in *column* in the range specified for the Rows option of the OLE object
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 data type 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*.

For graphs and OLE objects, you do not select the range when you call the function. The range for the object 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 or OLE presentation style, Rows is always All.
- For Graph objects, Rows can be All, Page, or Group.
- For OLE objects, Rows can be All, Current Row, Page, or Group. The available choices depend on the layer the object occupies.

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.

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

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 Object 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, report, or form 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:

```
Median(order_amt for all DISTINCT order_nbr)
```

See also

Avg
Mode

Mid

Description

Obtains a specified number of characters from a specified position in a string.

Syntax

Mid (*string*, *start* {, *length* })

Argument	Description
string	The string from which you want characters returned
start	A long specifying the position of the first character you want returned (the position of the first character of the string is 1)
length (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

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 "":

```
Mid("BABE RUTH", 40, 5)
```

This expression returns BE RUTH:

```
Mid("BABE RUTH", 3)
```

This expression in a computed field returns ACCESS DENIED if the fourth character in the column password is not R:

```
If(Mid(password, 4, 1) = "R", "ENTER", "ACCESS
DENIED")
```

To pass this validation rule the fourth character in the column password must be 6:

```
Mid(password, 4, 1) = "6"
```

See also

Mid in the *PowerScript Reference*

Min

Description

Gets the minimum value in the specified column.

Syntax

Min (*column* { FOR *range* { DISTINCT { *expres1* {, *expres2* {, ... } } } } })

Argument	Description
column	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 data type of *column* must be numeric
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 and OLE objects, specify the type of object for *range*. The values to be aggregated are determined by the range specified in the object definition. (See Usage for more information.) Values are: ♦ GRAPH — (Graphs only) The minimum of values in *column* in the range specified for the Rows option of the graph ♦ OBJECT — (OLE objects only) The minimum of values in *column* in the range specified for the Rows option of the OLE object
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

Return value

The data type 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 and OLE objects, you do not select the range when you call the function. The range for the object 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 or OLE presentation style, Rows is always All.
- For Graph objects, Rows can be All, Page, or Group.
- For OLE objects, Rows can be All, Current Row, Page, or Group. The available choices depend on the layer the object occupies.

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 DataWindow painter and Report painter. To do the aggregation, a DataWindow object or report always retrieves all rows.

Examples

This expression returns the minimum value in the column named age in group 2:

```
Min(age for group 2)
```

This expression returns the minimum of the values in column 3 on the page:

```
Min(#3 for page)
```

Assuming a DataWindow object, report, or form 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:

```
Min(order_amt for all DISTINCT order_nbr)
```

See also

Max
Min in the *PowerScript Reference*

Minute

Description

Obtains the number of minutes in the minutes portion of a time value.

Syntax

Minute (*time*)

Argument	Description
time	The time value from which you want the minutes

Return value

Integer. Returns the minutes portion of *time* (00 to 59).

Examples

This expression returns 1:

```
Minute(19:01:31)
```

See also

Hour
Second
Minute in the *PowerScript Reference*

Mod

Description

Obtains the remainder (modulus) of a division operation.

Syntax

Mod (*x*, *y*)

Argument	Description
x	The number you want to divide by *y*
y	The number you want to divide into *x*

Return value

The data type of *x* or *y*, whichever data type is more precise.

Examples

This expression returns 2:

```
Mod(20, 6)
```

This expression returns 1.5:

```
Mod(25.5, 4)
```

This expression returns 2.5:

```
Mod(25, 4.5)
```

See also

Mod in the *PowerScript Reference*

Mode

Description

Calculates the mode of the values of the column. The mode is the most frequently occurring value.

Syntax

Mode (*column* { FOR *range* { DISTINCT { *expres1* {, *expres2* {, ... } } } } })

Argument	Description
column	The column for which you want the mode 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 data type of *column* must be numeric
FOR *range* (optional)	The data that will be included in the mode. For most presentation styles, values for *range* are: ♦ ALL — (Default) The mode of all values in *column* ♦ 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 ♦ PAGE — The mode of the values in *column* on a page For Crosstabs, specify CROSSTAB for *range*: ♦ CROSSTAB — (Crosstabs only) The mode of all values in *column* in the crosstab For Graph and OLE objects, specify the type of object for *range*. The values to be aggregated are determined by the range specified in the object definition. (See Usage for more information.) Values are: ♦ GRAPH — (Graphs only) The mode of values in *column* in the range specified for the Rows option of the graph ♦ OBJECT — (OLE objects only) The mode of values in *column* in the range specified for the Rows option of the OLE object
DISTINCT (optional)	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
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 data type 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 and OLE objects, you do not select the range when you call the function. The range for the object 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 or OLE presentation style, Rows is always All.
- For Graph objects, Rows can be All, Page, or Group.
- For OLE objects, Rows can be All, Current Row, Page, or Group. The available choices depend on the layer the object occupies.

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 DataWindow painter and Report painter. To do the aggregation, a DataWindow object or report 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 Object 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, report, or form 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

Month (*date*)

Argument	Description
date	The date from which you want the month

Return value

Integer. Returns an integer (1 to 12) whose value is the month portion of *date*.

Examples

This expression returns 1:

```
Month(1994-01-31)
```

This expression for a computed column returns Wrong Month if the month in the column expected_grad_date is not 6:

```
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 expected_grad_date is 6:

```
Month(expected_grad_date) = 6
```

See also

Day
Date
Year
Month in the *PowerScript Reference*

Now

Description

Obtains the current time based on the system time of the client machine.

Syntax

Now ( )

Return value

Time. Returns the current time based on the system time of the client machine.

Usage

Use 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, report, or form determines the frequency at which the value of Now is updated. For example, if the timer interval is 1 second, it is updated every second.

Examples

This expression returns the current system time:

```
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:

```
If(Now() < 08:00:00, '08:00:00', String(Now()))
```

See also

If
Year
Now in the *PowerScript Reference*

Number

Description

Converts a string to a number.

Syntax

Number (*string*)

Argument	Description
string	The string you want returned as a number

Return value

A numeric data type. Returns the contents of *string* as a number. If *string* is not a valid number, Number returns 0.

Examples

This expression converts the string 24 to a number:

```
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:

```
If(Number(age) > 55, "N/A", age)
```

This validation rule checks that the number the user entered is between 25,000 and 50,000:

```
Number(GetText())>25000 AND Number (GetText())<50000
```

Page

Description

Gets the number of the current page.

Syntax

Page ( )

Return value

Integer. 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 of the paper.

Examples

This expression returns the number of the current page:

```
Page()
```

In the DataWindow object or report'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

PageAcross
PageCount
PageCountAcross

PageAcross

Description

Gets the number of the current horizontal page. For example, if a report is twice the width of the preview window and the window is scrolled horizontally to display the portion of the report that was outside the preview, PageAcross will return 2 because the current page is the second horizontal page.

Syntax

PageAcross ( )

Return value

Integer. Returns the number of the current horizontal page if it succeeds and -1 if an error occurs.

Examples

This expression returns the number of the current horizontal page:

```
PageAcross ( )
```

See also

Page
PageCount
PageCountAcross

PageCount

Description

Gets the total number of pages when viewing a DataWindow object in Print Preview or when previewing a report. This number is also the number of printed pages if the DataWindow object or report is not wider than the preview window. If the DataWindow object or report is wider than the preview window, the number of printed pages will be greater than PageCount gets.

Syntax

PageCount ( )

Return value

Integer. Returns the total number of pages.

Usage

PageCount applies to Print Preview in the DataWindow painter and Preview in the Report painter. PageCount does not apply to Preview in the DataWindow painter.

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 of the paper.

Examples

This expression returns the number of pages:

```
PageCount()
```

In the DataWindow object or report'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 when viewing a DataWindow object (in Print Preview) or report (in Preview) that is wider than the preview window.

Syntax

PageCountAcross ( )

Return value

Integer. Returns the total number of horizontal pages if it succeeds and -1 if an error occurs.

Usage

PageCountAcross applies to Print Preview in the DataWindow painter and Preview in the report painter. PageCountAcross does not apply to Preview in the DataWindow painter.

Examples

This expression returns the number of horizontal pages in the preview window:

```
PageCountAcross ( )
```

See also

Page
PageAcross
PageCount

Percent

Description

Gets the percentage that the current value is of the total of the values in the column.

Syntax

Percent (*column* { FOR *range* { DISTINCT { *expres1* {, *expres2* {, ... } } } } })

Argument	Description
column	The column for which you want the value of each row as a percentage of the total of the values of the column. *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 data type of *column* must be numeric
FOR *range* (optional)	The data that will be included in the percentage. For most presentation styles, values for *range* are: ♦ ALL — (Default) The percentage that the current value is of all rows in *column* ♦ GROUP *n* — The percentage that the current value is of rows in *column* in the specified group. Specify the keyword GROUP followed by the group number: for example, GROUP 1 ♦ PAGE — The percentage that the current value is of the rows in *column* on a page For Crosstabs, specify CROSSTAB for *range*: ♦ CROSSTAB — (Crosstabs only) The percentage that the current value is of all rows in *column* in the crosstab For Graph and OLE objects, specify the type of object for *range*. The values to be aggregated are determined by the range specified in the object definition. (See Usage for more information.) Values are: ♦ GRAPH — (Graphs only) The percentage that the current value is of values in *column* in the range specified for the Rows option of the graph ♦ OBJECT — (OLE objects only) The percentage that the current value is of values in *column* in the range specified for the Rows option of the OLE object
DISTINCT (optional)	Causes Percent to consider only the distinct values in *column* when determining the percentage. For a value of *column*, the first row found with the value is used and other rows that have the same value are ignored

Argument	Description
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

A numeric data type (decimal, double, integer, long, or real). Returns the percentage the current row of *column* is of the total value of the column.

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, it 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 the current row of *column* 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 the distinct value in *column* is of the total value of *column*, or if you specify *expresn*, the percent the value in *column* in the row in which the value of *expresn* is distinct is of the total for *column*.

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 and OLE objects, you do not select the range when you call the function. The range for the object 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 or OLE presentation style, Rows is always All.
- For Graph objects, Rows can be All, Page, or Group.
- For OLE objects, Rows can be All, Current Row, Page, or Group. The available choices depend on the layer the object occupies.

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 DataWindow painter and Report painter. To do the aggregation, a DataWindow object or report 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:

```
Percent(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:

```
Percent(qty_ordered for graph)
```

Assuming a DataWindow object, report, or form 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:

```
Percent(order_amt for all DISTINCT order_nbr)
```

See also

CumulativePercent

Pi

Description

Multiplies pi by a specified number.

Syntax

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 Pi to convert angles to and from radians.

Examples

This expression returns pi:

```
Pi(1)
```

Both these expressions return the area of a circle with the radius Rad:

```
Pi(1) * Rad^2
Pi(Rad^2)
```

This expression computes the cosine of a 45-degree angle:

```
Cos(45.0 * (Pi(2)/360))
```

See also

Cos
Sin
Tan
Pi in the *PowerScript Reference*

Pos

Description

Finds one string within another string.

Syntax

Pos (*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 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 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

Left
Mid
Right
Pos in the *PowerScript Reference*

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 will return 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

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.

You can use PowerScript SetProfileString to change values in the profile file to customize your application's configuration during execution. Before you make changes, you can use ProfileInt and ProfileString to obtain the original settings so you can optionally restore them when the user exits the application.

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
ProfileInt in the *PowerScript Reference*

ProfileString

Description

Obtains the string value of a setting in the specified profile file.

Syntax

ProfileString (*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, ProfileString 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	A string value that ProfileString will return 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

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.

You can use the PowerScript SetProfileString function to change values in the profile file to customize your application's configuration during execution. Before you make changes, you can use ProfileInt and ProfileString to obtain the original settings so you can optionally restore them when the user exits the application.

Examples

This example uses the following section in 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
ProfileString in the *PowerScript Reference*
SetProfileString in the *PowerScript Reference*

Rand

Description

Obtains a random whole number between 1 and a specified upper limit.

Syntax

Rand (*n*)

Argument	Description
n	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

Return value

A numeric data type, the data type 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. You can control whether the sequence is different each time your application runs by calling the PowerScript Randomize function to initialize the random number generator.

Examples

This expression returns a random whole number between 1 and 10:

```
Rand(10)
```

See also

Rand in the *PowerScript Reference*
Randomize in the *PowerScript Reference*

Real

Description

Converts a string value to a real data type.

Syntax

Real (*string*)

Argument	Description
string	The string whose value you want to convert to a real

Return value

Real. Returns the contents of a string as a real. If *string* is not a valid number, Real returns 0.

Examples

This expression converts 24 to a real:

```
Real("24")
```

This expression returns the value in the column temp_text as a real:

```
Real(temp_text)
```

See also

Real in the *PowerScript Reference*

RelativeDate

Description

Obtains the date that occurs a specified number of days after or before another date.

Syntax

RelativeDate (*date*, *n*)

Argument	Description
date	A date value
n	An integer indicating the number of days

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 1990-02-10:

```
RelativeDate(1990-01-31, 10)
```

This expression returns 1990-01-21:

```
RelativeDate(1990-01-31, -10)
```

See also

DaysAfter
RelativeDate in the *PowerScript Reference*

RelativeTime

Description

Obtains a time that occurs a specified number of seconds after or before another time within a 24-hour period.

Syntax

RelativeTime (*time*, *n*)

Argument	Description
time	A time value
n	A long number of seconds

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:

```
RelativeTime(19:01:31, 10)
```

This expression returns 19:01:21:

```
RelativeTime(19:01:31, -10)
```

See also

SecondsAfter
RelativeTime in the *PowerScript Reference*

Replace

Description

Replaces a portion of one string with another.

Syntax

Replace (*string1, start, n, string2*)

Argument	Description
string1	The string in which you want to replace characters with *string2*
start	A long whose value is the number of the first character you want replaced. (The first character in the string is number 1)
n	A long whose value is the number of characters you want to replace
string2	The string that will replace 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

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 BABY RUTH:

```
Replace("BABE RUTH", 1, 4, "BABY")
```

This expression returns Closed for the Winter:

```
Replace("Closed for Vacation", 12, 8, "the Winter")
```

See also

Replace in the *PowerScript Reference*

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*)

Argument	Description
red	The integer value of the red component of the color
green	The integer value of the green component of the color
blue	The integer value of the blue component of the color

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:

Red + (256 * Green) + (65536 * 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:

Color	Red value	Green value	Blue value
Black	0	0	0
White	255	255	255
Light Gray	192	192	192
Dark Gray	128	128	128
Red	255	0	0
Dark Red	128	0	0
Green	0	255	0

Color	Red value	Green value	Blue value
Dark Green	0	128	0
Blue	0	0	255
Dark Blue	0	0	128
Magenta	255	0	255
Dark Magenta	128	0	128
Cyan	0	255	255
Dark Cyan	0	128	128
Yellow	255	255	0
Brown	128	128	0

Examples

This expression returns as a long 8421376, which represents dark cyan:

```
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:

```
If(salary>50000, RGB(255,0,0), RGB(255,255,255))
```

See also

RGB in the *PowerScript Reference*

Right

Description

Obtains a specified number of characters from the end of a string.

Syntax

Right (*string, n*)

Argument	Description
string	The string from which you want characters returned
n	A long whose value is the number of characters you want returned from the right end of *string*

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 RUTH:

```
Right("BABE RUTH", 4)
```

This expression returns BABE RUTH:

```
Right("BABE RUTH", 75)
```

See also

Left
Mid
Pos
Right in the *PowerScript Reference*

RightTrim

Description

Removes spaces from the end of a string.

Syntax

RightTrim (*string*)

Argument	Description
string	The string you want returned with trailing blanks deleted

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
RightTrim in the *PowerScript Reference*

Round

Description

Rounds a number to the specified number of decimal places.

Syntax

Round (*x* , *n*)

Argument	Description
x	The number you want to round
n	The number of decimal places to which you want to round *x*

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.

Examples

This expression returns 9.62:

```
Round(9.624, 2)
```

This expression returns 9.63:

```
Round(9.625, 2)
```

This expression returns 9.600:

```
Round(9.6, 3)
```

This expression returns -9.63:

```
Round(-9.625, 2)
```

This expression returns -10:

```
Round(-9.625, -1)
```

See also

Ceiling
Int
Truncate
Round in the *PowerScript Reference*

RowCount

Description

Obtains the number of rows that are currently available in the primary buffer.

Syntax

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:

```
If(RowCount() = 0, "No Data", String(RowCount()))
```

See also

RowCount in the *PowerScript Reference*

RowHeight

Description	Reports the height of a row associated with a band in a DataWindow object or a report.
Syntax	**RowHeight** ( )
Return value	Long. Returns the height of the row in the units specified for the DataWindow object or report if it succeeds, and -1 if an error occurs.
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.
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*)

Argument	Description
time	The time value from which you want the seconds

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
Second in the *PowerScript Reference*

SecondsAfter

Description

Gets the number of seconds one time occurs after another.

Syntax

SecondsAfter (*time1*, *time2*)

Argument	Description
time1	A time value that is the start time of the interval being measured
time2	A time value that is the end time of the interval

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
SecondsAfter in the *PowerScript Reference*

Sign

Description

Reports whether the number is negative, zero, or positive by checking its sign.

Syntax

Sign (*n*)

Argument	Description
n	The number for which you want to determine the sign

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)
```

See also

Sign in the *PowerScript Reference*

Sin

Description

Calculates the sine of an angle.

Syntax

Sin (*n*)

Argument	Description
n	The angle (in radians) for which you want the sine

Return value

Double. Returns 1 if it succeeds and -1 if an error occurs.

Examples

This expression returns .8414709848078965:

```
Sin(1)
```

This expression returns 0:

```
Sin(0)
```

This expression returns 0:

```
Sin(pi(1))
```

See also

Cos
Pi
Tan
Sin in the *PowerScript Reference*

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

Small (*returnexp*, *column*, *nbottom* { FOR *range* { DISTINCT { *expres1* {, *expres2* {, ... } } } } })

Argument	Description
returnexp	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
column	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 data type of *column* must be numeric
nbottom	The relationship of the small value to the column's smallest value. For example, when *nbottom* is 2, Small finds the second-smallest value
FOR *range* (optional)	The data that will be included when finding the small value. For most presentation styles, values for *range* are: ♦ ALL — (Default) The small value of all rows in *column* ♦ 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 ♦ PAGE — The small value of the rows in *column* on a page For Crosstabs, specify CROSSTAB for *range*: ♦ CROSSTAB — (Crosstabs only) The small value of all rows in *column* in the crosstab For Graph and OLE objects, specify the type of object for *range*. The values to be aggregated are determined by the range specified in the object definition. (See Usage for more information.) Values are: ♦ GRAPH — (Graphs only) The small value in *column* in the range specified for the Rows option of the graph ♦ OBJECT — (OLE objects only) The small value in *column* in the range specified for the Rows option of the OLE object

Argument	Description
DISTINCT (optional)	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
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 data type of *returnexp*. Returns the *nbotton*-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 and OLE objects, you do not select the range when you call the function. The range for the object 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 or OLE presentation style, Rows is always All.
- For Graph objects, Rows can be All, Page, or Group.
- For OLE objects, Rows can be All, Current Row, Page, or Group. The available choices depend on the layer the object occupies.

Min may be faster If you don't need a return value from another column and you want to find the smallest value (*nbottom* = 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 DataWindow painter and Report painter. To do the aggregation, a DataWindow object or report 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)
```

See also

Fill
Space in the *PowerScript Reference*

Sqrt

Description

Calculates the square root of a number.

Syntax

Sqrt (*n*)

Argument	Description
n	The number for which you want the square root

Return value

Double. Returns the square root of *n*.

Usage

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:

```
Sqrt(2)
```

This expression results in an error at execution time:

```
Sqrt(-2)
```

See also

Sqrt in the *PowerScript Reference*

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

StDev (*column* { FOR *range* { DISTINCT { *expres1* {, *expres2* {, ... } } } } })

Argument	Description
column	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 data type of *column* must be numeric
FOR *range* (optional)	The data that will 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*: ♦ CROSSTAB — (Crosstabs only) The estimate of the standard deviation for all values in *column* in the crosstab For Graph and OLE objects, specify the type of object for *range*. The values to be aggregated are determined by the range specified in the object definition. (See Usage for more information.) Values are: ♦ GRAPH — (Graphs only) The estimate of the standard deviation for values in *column* in the range specified for the Rows option of the graph ♦ OBJECT — (OLE objects only) The estimate of the standard deviation for values in *column* in the range specified for the Rows option of the OLE object
DISTINCT (optional)	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
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

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 and OLE objects, you do not select the range when you call the function. The range for the object 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:

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

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 DataWindow painter and Report painter. To do the aggregation, a DataWindow object or report 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:

```
StDev(salary)
```

This expression returns an estimate for standard deviation of the values in the column named salary in group 1:

```
StDev(salary for group 1)
```

This expression returns an estimate for standard deviation of the values in column 4 on the page:

```
StDev(#4 for page)
```

This expression entered in the Value box on the Data tab page in the Graph Object property sheet returns an estimate for standard deviation of the values in qty_used column in the graph:

```
StDev(qty_used for graph)
```

This expression for a computed field in a crosstab returns the estimate for standard deviation of the values in qty_ordered column in the crosstab:

```
StDev(qty_ordered for crosstab)
```

Assuming a DataWindow object, report, or form 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:

```
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

StDevP (*column* { FOR *range* { DISTINCT { *expres1* {, *expres2* {, ... } } } } })

Argument	Description
column	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 data type of *column* must be numeric
FOR *range* (optional)	The data that will be included in the standard deviation. For most presentation styles, values for *range* are: ♦ ALL — (Default) The standard deviation for all values in *column* ♦ 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 ♦ PAGE — The standard deviation for the values in *column* on a page For Crosstabs, specify CROSSTAB for *range*: ♦ CROSSTAB — (Crosstabs only) The standard deviation for all values in *column* in the crosstab For Graph and OLE objects, specify the type of object for *range*. The values to be aggregated are determined by the range specified in the object definition. (See Usage for more information.) Values are: ♦ GRAPH — (Graphs only) The standard deviation for values in *column* in the range specified for the Rows option of the graph ♦ OBJECT — (OLE objects only) The standard deviation for values in *column* in the range specified for the Rows option of the OLE object
DISTINCT (optional)	Causes StDevP 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
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

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, 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 and OLE objects, you do not select the range when you call the function. The range for the object 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:

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

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 DataWindow painter and Report painter. To do the aggregation, a DataWindow object or report 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:

```
StDevP(salary)
```

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

```
StDevP(salary for group 1)
```

This expression returns the standard deviation of the values in column 4 on the page:

```
StDevP(#4 for page)
```

This expression entered in the Value box on the Data tab page in the Graph Object property sheet returns the standard deviation of the values in qty_ordered column in the graph:

```
StDevP(qty_ordered for graph)
```

This expression for a computed field in a crosstab returns the standard deviation of the values in qty_ordered column in the crosstab:

```
StDevP(qty_ordered for crosstab)
```

Assuming a DataWindow object, report, or form 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:

```
StDevP(order_amt for all DISTINCT order_nbr)
```

See also

StDev
VarP

String

Description

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* })

Argument	Description
data	The data you want returned as a string with the specified formatting. *Data* can have a date, DateTime, numeric, time, or string data type
format (optional)	A string of the display masks you want to use to format the data. The masks consist of formatting information specific to the data type of *data*. If *data* is type string, *format* is required The format string can consist of more than one mask, depending on the data type of *data*. Each mask is separated by a semicolon. See Usage for details on each data type

Return value

String. Returns *data* in the specified format if it succeeds and the empty string ("") if the data type 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 don't 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, which are displayed in the DataWindow object, report, or form. When you use the String function in a PowerBuilder script, rather than in the DataWindow painter, color specifications have no effect.

If the display format doesn't match the data type, PowerBuilder and InfoMaker will try to apply the mask, which can produce unpredictable results.

FOR INFO For information on specifying display formats, see the *PowerBuilder User's Guide.*

When you use String to format a date and the month is displayed as text (for example, when the display format includes "mmm"), the month is in the language of the deployment kit (DDDK) available when the application is run. If you have installed a localized DDDK in the development environment or on a user's machine, then on that machine the month in the resulting string will be in the language of the localized DDDK.

FOR INFO For information about localized deployment kits, which are available in French, German, Italian, Spanish, Dutch, Danish, Norwegian, and Swedish, see "The deployed application" on page 650the section on deploying your application in the *PowerBuilder User's Guide.*

Examples

This expression returns Jan 31, 1998:

```
String(1998-01-31, "mmm dd, yyyy")
```

This expression returns Jan 31, 1998, 6 hrs and 8 min:

```
String(1998-01-31 06:08:00, 'mmm dd, yyyy, h "hrs
and" m "min"')
```

This expression:

```
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:

```
String("ABC", "@-@-@")
```

This expression returns A*B:

```
String("ABC", "@*@")
```

This expression returns ABC:

```
String("ABC", "@@@")
```

This expression returns a space:

```
String("ABC", " ")
```

This expression returns 6 hrs and 8 min:

```
String(06:08:02,'h "hrs and" m "min"')
```

This expression returns 08:06:04 pm:

```
String(20:06:04, "hh:mm:ss am/pm")
```

This expression returns 8:06:04 am:

```
String(08:06:04, "h:mm:ss am/pm")
```

This expression returns 6:11:25.300000:

```
String(6:11:25.300000, "h:mm:ss.ffffff")
```

See also

String in the *PowerScript Reference*

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 data type of *column* must be numeric
FOR *range* (optional)	The data that will 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 and OLE objects, specify the type of object for *range*. The values to be aggregated are determined by the range specified in the object definition. (See Usage for more information.) Values are: ♦ GRAPH — (Graphs only) The sum of values in *column* in the range specified for the Rows option of the graph ♦ OBJECT — (OLE objects only) The sum of values in *column* in the range specified for the Rows option of the OLE object
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 data type. 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 and OLE objects, you do not select the range when you call the function. The range for the object 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 or OLE presentation style, Rows is always All.
- For Graph objects, Rows can be All, Page, or Group.
- For OLE objects, Rows can be All, Current Row, Page, or Group. The available choices depend on the layer the object occupies.

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 DataWindow painter and Report painter. To do the aggregation, a DataWindow object or report always retrieves all rows.

Examples

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

```
Sum(salary for group 1)
```

This expression returns the sum of the values in column 4 on the page:

```
Sum(#4 for page)
```

Assuming a DataWindow object, report, or form 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:

```
Sum(order_amt for all DISTINCT order_nbr)
```

Tan

Description

Calculates the tangent of an angle.

Syntax

Tan (*n*)

Argument	Description
n	The angle (in radians) for which you want the tangent

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
Tan in the *PowerScript Reference*

Time

Description

Converts a string to a time data type.

Syntax

Time (*string*)

Argument	Description
string	A string containing a valid time (such as 8 AM or 10:25) that you want returned as a time data type. 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

Return value

Time. Returns the time in *string* as a time data type. If *string* does not contain a valid time, Time returns 00:00:00.

Examples

This expression returns the time data type 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))
```

See also

Time in the *PowerScript Reference*

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.

For PowerBuilder, the PowerScript and DataWindow painter versions of the Today function have different data types. The return value of the PowerScript Today function is date.

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
Today in the *PowerScript Reference*

Trim

Description

Removes leading and trailing spaces from a string.

Syntax

Trim (*string*)

Argument	Description
string	The string you want returned with leading and trailing spaces deleted

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 may have typed before or after newly entered data.

Examples

This expression returns BABE RUTH:

```
Trim(" BABE RUTH ")
```

See also

LeftTrim
RightTrim
Trim in the *PowerScript Reference*

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*

Return value

The data type 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
Truncate in the *PowerScript Reference*

Upper

Description

Converts all characters in a string to uppercase letters.

Syntax

Upper (*string*)

Argument	Description
string	The string you want to convert to uppercase letters

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
Upper in the *PowerScript Reference*

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* {, ... } } } } })

Argument	Description
column	The column for which 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 data type of *column* must be numeric
FOR *range* (optional)	The data that will be included in the estimate of the variance. For most presentation styles, values for *range* are: ♦ ALL — (Default) The estimate of the variance for all rows in *column* ♦ GROUP *n* — The estimate of the variance for rows in *column* in the specified group. Specify the keyword GROUP followed by the group number: for example, GROUP 1 ♦ PAGE — The estimate of the variance for the rows in *column* on a page For Crosstabs, specify CROSSTAB for *range*: ♦ CROSSTAB — (Crosstabs only) The estimate of the variance for all rows in *column* in the crosstab For Graph and OLE objects, specify the type of object for *range*. The values to be aggregated are determined by the range specified in the object definition. (See Usage for more information.) Values are: ♦ GRAPH — (Graphs only) The estimate of the variance for rows in *column* in the range specified for the Rows option of the graph ♦ OBJECT — (OLE objects only) The estimate of the variance for rows in *column* in the range specified for the Rows option of the OLE object
DISTINCT (optional)	Causes Var to consider only the distinct values in *column* when determining the variance. 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

Double. Returns an estimate for the variance for *column*. If you specify *group*, Var returns an estimate for the variance for *column* within *group*.

Usage

If you specify *range*, Var returns an estimate for the variance for *column* within *range*. If you specify DISTINCT, Var returns the variance for the distinct values in *column* or if you specify *expresn*, the estimate for the variance of the rows 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 for the object 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 or OLE presentation style, Rows is always All.
- For Graph objects, Rows can be All, Page, or Group.
- For OLE objects, Rows can be All, Current Row, Page, or Group. The available choices depend on the layer the object occupies.

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 DataWindow painter and Report painter. To do the aggregation, a DataWindow object or report always retrieves all rows.

Examples

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:

```
Var(salary)
```

This expression returns an estimate for the variance of the values in the column named salary in group 1:

```
Var(salary for group 1)
```

This expression entered in the Value box on the Data property page in the Graph Object property sheet returns an estimate for the variance of the values in quantity column in the graph:

```
Var(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:

```
Var(quantity for crosstab)
```

Assuming a DataWindow object, report, or form 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:

```
Var(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

VarP (*column* { FOR *range* { DISTINCT { *expres1* {, *expres2* {, ... } } } } })

Argument	Description
column	The column for which for which you want 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 data type of *column* must be numeric
FOR *range* (optional)	The data that will be included in the variance. For most presentation styles, values for *range* are: ♦ ALL — (Default) The variance for all rows in *column* ♦ GROUP *n* — The variance for rows in *column* in the specified group. Specify the keyword GROUP followed by the group number: for example, GROUP 1 ♦ PAGE — The variance for the rows in *column* on a page For Crosstabs, specify CROSSTAB for *range*: ♦ CROSSTAB — (Crosstabs only) The variance for all rows in *column* in the crosstab For Graph and OLE objects, specify the type of object for *range*. The values to be aggregated are determined by the range specified in the object definition. (See Usage for more information.) Values are: ♦ GRAPH — (Graphs only) The variance for rows in *column* in the range specified for the Rows option of the graph ♦ OBJECT — (OLE objects only) The variance for rows in *column* in the range specified for the Rows option of the OLE object
DISTINCT (optional)	Causes VarP to consider only the distinct values in *column* when determining the variance. 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

Double. Returns the variance for *column*. If you specify *group*, Var returns the variance for *column* within *range*.

Usage

If you specify *range*, VarP returns the variance for *column* within *range*. If you specify DISTINCT, VarP returns the variance for the distinct values in *column* or if you specify *expresn*, the variance of the rows 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 for the object 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 or OLE presentation style, Rows is always All.
- For Graph objects, Rows can be All, Page, or Group.
- For OLE objects, Rows can be All, Current Row, Page, or Group. The available choices depend on the layer the object occupies.

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 DataWindow painter and Report painter. To do the aggregation, a DataWindow object or report 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:

```
VarP(salary)
```

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

```
VarP(salary for group 1)
```

This expression returns the variance of the values in column 4 on the page:

```
VarP(#4 for page)
```

This expression entered in the Value box on the Data property page in the Graph Object property sheet returns the variance of the values in quantity column in the graph:

```
VarP(quantity for graph)
```

This expression for a computed field in a crosstab returns the variance of the values in quantity column in the crosstab:

```
VarP(quantity for crosstab)
```

Assuming a DataWindow object, report, or form 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:

```
VarP(order_amt for all DISTINCT order_nbr)
```

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

WordCap (*string*)

Argument	Description
string	A string or expression that evaluates to a string that you want to display with initial capital letters (for example, Monday Morning)

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:

```
WordCap("boston, MASSACHUSETTS")
```

This expression concatenates the characters in the emp_fname and emp_lname columns and makes the first letter of each word uppercase:

```
WordCap(emp_fname + " " + emp_lname)
```

Year

Description

Gets the year of a date value.

Syntax

Year (*date*)

Argument	Description
date	The date value from which you want the year

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 PowerBuilder and InfoMaker choose the century as follows. If the year is between 00 to 49, PowerBuilder and InfoMaker assume 20 as the first two digits; if it is between 50 and 99, PowerBuilder and InfoMaker assume 19.

Usage

Obtains the year portion of *date*. PowerBuilder and InfoMaker handle years from 1000 to 3000 inclusive.

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.

Examples

This expression returns 1995:

```
Year(1995-01-31)
```

See also

Day
Month
Year in the *PowerScript Reference*

CHAPTER 3

Accessing Data in Scripts

About this chapter

This chapter explains the syntax for constructing expressions that access data in a DataWindow control.

Contents

Methods for accessing data

Two methods

There are two ways to access data values in a DataWindow control:

- **Functions** SetItem and the group of GetItem functions access single values in specific rows and columns. For example:

```
dw_1.SetItem(1, "empname", "Phillips")
ls_name = dw_1.GetItemString(1, "empname")
```

- **Expressions** DataWindow data expressions use dot notation and can refer to single items, columns, blocks of data, selected data, or the whole DataWindow control. For example:

```
dw_1.Object.empname[1] = "Phillips"
dw_1.Object.Data[1,1] = "Phillips"
```

Both methods allow you to access data in any buffer and to get original or current values.

Which method to use

The method you use depends on how much data you are accessing and whether you know the names of the DataWindow columns when the script is compiled:

If you want to access	Use
A single item	Either an expression or a function. Both are equally efficient when referring to single items **Exception** If you want to use a column name rather than its number, and the name is not known until runtime, use a function; functions allow you to name the column dynamically
More than one item, such as: ♦ All the data in a column ♦ A block of data specified by ranges of rows and columns ♦ Data in selected rows ♦ All the data in the DataWindow	An expression. Specifying the data you want in a single statement is much more efficient than calling the functions repeatedly in a program loop

What's in this chapter

The rest of this chapter describes how to construct expressions for accessing DataWindow data.

For information on functions

For information about the functions for accessing data, see SetItem, GetItemDate, GetItemDateTime, GetItemNumber, GetItemString, and GetItemTime in the *PowerScript Reference*.

About DataWindow data expressions

The Object property of the DataWindow control lets you specify expressions that refer directly to the data of the DataWindow object in the control. This direct data manipulation allows you to access small and large amounts of data in a single statement, without calling functions.

There are several variations of data expression syntax, divided into three groups. This section summarizes the syntax for the three groups. The syntaxes are described in detail later in this chapter.

Data in columns or computed fields when you know the name

One or all items in the column (if rownum is absent, include either *buffer* or *datasource*)

dwcontrol.Object.*columnname* {.*buffer* } {.*datasource* } { [*rownum*] }

Returns a single value (for a specific row number) or an array of values (when *rownum* is omitted).

FOR INFO See "One or all items in a named column or computed field" on page 182.

Selected items

dwcontrol.Object.*columnname* {.Primary }{.*datasource* }.Selected

Returns an array of values with an array element for each selected row.

FOR INFO See "Items in the column from selected rows" on page 186.

Range of items

dwcontrol.Object.*columnname* {.*buffer* } {.*datasource* } [*startrownum*, *endrownum*]

Returns an array of values with an array element for each row in the range.

FOR INFO See "Items from a range of rows in a column or computed field" on page 187.

Data in numbered columns

Single items

dwcontrol.Object.Data {.*buffer* } {.*datasource* } [*rownum*, *colnum*]

Returns a single item whose data type is the data type of the column.

FOR INFO See Single items.

Blocks of data involving a range of rows and columns

dwcontrol.Object.Data {.*buffer* } {.*datasource* } [*startrownum*, *startcolnum*, *endrownum*, *endcolnum*]

Returns an array of structures or user objects. The structure elements match the columns in the range. There is one array element for each row in the range.

FOR INFO See "A block of rows and columns" on page 190.

Whole rows

Single row or all rows

dwcontrol.Object.Data {.*buffer*} {.*datasource*} { [*rownum*] }

Returns one structure or user object (for a single row) or an array of them (for all rows). The structure elements match the columns in the DataWindow object.

FOR INFO See "Single rows or all rows" on page 193.

Selected rows

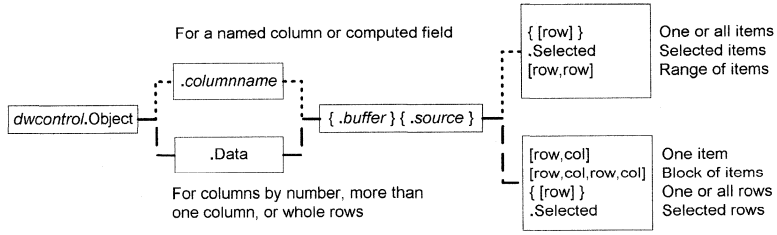
dwcontrol.Object.Data {.Primary} {.*datasource*} .Selected

Returns an array of structures or user objects. The structure elements match the columns in the DataWindow object. There is one array element for each selected row.

FOR INFO See "Selected rows" on page 194.

Summary of syntaxes

This diagram summarizes the variations in data expression syntax:



FOR INFO For information about getting and setting values of DataWindow object properties using a similar syntax, see "Basic syntax" on page 207 and "Syntax for nested objects" on page 210.

When the expression is evaluated

Expressions that refer to DataWindow data are not verified until execution of your application.

No compiler checking

When your script is compiled, PowerBuilder does not verify the parameters of the expression that follow the Object property. Your application can select or change the DataWindow object in a DataWindow control during execution without invalidating the compiled script.

Potential execution errors

If the data type of the expression isn't compatible with how the expression is used, or if the specified rows or columns do not exist, an error will occur during execution.

You can handle the error by writing a script for the DataWindow control's Error event.

Getting and storing the data

A DataWindow data expression can return a large amount of data.

Data structures for data

Single row and column When your data expression refers to a single row and column, you can assign the data to a variable whose data matches the column's data type. When the expression refers to a single column but can refer to multiple rows, you must specify an array of the appropriate data type.

More than one column When the expression refers to more than one column, you can get or set the data with a structure or user object. When you create the definition, you must assign data types to the fields (in a structure) or instance variables (in a user object) that match the data types of the columns. When your expression refers to multiple rows, you will have an array of the structure or user object.

Likewise, if you want to set data in the DataWindow control, you will set up the data in structures or user objects whose elements match the columns referred to in the expression.

Data types For matching purposes, the data types should be appropriate to the data—for example, any numeric data type matches any other numeric type.

Examples of data structures

The following table presents some examples of data specified by an expression and the type of data structures you might define for storing the data:

Type of selection	Sample data storage
A single item	A single variable of the appropriate data type
A column of values	An array of the appropriate data type

Type of selection	Sample data storage
A row	A structure whose elements have data types that match the DataWindow object's columns A user object whose instance variables match the DataWindow object's columns
Selected rows or all rows	An array of the structure or user object defined for a row
A block of values	An array of structures or user objects whose elements or instance variables match the columns included in the selected range

Assigning data to arrays

When a data expression is assigned to an array, values are assigned beginning with array element 1 regardless of the starting row number. If the array is larger than the number of rows accessed, elements beyond that number are unchanged. If it is smaller, a variable-size array will grow to hold the new values. However, a fixed-size array that is too small for the number of rows will cause an execution error.

Two ways to instantiate user objects

A user object needs to be instantiated before it is used.

One way is to use the CREATE statement after you declare the user object. If you declare an array of the user object, you must use CREATE for each array element.

The second way is to select the Autoinstantiate box for the user object in the User Object painter. When you declare the user object in a script, the user object will be automatically instantiated, like a structure.

Any data type and data expressions

The actual data type of a DataWindow data expression is Any, which allows the compiler to process the expression even though the final data type is unknown. When data is accessed at runtime, you can assign the result to another Any variable or to a variable, structure, or user object whose data type matches the real data.

Examples

A single value This example gets a value from column 2, whose data type is string:

```
string ls_name
ls_name = dw_1.Object.Data[1,2]
```

A structure that matches DataWindow columns In this example, a DataWindow object has four columns:

- An ID (number)
- A name (string)
- A retired status (boolean)
- A birth date (date)

A structure to hold these values has been defined in the Structure painter. It is named str_empdata and has four elements whose data types are integer, string, boolean, and date. To store the values of an expression that accesses some or all the rows, you need an array of str_empdata structures to hold the data:

```
str_empdata lstr_currdata[]
lstr_currdata = dw_1.Object.Data
```

After this example executes, the upper bound of the array of structures, which is variable-size, is equal to the number of rows in the DataWindow control.

A user object that matches DataWindow columns If the preceding example involved a user object instead of a structure, then a user object defined in the User Object painter, called uo_empdata, would have four instance variables, defined in the same order as the DataWindow columns:

```
integer id
string name
boolean retired
date birthdate
```

Before accessing three rows, three array elements of the user object have been created (you could use a FOR NEXT loop for this). The user object was not defined with the Autoinstantiate box:

```
uo_empdata luo_empdata[3]
luo_empdata[1] = CREATE uo_empdata
luo_empdata[2] = CREATE uo_empdata
luo_empdata[3] = CREATE uo_empdata
luo_empdata = dw_1.Object.Data[1,1,3,4]
```

Setting DataWindow data

When you set data in a DataWindow control, the data types of the source values must match the data types of the columns being set.

Data structures for setting data

When your data expression refers to a single row and column, you can set the value in the DataWindow control with a value that matches the column's data type. When you are setting values in a single column and specifying an expression that can refer to multiple rows, the values you assign must be in an array of the appropriate data type.

Defining user objects or structures When the expression refers to more than one column, you can assign the data with a structure or user object to the DataWindow data. When you create the definition, the fields (in a structure) or instance variables (in a user object) must match the columns. There must be the same number of fields or variables, defined in the same order as the columns, with compatible data types.

When your expression can refer to multiple rows, you will need an array of the structure or user object.

Using arrays to set values

You don't have to know in advance how many rows are involved when you are setting data in the DataWindow control. PowerBuilder will use the number of elements in the source array and the number of rows in the target expression to determine how to make the assignment and whether it is necessary to insert rows.

If the target expression is *selected rows or a range of rows*, then:

- When there are *more* array elements than target rows, the extra array elements are ignored
- When there a *fewer* array elements than target rows, the column(s) in the extra target rows are filled with default values

If the target expression is *all rows in a column or subset of columns*, then:

- When there are *more* array elements than target rows, the extra array elements are ignored
- When there a *fewer* array elements than target rows, only the first rows up to the number of array elements are affected

If the target expression is *all rows and all columns*, then the source data replaces all the existing rows, resetting the DataWindow control to the new data.

Inserting new rows When you are setting data and you specify a range, then if rows do not exist in that range, rows will be inserted to fill the range. For example, if the DataWindow control has four rows and your expression says to assign data to rows 8 through 12, then eight more rows are added to the DataWindow control. The new rows use the initial default values set up for each column. After the rows are inserted, the array of source data is applied to the rows as described above.

Examples

These examples refer to a DataWindow object that has three columns: emp_id, emp_lname, and salary. The window declares these arrays as instance variables and the window's Open event assigns four elements to each array:

```
integer ii_id[]
string is_name[]
double id_salary[]
uo_empdata iuo_data[]
uo_empid_name iuo_id[]
```

The uo_empdata user object has three instance variables: id, name, and salary. The uo_empid_name user object has two instance variables: id and name.

This example sets emp_lname in the selected rows to the values of is_name, an array with four elements. If two rows are selected, only the first two values of the array are used. If six rows are selected, the last two rows of the selection are set to an empty string:

```
dw_1.Object.emp_lname.Selected = is_name
```

This example sets salary in rows 8 to 12 to the values in the array id_salary. The id_salary array has only four elements, so the extra rows in the range are set to 0 or a default value:

```
dw_1.Object.salary[8,12] = id_salary
```

This statement resets the DataWindow control and inserts four rows to match the array elements of iuo_data:

```
dw_1.Object.Data.Primary = iuo_data
```

This example sets columns 1 and 2 in rows 5 to 8 to the values in the array iuo_id:

```
dw_1.Object.Data.Primary[5,1, 8,2] = iuo_id
```

This example sets emp_id in the first four rows to the values in the ii_id array:

```
dw_1.Object.emp_id.Primary = ii_id
```

Syntaxes for DataWindow data expressions

Expressions with a named column or computed field

The syntaxes in this section allow you to name a column or computed field and retrieve:

- One or all of its items
- Items for selected rows
- Items for a range of rows

The accessed data can be stored in an array whose data type matches the column data type.

One or all items in a named column or computed field

Description

You can access a single item in a column or computed field by specifying the object name and a row number. You can access all the data in the column by omitting the row number.

Syntax

dwcontrol.Object.*columnname* {.*buffer* } {.*datasource* } { [*rownum*] }

Parameter	Description
dwcontrol	The name of the DataWindow control or child DataWindow in which you want to get or set data
columnname	The name of a column or computed field in the DataWindow object in *dwcontrol*. If the column or computed field doesn't exist at runtime, an execution error occurs
buffer (optional)	The name of the buffer from which you want to get or set data. Values are: ♦ Primary — (Default) The data in the primary buffer (the data that has not been deleted or filtered out) ♦ Delete — The data in the delete buffer (data deleted from the DataWindow control) ♦ Filter — The data in the filter buffer (data that was filtered out)

Parameter	Description
datasource (optional)	The source of the data. Values are: ♦ Current — (Default) The current values in the DataWindow control ♦ Original — The values that were initially retrieved from the database. For a computed field, you must specify Original because computed fields cannot be changed and don't have current values
rownum (optional)	The row number of the desired item. *The row number must be enclosed in brackets* To access all the data in the column, omit *rownum* **When buffer or datasource is not optional** When *rownum* is omitted, you must specify at least one of the other elements in the expression: either *buffer* or *datasource*

Return value

A single value (for a specific row number) or an array of values (when *rownum* is omitted). Each value's data type is the data type of *columnname*.

Usage

Is the expression a DWObject or data? When you want to access all the data in the column, remember to specify at least one of the other optional parameters. Otherwise, the expression you specify refers to the column *object*, not its data. This expression refers to the DWObject empname, not the data in the column:

```
dw_1.Object.empname
```

In contrast, this expressions all refer to data in the empname column:

```
dw_1.Object.empname.Primary // All rows
dw_1.Object.empname[5]      // Row 5
```

Row numbers for computed fields When you refer to an object in a band other than the detail band (usually a computed field) you still specify a row number. For the header, footer, or summary, specify a row number of 1. For the group header or trailer, specify the group number:

```
dw_1.Object.avg_cf[1]
```

If you specify nothing after the computed field name, you refer to the computed field DWObject, not the data. For a computed field that occurs more than once, you can get all values by specifying *buffer* or *datasource* instead of *rownum*, just as for columns.

When the expression is an array When the expression returns an array (because there is no row number), you must assign the result to an array, even if you know there is only one row in the result.

This expression returns an array, even if there is only one row in the DataWindow control:

```
dw_1.Object.empname.Primary
```

This expression returns a single value:

```
dw_1.Object.empname[22]
```

Examples

Because the default setting is current values in the primary buffer, the following expressions are equivalent—both get the value in row 1 for the emp_name column:

```
dw_1.Object.emp_name[1]
dw_1.Object.emp_name.Primary.Current[1]
```

This statement sets the emp_name value in row 1 to Wilson:

```
dw_1.Object.emp_name[1] = "Wilson"
```

This statement gets values for all the emp_name values that have been retrieved and assigns them to an array of strings:

```
string ls_namearray[]
ls_namearray = dw_1.Object.emp_name.Current
```

This statement gets current values of emp_name from all rows in the filter buffer:

```
string ls_namearray[]
ls_namearray = dw_1.Object.emp_name.Filter
```

This statement gets original values of emp_name from all rows in the filter buffer:

```
string ls_namearray[]
ls_namearray = dw_1.Object.emp_name.Filter.Original
```

This statement gets current values of emp_name from row 14 in the delete buffer:

```
string ls_name
ls_name = dw_1.Object.emp_name.Delete[14]
```

This statement gets original values of emp_name from row 14 in the delete buffer:

```
string ls_name
ls_name = dw_1.Object.emp_name.Delete.Original[14]
```

This statement gets all the values of the computed field review_date:

```
string ld_review[]
ld_review = dw_1.Object.review_date.Original
```

Items in the column from selected rows

Description

The Selected property accesses values in the named column or computed field for the currently selected rows.

Syntax

dwcontrol.Object.*columnname* {.Primary } {.*datasource* }.Selected

Parameter	Description
dwcontrol	The name of the DataWindow control or child DataWindow in which you want to get or set data
columnname	The name of a column or computed field in the DataWindow object in *dwcontrol*. If the column or computed field doesn't exist at runtime, an execution error occurs
datasource (optional)	The source of the data. Values are: ♦ Current — (Default) The current values in the DataWindow control ♦ Original — The values that were initially retrieved from the database. For a computed field, you must specify Original (because computed fields cannot be changed and don't have current values)

Return value

An array of values with the data type of *columnname*.

Usage

When you specify selected values, the expression always returns an array and you must assign the result to an array, even if you know there is only row selected.

For selected rows, the primary buffer is the only applicable buffer. For consistency, you can include Primary in this syntax also.

Examples

Because the primary buffer is the only applicable buffer for selected data and current data is the default, these expressions are all equivalent. They access values in the emp_name column for selected rows:

```
dw_1.Object.emp_name.Selected
dw_1.Object.emp_name.Primary.Selected
```

```
dw_1.Object.emp_name.Current.Selected
dw_1.Object.emp_name.Primary.Current.Selected
```

These expressions both access original values for selected rows:

```
dw_1.Object.emp_name.Original.Selected
dw_1.Object.emp_name.Primary.Original.Selected
```

This example sets the emp_name value in the first selected row to an empty string. The rest of the selected rows are set to a default value, which may be an empty string:

```
string ls_empty[]
ls_empty[1] = ""
dw_1.Object.emp_lname.Selected = ls_empty
```

This statement gets the original emp_name values in selected rows and assigns them to an array of strings:

```
string ls_namearray[]
ls_namearray = dw_1.Object.emp_name.Original.Selected
```

Items from a range of rows in a column or computed field

Description

You can access values in a named column or computed field for a range of rows by specifying the starting and ending row numbers.

Syntax

dwcontrol.Object.*columnname* {.*buffer*} {.*datasource*} [*startrownum*, *endrownum*]

Parameter	Description
dwcontrol	The name of the DataWindow control or child DataWindow in which you want to get or set data
columnname	The name of a column or computed field in the DataWindow object in *dwcontrol*. If the column or computed field doesn't exist at runtime, an execution error occurs
buffer (optional)	The name of the buffer from which you want to get or set data. Values are: ♦ Primary — (Default) The data in the primary buffer (the data that has not been deleted or filtered out) ♦ Delete — The data in the delete buffer (data deleted from the DataWindow control) ♦ Filter — The data in the filter buffer (data that was filtered out)

Parameter	Description
datasource (optional)	The source of the data. Values are: ♦ Current — (Default) The current values in the DataWindow control ♦ Original — The values that were initially retrieved from the database. For a computed field, you must specify Original (because computed fields cannot be changed and don't have current values)
startrownum	The number of the first row in the desired range of rows
endrownum	The number of the last row in the desired range of rows *The row numbers must be enclosed in brackets and separated by commas*

Return value

An array of values with an array element for each row in the range. Each value's data type is the data type of *columnname*.

Usage

When you specify a range, the expression always returns an array and you must assign the result to an array, even if you know there is only one value in the result. For example, this expression returns an array of one value:

```
dw_1.Object.empname[22,22]
```

Examples

Because the primary buffer and current data are the default, these expressions are all equivalent:

```
dw_1.Object.emp_name[11,20]
dw_1.Object.emp_name.Primary[11,20]
dw_1.Object.emp_name.Current[11,20]
dw_1.Object.emp_name.Primary.Current[11,20]
```

This example resets the emp_name value in rows 11 through 20 to an empty string. Rows 12 to 20 are set to a default value, which may be an empty string:

```
string ls_empty[]
ls_empty[1] = ""
dw_1.Object.emp_name[11,20] = &
    {"","","","","","","","","",""}
```

This statement gets the original emp_name values in rows 11 to 20 and assigns them to elements 1 to 10 in an array of strings:

```
string ls_namearray[]
ls_namearray = dw_1.Object.emp_name.Original[11,20]
```

This statement gets current values of emp_name from rows 5 to 8 in the Filter buffer and assigns them to elements 1 to 4 in an array of strings:

```
string ls_namearray[]
ls_namearray = dw_1.Object.emp_name.Filter[5,8]
```

This statement gets original values of emp_name instead of current values, as shown in the previous example:

```
string ls_namearray[]
ls_namearray = &
   dw_1.Object.emp_name.Filter.Original[5,8]
```

This statement gets current values of emp_name from rows 50 to 200 in the delete buffer and assigns them to elements 1 to 151 in an array of strings:

```
string ls_namearray[]
ls_namearray = dw_1.Object.emp_name.Delete[50,200]
```

This statement gets original values of emp_name instead of current values, as shown in the previous example:

```
string ls_namearray[]
ls_namearray = &
   dw_1.Object.emp_name.Delete.Original[50,200]
```

Expressions with column numbers

If you know the column's number but not the name, you can specify row and column numbers in brackets to refer to:

- Single items
- Blocks of data made up of ranges or rows and columns

Single items

Description

You can access an single data item by specifying its row and column number.

Syntax

dwcontrol.Object.Data {.*buffer*} {.*datasource*} [*rownum*, *colnum*]

Parameter	Description
dwcontrol	The name of the DataWindow control or child DataWindow in which you want to get or set data

Parameter	Description
buffer (optional)	The name of the buffer from which you want to get or set data. Values are: ♦ Primary — (Default) The data in the primary buffer (the data that has not been deleted or filtered out) ♦ Delete — The data in the delete buffer (data deleted from the DataWindow control) ♦ Filter — The data in the filter buffer (data that was filtered out)
datasource (optional)	The source of the data. Values are: ♦ Current — (Default) The current values in the DataWindow control ♦ Original — The values that were initially retrieved from the database
rownum	The row number of the desired item
colnum	The column number of the desired item *The row and column numbers must be enclosed in brackets and separated by commas*

Return value

A single item in the DataWindow control. Its data type is the data type of the column.

Examples

These expressions both refer to a single item in row 1, column 2. It accesses current data in the primary buffer:

```
dw_1.Object.Data[1,2]
dw_1.Object.Data.Primary.Current[1,2]
```

This statement changes the value of the original data to 0 for the item in row 1, column 2 in the Filter buffer. Column 2 holds numeric data:

```
dw_1.Object.Data.Filter.Original[1,2] = 0
```

A block of rows and columns

Description

You can access data in a range of rows and columns by specifying the starting and ending row and column numbers.

Syntax

dwcontrol.Object.Data {.*buffer*} {.*datasource*} [*startrownum*, *startcolnum*, *endrownum*, *endcolnum*]

Parameter	Description
dwcontrol	The name of the DataWindow control or child DataWindow in which you want to get or set data
buffer (optional)	The name of the buffer from which you want to get or set data. Values are: ♦ Primary — (Default) The data in the primary buffer (the data that has not been deleted or filtered out) ♦ Delete — The data in the delete buffer (data deleted from the DataWindow control) ♦ Filter — The data in the filter buffer (data that was filtered out)
datasource (optional)	The source of the data. Values are: ♦ Current — (Default) The current values in the DataWindow control ♦ Original — The values that were initially retrieved from the database
startrownum	The number of the first row in the desired range of rows
startcolnum	The number for the first column in the range
endrownum	The number of the last row in the desired range of rows
endcolnum	The number for the last column in the range *The row and column numbers must be enclosed in brackets and separated by commas*

Return value

An array of structures or user objects. There is one structure element or user object instance variable for each column in the designated range. The data type of each element matches the data type of the corresponding column. There is one structure or user object in the array for each row in the range of rows.

Usage

When you specify a block, the expression always returns an array and you must assign the result to an array, even if you know there is only one structure in the result.

This expression returns an array of one structure from row 22:

```
dw_1.Object.data[22,1,22,4]
```

This expression returns an array of one value from row 22, column 1:

```
dw_1.Object.data[22,1,22,1]
```

Examples

These statements both refer to data in the first ten rows and first four columns of the DataWindow object in the control dw_1. The primary buffer and current data are the default:

```
dw_1.Object.Data[1,1,10,4]
dw_1.Object.Data.Primary.Current[1,1,10,4]
```

This example gets employee IDs and last names for all the rows in the delete buffer. The IDs and names are the first two columns. It saves the information in a structure, called str_namelist, of two elements: an integer called id and a string called lastname. The structure was defined previously in the Structure painter. The list of IDs and names is then saved in the file DELETED.TXT:

```
integer li_fileNum
long ll_deletedrows
str_namelist lstr_namelist[]

ll_deletedrows = dw_1.DeletedCount()
lstr_namelist = &
   dw_1.Object.Data.Delete[1,1, ll_deletedrows,2]

li_fileNum = FileOpen("C:\HR\DELETED.TXT", &
   LineMode!, Write!)
FOR ll_count = 1 to UpperBound(lstr_namelist)
   FileWrite(li_fileNum, &
      String(lstr_namelist.id) + &
      " " + &
      lstr_namelist.lastname + &
      "~r~n")
NEXT
FileClose(li_fileNum)
```

Using the structure from the previous example that holds IDs and last names, this example sets all the IDs and last names in the DataWindow control to NULL:

```
long ll_n
str_namelist lstr_namelist[]

SetNull(lstr_namelist[1].id)
SetNull(lstr_namelist[1].lastname)

FOR ll_n = 2 to dw_1.RowCount()
   lstr_namelist[ll_n] = lstr_namelist[1]
NEXT
```

```
dw_1.Object.Data[1,1, dw_1.RowCount(),2] = lstr_data
```

Expressions that access whole rows

The syntaxes in this section access single rows, all rows, or currently selected rows.

Single rows or all rows

Description

You can access a single row by specifying the row number, and you can access all the data in the DataWindow control by omitting the row number.

Syntax

dwcontrol.Object.Data {.*buffer*} {.*datasource*} { [*rownum*] }

Parameter	Description
dwcontrol	The name of the DataWindow control or child DataWindow in which you want to get or set data
buffer (optional)	The name of the buffer from which you want to get or set data. Values are: ♦ Primary — (Default) The data in the primary buffer (the data that has not been deleted or filtered out) ♦ Delete — The data in the delete buffer (data deleted from the DataWindow control) ♦ Filter — The data in the filter buffer (data that was filtered out)
datasource (optional)	The source of the data. Values are: ♦ Current — (Default) The current values in the DataWindow control ♦ Original — The values that were initially retrieved from the database
rownum (optional)	The number of the row you want to access To access data for all rows, omit *rownum* *The row number must be enclosed in brackets*

Return value

One structure or user object (for a single row) or an array of them (for all rows). There is one structure element or instance variable for each column in the DataWindow object. The data type of each element matches the data type of the corresponding column.

Usage

When you specify an array (the row number is omitted), the expression always returns an array and you must assign the result to an array, even if you know there is only one row in the DataWindow control.

Examples

These statements both access current data for row 5 in the primary buffer in the DataWindow object contained in the DataWindow control dw_1:

```
dw_1.Object.Data[5]
dw_1.Object.Data.Primary.Current[5]
```

This example assigns all the data in dw_1 to the Any variable la_dwdata:

```
any la_dwdata
la_dwdata = dw_1.Object.Data
```

This example assigns all the data in the delete buffer for dw_1 to the Any variable la_dwdata:

```
any la_dwdata
la_dwdata = dw_1.Object.Data.Delete
```

This example replaces all the data in the nested report in row 2 with data from dw_2. The columns in the DataWindow object in dw_2 must match the columns in the DataWindow object for the nested report:

```
dw_1.Object.NestRep[2].Object.Data = &
    dw_2.Object.Data
```

Selected rows

Description

You can access all the data in the currently selected rows by specifying the Selected property. Selected rows are always in the primary buffer.

Syntax

dwcontrol.Object.Data {.Primary } {.*datasource* }.Selected

Parameter	Description
dwcontrol	The name of the DataWindow control or child DataWindow in which you want to get or set data
datasource (optional)	The source of the data. Values are: ♦ Current — (Default) The current values in the DataWindow control ♦ Original — The values that were initially retrieved from the database

Return values

An array of structures or user objects. There is one structure element or instance variable for each column in the DataWindow object. The data type of each element matches the data type of the corresponding column.

Usage

When you specify selected rows, the expression always returns an array and you must assign the result to an array, even if you know there is only one row selected.

Examples

Because the primary buffer is the only applicable buffer for selected data and current data is the default, these expressions are all equivalent. They access data in the selected rows:

```
dw_1.Object.Data.Selected
dw_1.Object.Data.Primary.Selected
dw_1.Object.Data.Current.Selected
dw_1.Object.Data.Primary.Current.Selected
```

These expressions both access original values for selected rows:

```
dw_1.Object.Data.Original.Selected
dw_1.Object.Data.Primary.Original.Selected
```

This example takes the values in the selected rows in dw_2 and populates a DropDownDataWindow in dw_1 with the values, replacing existing data in the DropDownDataWindow. The column with the DropDownDataWindow is called useroptions. The columns of the DataWindow object in dw_2 must match the columns of the DataWindow object for the DropDownDataWindow:

```
dw_1.Object.useroptions.Object.Data = &
   dw_2.Object.Data.Selected
```

CHAPTER 4

Accessing DataWindow Properties in Scripts

About this chapter

This chapter explains the syntax for constructing expressions that access properties of objects within a DataWindow.

Contents

What are DataWindow object properties?

DataWindow object properties store the information that controls the behavior of the DataWindow object. They are not properties of the DataWindow control, but of the DataWindow object displayed in the control.

Types of values

Property values may be constants or they may be DataWindow painter expressions. **DataWindow painter expressions** allow the property value to be based on other conditions in the DataWindow, including data values. Conditional expressions based on data give the property a different value for each row.

Getting and setting values

You establish initial values for DataWindow properties in the DataWindow painter. You can also get and set property values during execution in scripts.

There are several methods for accessing property values. A particular property may be accessible by a subset of those methods. For example, some properties are read-only during execution, some can be set only at execution, and some accept only constants (not DataWindow painter expressions) as values.

FOR INFO For a complete list of properties and the ways you can access each one, see Chapter 5, "DataWindow Object Properties."

Examples: ways of setting the Border property

This table lists the ways you can access properties, using the Border property as an example:

What you can do with properties	How to do it, using the Border property as an example	What happens
Set the initial value of the property in the workspace	StyleBar *or* Properties dialog box, General tab, Border box	The Border property takes on the value you set unconditionally. At runtime, the object has the border you indicated in the workspace unless you set the Border property again in some way

What you can do with properties	How to do it, using the Border property as an example	What happens
Control the value of the property at runtime based on an expression defined for the object in the workspace	Properties dialog box, Expression tab, Border property	At runtime, the value of the Border property changes as specified in the expression coded for the property. This value overrides the explicit DataWindow painter setting. For example, you can code an expression that gives the Salary column value a ShadowBox border whenever the salary exceeds $70,000
Get the value of the property in an event script	Dot notation syntax for the Border property *or* Describe function	Both the dot notation syntax and the Describe function return the value of the Border property for the specified object
Change the value of the property at runtime in an event script	Dot notation syntax for the Border property *or* Modify function	At runtime, the value of the property changes when the code for the event executes. For example, you could code Modify in the Clicked event for an object and change the border of the object
Set the initial value of the property at runtime in an event script for a DataWindow being created	SyntaxFromSQL	At runtime, the value of the property is set when the SyntaxFromSQL executes. For example, you could set the value of the Border property of all columns in a DataWindow being created with SyntaxFromSQL FOR INFO For more information, see the SyntaxFromSQL function in the *PowerScript Reference*

Specifying property values in the DataWindow painter

When you make selections and specify values on any of the property sheets in the DataWindow painter, you are setting DataWindow object properties.

Properties for each DataWindow object

Each object in the DataWindow (columns, text, drawing objects) has its own property sheets, because there are different sets of properties for each object. To access the property sheet for an object, right-click on the object and select Properties from the popup menu.

FOR INFO For information on valid values and information on the property names associated with property sheet settings, see the Help topic for each property sheet.

DataWindow painter expressions for properties

On the Expressions tab page in an object's property sheet, you can provide DataWindow painter expressions for the listed properties. At runtime, the expression is evaluated for each row. When the expression includes the row-dependent information in the calculation (such as data), each row can have a different value for the property. You do not see the results in the painter, but only when you run the DataWindow.

A DataWindow painter expression can include:

- Column or computed field names. The actual data in each row is factored into the calculation.
- DataWindow painter functions. Functions, such as IsRowNew, can refer to characteristics of an individual row.

FOR INFO For information about the components of expressions, see "Using DataWindow painter and InfoMaker functions" on page 16 and the *PowerBuilder User's Guide*. For examples of expressions, see "Using DataWindow painter expressions" on page 218.

Methods for accessing property values in scripts

Two methods

There are two ways to access property values in a DataWindow:

- **Functions** The Describe and Modify functions use strings to specify the property names. For example:

```
dw_1.Describe("empname.Border")
dw_1.Modify("empname.Border=1")
```

- **Expressions** DataWindow property expressions use the Object property and dot notation. For example:

```
dw_1.Object.empname.Border = 1
li_border = Integer(dw_1.Object.empname.Border)
```

Which method to use

The method you use depends on the type of error checking you want to provide and on whether you know the names of the objects and properties you want to access when the script is compiled.

If you want to	Use
Use column and property names that are known when the script is compiled	An expression
Avoid extra nested tildes (and you know the column and property names you want to access)	An expression
Build a string at runtime that names objects and properties	A function
Use the Error event to handle problems with incorrect object or property names	An expression and a script for the Error event
Avoid using the Error event for handling problems with incorrect object or property names	A function and code that evaluates its return value

About the Modify and Describe functions

This section describes:

- The advantage and drawbacks of the Modify and Describe functions
- How to handle errors

Advantage and drawbacks

Using the Describe and Modify functions to access property values has advantages and drawbacks. The examples here use Modify as illustrations, but similar considerations apply to Describe.

Advantage

Specifying column and property names dynamically In your script, you can build a string that specifies the column and property names.

For example, the following code builds a string in which the default color value and the two color values in the If function are determined in the script. Notice how the single quotes around the expression are included in the first and last pieces of the string:

```
red_amount = Integer(sle_1.Text)
modstring = "emp_id.Color='" + &
    String(RGB(red_amount, 0, 0)) + &
    "~tIf(emp_status=~~'A~~'," + &
    String(RGB(255, 0, 0)) + &
    "," + &
    String(RGB(red_amount, 0, 0)) + &
    ")'"
Modify(modstring)
```

The resulting string when red_amount is set to 128 is:

```
emp_id.Color='128~tIf(emp_status=~'A~',255,128)'
```

The following is a simpler example without the If function. You don't need quotes around the value if you are not specifying an expression. Here the String and RGB functions result in a constant value in the resulting modstring:

```
Modify(ls_columnname + ".Color=" + &
    String(RGB(red_amount, 255, 255)))
```

Drawbacks

Setting several properties at once Although you can set several properties in a single function call, it is harder to understand and debug scripts that do so.

For example, assume the following is entered on a single line in the script editor:

```
rtn = dw_1.Modify("emp_id.Font.Italic=0
oval_1.Background.Mode=0
oval_1.Background.Color=255")
```

Using a DWObject variable in several property expressions is a little more efficient that setting several properties in a single function call. However, if you want to take advantage of dynamically naming objects, you may still choose to use a function.

FOR INFO For examples of using a DWObject variable, see "Using the DWObject variable" on page 214.

Specifying complex quoted strings When you specify an expression for a property value, it is difficult to specify nested quotes correctly—the code is hard to understand and prone to error. For Describe, this is less of a drawback—strings will not become as complex because they don't include an expression.

For example, this string entered on a single line in a script assigns a DataWindow painter expression to the Color property:

```
Modify("emp_id.Color=~"16777215 ~t
If(emp_status=~~~"A~~~",255,16777215)~"")
```

FOR INFO For more information about quoted strings, see the *PowerScript Reference.*

Handling errors

No runtime error occurs when Describe and Modify try to access invalid objects or properties. The validity of the argument string is evaluated before the objects are accessed.

Modify

When the string that specifies the object and property to be accessed is invalid, Modify returns an error string, instead of the expected value, such as:

```
Line 1 Column 12: incorrect syntax.
```

You can use the error message to figure out what part of the string is incorrect. This is most useful when you are testing your scripts. The error message, which names the line and column number after which the string was not recognized, may not be helpful after your application is deployed.

Describe

When the string for Describe has an unrecognized property, Describe's return value ends with an exclamation point (!). It will return as many values as it recognizes up to the incorrect one.

When you specify a valid property but that property doesn't have a value (either because it hasn't been set or because its value is an expression that can't be evaluated), Describe returns a question mark (?) for that property. The property's actual value is NULL.

Always check for errors
You should include error-checking code that checks for these return values. Other errors can occur later if you depend on settings that failed to take effect.

For more information

For more information about the syntax and ways of using Describe and Modify, see the *PowerScript Reference*.

About DataWindow property expressions

Before presenting the full syntax of a property expression, this section describes:

- How the property expression syntax is related to the structure of the DataWindow object
- The data types of property values and property expressions
- When the expression is evaluated
- How to handle errors

Basic structure of DataWindows and property expressions

Objects in a DataWindow

A DataWindow object is made up of many other objects (such as Columns, Text, Bitmaps, and Reports) that are themselves DataWindow objects with their own objects. The syntax of a property expression allows you to address any of these objects.

Object property

A DataWindow property expression uses the Object property of the DataWindow control to access the DataWindow object. Following the Object property, you specify an object name and one or more properties.

The simple syntax is:

dwcontrol.**Object**.*dwobjectname.property*

For example:

```
dw_1.Object.empname.Resizeable
```

FOR INFO For the full syntax, see "Syntax for property expressions" on page 207.

About DataWindow data expressions

Expressions that access data in a DataWindow object using dot notation use the Data property. These expressions are called **data expressions** (in contrast to property expressions); because of the intricate syntax for data expressions, they are described separately, in Chapter 3, "Accessing Data in Scripts."

Data types of property expressions

DataWindow property values

The values of DataWindow object properties are strings. These strings may contain numeric or yes/no values, but the values you access are strings, not integers or boolean values.

Although the property values are really strings, the script compiler allows you to assign numbers and boolean values to properties whose strings represent numeric values or contain yes/no strings. This does not mean the data type is integer or boolean. It is just a convenience when assigning a value to the property.

For example, both of these statements are correct:

```
dw_1.Object.empname.Border = 1
dw_1.Object.empname.Border = '1'
```

DataWindow property expressions

The data type of a property expression is Any (not string), but the value of the data in the Any variable is a string. This may sound like an unnecessary distinction, but it does matter when you use a property expression as a function argument. If the function doesn't accept an Any variable as a argument, you may need to use the String function to cast the data to the correct data type.

For example, because the MessageBox function accepts a string argument not an Any, the property expression is enclosed in a String conversion function:

```
MessageBox("Border", &
    String(dw_1.Object.empname.Border))
```

When the expression is evaluated

Expressions that refer to DataWindow object properties are not verified until your application runs.

No compiler checking

When your script is compiled, PowerBuilder does not verify the parameters of the expression that follow the Object property. Your application can select the DataWindow object in a DataWindow control during execution without invalidating the compiled script.

Potential execution errors

If the data type of the expression isn't compatible with how the expression is used, or if the specified rows or columns do not exist, then an error will occur during execution.

You can handle the error by writing a script for the DataWindow Error event.

Handling errors in the Error event

What causes errors

An invalid property expression causes a runtime error in your application. A runtime error causes the application to terminate unless there is a script for the Error event:

Conditions that cause errors	Possible causes
Invalid names of objects within the DataWindow object	Mistyping, which the compiler doesn't catch because it doesn't evaluate the expression A different DataWindow object has been inserted in the control and it has different columns and objects
A property isn't valid for the specified object	Mistyping The object is a different type than expected

Responding to errors in the Error event script

You can prevent the application from terminating by handling the error in the DataWindow control's Error event. The Error event's arguments give you several options for responding to the error. You choose a course of action and set the *action* argument to a value of the ExceptionAction enumerated data type:

Response	ExceptionAction value	Considerations
Ignore the error and allow the application to continue	ExceptionIgnore!	Use only if the correct evaluation of the expression was not critical to the application
Pass the error to the SystemError event	ExceptionFail!	Generally, if the SystemError event occurs, you should do application cleanup tasks and terminate the application
Provide a value to use instead	ExceptionSubstitute ReturnValue!	Useful when you were getting the value of a property In addition to setting *action*, you set the *returnvalue* argument to the value you want to use instead

Providing a default value

If you are trying to find out a property value and you know the expression might cause an error, you can include code that prepares for the error by storing a default value in an instance variable. Then the Error event script can return that value in place of the failed expression.

There are three elements to this technique: the declaration of an instance variable, the script that sets the variable's default value and then accesses a DataWindow property, and the Error event script. These elements are shown in Example 3 below.

Examples

Example 1. In the Error event, you might display a MessageBox reporting the error to the user. This message reports the object that had the error and the error message itself:

```
MessageBox(errorobject, errortext)
```

Example 2. This code in an Error event script displays complete information about the error in a MultiLineEdit mle_1:

```
mle_1.Text = &
    "Error#: " + String(errornumber) + "~r~n" + &
    "Text: " + errortext + "~r~n" + &
    "Parent: " + errorwindowmenu + "~r~n" + &
    "Object: " + errorobject + "~r~n" + &
    "Line: " + String(errorline) + "~r~n"
```

Example 3. This example provides a return value that will become the expression's value if evaluating the expression causes an error. There are three elements to code.

The instance variable is a string:

```
string is_dwvalue
```

This script for a button or other control sets the instance variable to a valid return value and accesses a DataWindow property:

```
is_dwvalue = "5"
ls_border = dw_1.Object.id.Border
```

The Error event script includes these lines:

```
action = ExceptionSubstituteReturnValue!
returnvalue = is_dwvalue
```

During execution, if the id column does not exist or some other error occurs, then the expression returns the value assigned to is_dwvalue. As a result, ls_border is set to a valid border value, here the string "5".

Syntax for property expressions

This section describes:

- Basic syntax for property expressions
- Syntax for property expressions involving nested objects (derived from the basic syntax)

Basic syntax

Description

DataWindow property expressions use dot notation to specify the objects, objects within objects, and properties that you want to access.

Syntax

dwcontrol.**Object**.*dwobjectname* {.**Object**.*dwobjectnameproperty* } {.*propertyvalue* } { = value }

Argument	Description
dwcontrol	The name of the DataWindow control or child DataWindow in which you want to get or set properties
Object	Object indicates that subsequent elements refer to the object within *dwcontrol* or *dwobjectname*
dwobjectname	An object within the DataWindow. Possible values are DataWindow (for properties that apply to the whole DataWindow) or the name of a bitmap, column, computed field, graph, line, oval, rectangle, round rectangle, report, TableBlob, or text object If *dwobjectname* is a column with the DropDownDataWindow style, a report object, or an OLE object, you can specify another Object keyword and *dwobjectname* to refer to properties of objects within the nested object. You can specify Object.*dwobjectname* as many times as needed to refer to a deeply nested report FOR INFO For nested syntax, see "Syntax for nested objects" on page 210
dwobjectnameproperty	A property that applies to *dwobjectname*. If the property requires additional qualifying properties, list the additional properties, separating them with a dot FOR INFO For lists of applicable properties, see the Property tables at the beginning of Chapter 5, "DataWindow Object Properties"

Argument	Description
value	A string whose value is to be assigned to the property Although *value* is stored as a string, if the property value is a number *value* can either be a string whose value is a number or a numeric data type. The value is stored as a string If the property value is a yes or no value, *value* can be either a string whose value is "yes" or "no" or a boolean value (TRUE or FALSE). The value is stored as "yes" or "no" strings If the property value can be an expression, then *value* can be a string that takes the form: *defaultvalue*~t *DataWindowpainterexpression* where: ♦ *Defaultvalue* is any value that is allowed for *property* ♦ *DataWindowpainterexpression* is an expression that can include names of objects in the DataWindow and DataWindow painter functions ♦ *Defaultvalue* and *DataWindowpainterexpression* are separated by a tab character (~t) FOR INFO For examples of DataWindow painter expressions, see "Using DataWindow painter expressions" on page 218

Return value

Any. The data type of the value is string.

FOR INFO For more information about the expression's data type, see "Data types of property expressions" on page 204.

Examples

Example 1. In this statement, the boolean value FALSE is stored as the string "no":

```
dw_1.Object.DataWindow.ReadOnly = FALSE
```

This statement displays the value of the ReadOnly property (either "yes" or "no") in the StaticText st_status:

```
st_status.Text = dw_1.Object.DataWindow.ReadOnly
```

When you test the value of a property in a relational expression, you must compare your test value to the stored values. For ReadOnly, stored values are yes or no, not boolean TRUE or FALSE:

```
IF dw_1.Object.DataWindow.Readonly = 'yes' THEN
```

This statement fails because the expression is not boolean:

```
IF dw_1.Object.DataWindow.Readonly THEN // Not valid
```

Example 2. Valid values for the Visible property are 0 and 1. You can set the property to numbers, yes and no, or TRUE and FALSE. Therefore, these three statements are equivalent:

```
dw_1.Object.street.Visible = FALSE
dw_1.Object.street.Visible = "NO"
dw_1.Object.street.Visible = 0
```

Example 3. This example tests whether the X property contains a constant (which can be converted to a number) or a DataWindow painter expression. A default value of 50 is specified if the property contains an expression the script can't convert:

```
integer li_x
IF IsNumber( dw_1.Object.id.X ) THEN
    li_x = Integer( dw_1.Object.id.X )
ELSE
    li_x = 50
END IF
```

Example 4. This script sets the X property to a DataWindow painter expression. The expression indents ID values for IDs less than 10:

```
string modstring, ls_x
ls_x = "50"
modstring = ls_x + "~t" + &
    "If(id > 10, " + ls_x + "," + &
    String(li_x + 20 ) + ")"
dw_1.Object.id.X = modstring
```

Example 5. This example makes three columns updatable and reports the value of the Update property in the StaticText st_status. The reported value is "yes," not TRUE:

```
dw_1.Object.id.Update = TRUE
dw_1.Object.street.Update = TRUE
dw_1.Object.last_name.Update = TRUE

st_status.Text = "Updateable: id " + &
    dw_1.Object.id.Update + ", street " + &
    dw_1.Object.street.Update + ", last_name " + &
    dw_1.Object.last_name.Update
```

Example 6. This example checks whether the id column is set up as a spin control. If so, it sets the spin range to 0 through 10:

```
IF dw_1.Object.id.EditMask.Spin = "yes" THEN
   dw_1.Object.id.EditMask.SpinRange = "0~~~~10"
END IF
```

Syntax for nested objects

Description

DataWindow property expressions involving nested objects can become complex. Nested objects include composite or related nested reports and child DataWindows associated with DropDownDataWindow columns. Related nested and composite reports can include their own nested objects. You can extend the dot notation to refer to any level of nesting.

Syntax

dwcontrol.Object.*nestedobjectname* { [*row*] } .Object.*dwobjectname*.
property { .*property* } { = *value* }

Argument	Description
dwcontrol	The name of the DataWindow control or child DataWindow in which you want to get or set properties
Object	The Object keyword indicates that subsequent elements refer to the object within *dwcontrol*
nestedobjectname	The name of a DropDownDataWindow column, nested report, or OLE object within the DataWindow object in *dwcontrol* **About nested reports** A nested report can be one of a group of reports in the Composite presentation style or a nested report included in a base report, which is associated with a specific row
row	When *nestedobjectname* is a nested report in a base report, the number of the row the report is associated with If the report is in a band other than the detail band, it is still associated with a row (see Usage below)

Argument	Description
dwobjectname	The name of an object within the nested DataWindow object. Possible values are DataWindow (for properties that apply to the whole DataWindow) or the name of a Bitmap, Button, Column, Computed field, Graph, GroupBox, Line, Oval, Rectangle, RoundRectangle, Report, TableBlob, or Text object If *dwobjectname* is a column with the DropDownDataWindow style, a Report object, or an OLE object, you can specify an additional Object keyword and *dwobjectname* to refer to properties of objects within the nested object. You can specify Object.*dwobjectname* as many times as needed to refer to a deeply nested object
property	A property that applies to *dwobjectname*. If the property requires additional qualifying properties, list the additional properties, separating them with a dot FOR INFO For lists of applicable properties, see the Property tables at the beginning of Chapter 5, "DataWindow Object Properties"
value	A string whose value is to be assigned to the property. FOR INFO For examples of DataWindow painter expressions, see "Using DataWindow painter expressions" on page 218

Return value

Any. The data type of the value is string.

FOR INFO For more information about the expression's data type, see "Data types of property expressions" on page 204.

Usage

A nested report within a base report is usually in the detail band, and each instance of the report is associated with a row. The property expression must include a row number to identify which report to access. If the nested report is in a band other than detail, there may be only one or a few instances of the report, but it is still associated with a row. The expression must include a row number that has an instance of the report.

The following table lists the band and the row that is associated with the report:

If the report is in this band	This row is associated with the report
detail	The specified row
header	The first row on the page. Onscreen, this is the first row visible in the DataWindow body

If the report is in this band	This row is associated with the report
footer	The last row on the page. Onscreen, this is the last row visible in the DataWindow body
header.n (group header)	The first row of the group
trailer.n (group trailer)	The last row of the group
summary	The last row in the report

Examples

Example 1. Suppose that a DataWindow has the Composite presentation style and includes a report called rpt_employee. The report includes a column emp_id. This expression gets the validation expression for the column:

```
string ls_valid
ls_valid = dw_composite.Object.rpt_employee.&
   Object.emp_id.Validation
```

Example 2. In a Composite DataWindow, one of the reports rpt_1 has a graph gr_1. This example turns on grid lines for the category axis of that graph. The example sets an instance variable to a default value of "not found." If the expression fails and triggers the Error event, the ExceptionSubstituteReturnValue! action causes the text "not found" to be returned so that the second assignment succeeds:

```
is_dwvalue = "not found"
dw_1.Object.rpt_1.Object.&
   gr_1.Category.MajorGridline = 5
st_status.Text = dw_1.Object.rpt_1.Object.&
   gr_1.Category.MajorGridline
```

The script for the Error event includes these lines:

```
action = ExceptionSubstituteReturnValue!
returnvalue = is_dwvalue
```

Example 3. Suppose that a DataWindow called dw_emp is a base report with employee information. The detail band includes a nested report of salary history called rpt_salary. This means there is a separate report with its own properties in each row.

The script checks whether the employee belongs to management (the value in the rank column in the base report is M). If so, the script assigns a DataWindow painter expression to the Color property of the salary column in the rpt_salary nested report. The expression highlights salaries that are over $60,000 in red. Another statement sets the salary column's Mode property so the color change will be visible:

```
integer li_row

FOR li_row = 1 to RowCount( )
   IF dw_emp.Object.rank.Data = "M" THEN

      dw_emp.Object.rpt_salary[li_row].Object.&
         salary.Background.Color = &
         '255 ~t If(salary > 60000, 255, 0)'

      dw_emp.Object.rpt_salary[li_row].Object.&
         salary.Background.Mode = 255

   END IF
NEXT
```

Example 4. In this example there is a graph in the summary band of a base report called dw_emp. The graph is a nested report called rpt_graph_salaries. Although the graph is not related to a particular row, you still need to provide the row number associated with the summary band when you refer to its properties. This statement turns on autoscaling for the values axis:

```
dw_emp.Object.rpt_graph_salaries.Object.&
   gr_1.Values.AutoScale = 1
```

Example 5. If a column has a DropDownDataWindow edit style, there are properties that affect the column's appearance. Using nested object syntax, you can also change properties of the child DataWindow for the column. In this example, the DataWindow dw_gift allows a clerk at a nonprofit organization to record donations. The clerk can pick a standard donation amount from a dropdown DataWindow.

This example makes the dropdown DataWindow column called amount a required value and changes the display format for the dollars column in the child DataWindow:

```
dw_gift.Object.amount.dddw.Required = "Yes"
dw_gift.Object.amount.Object.dollars.Format = "$#,##0"
```

Using the DWObject variable

A DWObject object is an object that exists within a DataWindow object. Each column, computed field, text object, or drawing object is a DWObject.

A DWObject reference allows you to refer directly to objects within a DataWindow.

You can use a DWObject variable to simplify DataWindow property and data expressions. A DWObject variable takes the place of several elements of the object's dot notation.

The following syntaxes and examples show how using a DWObject variable affects property and data expressions.

Property expressions

The simple syntax for a property expression is:

dwcontrol.Object.*dwobjectname.property*

You can use a DWObject variable to refer to *dwobjectname*.

If the code declares a DWObject variable and assigns the object within the DataWindow to the variable, using syntax like this:

DWObject *dwobjectvar*

dwobjectvar = *dwcontrol*.Object.*dwobjectname*

The syntax of the expression itself becomes:

dwobjectvar.property

For example, if the DataWindow had a column named empname, a text object named t_emplabel, and a computed field named cf_average, you could make the following assignments:

```
DWObject dwo_column, dwo_text, dwo_compute
dwo_column = dw_1.Object.empname
dwo_text = dw_1.Object.t_emplabel
dwo_compute = dw_1.Object.cf_average
```

Data expressions

One syntax variation for data expressions uses column names. The syntax is:

dwcontrol.Object.*columnname* {.*buffer* } {.*datasource* } { [*rownum*] }

You can use a DWObject variable to refer to the column.

If the code declares a DWObject variable and assigns the object within the DataWindow to the variable, using syntax like this:

DWObject *dwobjectvar*

dwobjectvar = *dwcontrol*.Object.*columnname*

The syntax of the expression itself becomes:

dwobjectvar. {*.buffer* } {*.datasource* } { [*rownum*] }

DWObject variables in scripts

You can get better performance by using a DWObject variable to resolve the object reference in a DataWindow property or data expression. Evaluating the reference once and reusing the resolved reference is more efficient than fully specifying the object reference again.

This technique yields the most benefit if your application uses compiled code or if you are using a DataWindow expression in loop.

For example, this code is not optimized for best performance, because the fully specified data expression within the loop must be resolved during each pass:

```
integer li_data
FOR li_cnt = 1 to 100
    li_data = dw_1.Object.emp_salary[li_cnt]
    .. // Code to process data value
NEXT
```

This code has been optimized. The reference to the object within the DataWindow (emp_salary) is resolved once before the loop begins. The reference stored in the DWObject variable is reused repeatedly in the loop:

```
integer li_data
DWObject dwo_empsalary

dwo_empsalary = dw_1.Object.emp_salary

FOR li_cnt = 1 to 100
    li_data = dwo_empsalary[li_cnt]
    .. // Code to process data value
NEXT
```

DWObject arguments for DataWindow events

Several DataWindow events pass a DWObject argument called dwo to the event script. The value is an resolved reference to an object within the DataWindow having something to do with the user's action that triggered the event. Often it is the column the user is changing or the object the user clicked.

What type of DWObject?

You can use DataWindow properties to find out more about the object stored in dwo. The first step is to find out the object's type so that subsequent statements will use properties that are appropriate for the object type. If an expression uses a property that doesn't correspond to the object's type, it will trigger the Error event:

```
ls_type = dwo.Type
```

The possible values that can be assigned to ls_type are:

bitmap
button
column
compute
graph
groupbox
line
ole
ellipse (for oval)
rectangle
roundrectangle
report
tableblob
text
datawindow (*when the user doesn't click a specific object*)

You can write a CHOOSE CASE statement for the expected types.

After you've determined the type, you can get more details about the specific object.

Examples

If the object is a column, you can get the column name with this statement:

```
ls_name = dwo.Name
```

If the object is a column, you can get data from the whole column or from specific rows. In this statement, row is another argument passed to the event so the value in ls_data is the data in the row and column the user clicked. In this example, if the value is not a string, an error occurs:

```
ls_data = dwo.Data[row]
```

This statement assigns a new value to the row and column the user clicked. The assignment does not trigger the ItemChanged event and bypasses validation. If the column is not numeric, an error occurs:

```
dwo.Data[row] = 41
```

This statement gets all the data in the column the user clicked. The data is stored as an array in the Any variable:

```
Any ls_data
ls_data = dwo
```

This statement gets data in the column from selected rows. The data is stored as an array in the Any variable:

```
Any ls_data
ls_data = dwo.Selected
```

Using DataWindow painter expressions

When a property's value can be an expression, you can make the object's appearance or other properties depend on other information in the DataWindow.

A DataWindow painter expression can include:

- Operators
- The names of objects within the DataWindow, especially column names
- DataWindow painter functions
- User-defined functions

Different formats for the expression

When you assign an expression in the painter, you specify the expression.

DataWindowpainterexpression

When you assign an expression in a script, you specify a default value, a tab, and the expression.

defaultvalue~t *DataWindowpainterexpression*

Examples

Expressions tab in the painter This expression for a column called emp_lname is applied to the Background.Color property. It causes the name's background to be light gray (15790320) if the current row (person) uses the day care benefit. If not, the background color is set to white:

```
If(bene_day_care = 'Y', 15790320, 1677215)
```

In a script The expression assigned to the Background.Color property includes a default value and includes nested quotes. (Tildes aren't needed, because the example nests single quotes inside double quotes):

```
dw_1.Object.emp_lname.Background.Color = "16777215 ~t
If(bene_day_care = 'Y', 15790320, 16777215)"
```

More examples in the DataWindow painter and in scripts

All of these expressions are specified on the Expressions tab page of the Properties dialog box for an object in the DataWindow, except where noted.

Border property The expression applied to the Border property of the salary_plus_benefits column displays a border around salaries over $60,000:

```
If(salary_plus_benefits > 60000, 1, 0)
```

This statement changes the expression in a script:

```
dw_1.Object.salary_plus_benefits.Border = &
    "If(salary_plus_benefits > 60000, 1, 0)"
```

Font.Weight property for a column To make out-of-state (not in Massachusetts) names and numbers bold in a phone list, apply this expression to the name and phone_number columns. The state column must be part of the data source but it doesn't have to be displayed:

```
If(state = 'MA', 400, 700)
```

This statement changes the expression in a script:

```
dw_1.Object.name.Font.Weight = &
    "If(state = 'MA', 400, 700)"
dw_1.Object.phone_number.Font.Weight = &
    "If(state = 'MA', 400, 700)"
```

Brush.Color property for a rectangle This expression, applied to a rectangle drawn around all the columns in a tabular report, causes alternate row to be shaded (a graybar effect). Make sure the columns and computed fields have a transparent background. The expression Mod(GetRow( ), 2) = 1 distinguishes odd rows from even rows:

```
If(Mod(GetRow(), 2) = 1, 16777215, 15790320)
```

This statement changes the expression in a script:

```
dw_1.Object.rectangle_1.Brush.Color = &
    "If(Mod(GetRow(), 2) = 1, 16777215, 15790320)"
```

Brush.Color and Brush.Hatch properties for a rectangle To highlight employees whose review date is approaching, draw a rectangle behind the row. This expression for the rectangle's Brush.Color property makes the rectangle light gray for employees for whom the month of the start date matches the current month or the next month:

```
If(month(start_date) = month(today())
or month(start_date) = month(today()) + 1
or (month(today()) = 12 and month(start_date) = 1),
12632256, 16777215)
```

A similar expression for the Brush.Hatch property makes the fill pattern of the rectangle Bdiagonal (1) for review dates that are approaching. Otherwise, the rectangle is transparent (7) so that it doesn't show:

```
If(month(start_date) = month(today())
or month(start_date) = month(today()) + 1
or (month(today()) = 12 and month(start_date) = 1),
1, 7)
```

You can also set the Pen.Color and Pen.Style properties to affect the outline of the rectangle.

If you wanted to change the Brush.Color property in a script, instead of on the Expressions tab, the code would look like this:

```
dw_1.Object.rectangle_1.Brush.Color = &
    "'16777215 ~t " + &
    "If(month(start_date) = month(today()) " + &
    "or month(start_date) = month(today()) + 1 " + &
    "or (month(today()) = 12 " + &
    "and month(start_date) = 1), 12632256, 16777215)'"
```

Font.Height property for a rectangle This expression applied to the Font.Height property of a text object makes the text object in the first row of a DataWindow larger than it appears in other rows. Make sure the borders of the text object are large enough to accommodate the increased size:

```
If(GetRow() = 1, 500, 200)
```

This statement changes the expression for the text object t_desc in a script:

```
dw_1.Object.t_desc.Font.Height = &
    "If(GetRow() = 1, 500, 200)"
```

For more information

For information about evaluating DataWindow painter expressions in scripts, see Chapter 1, "Operators and Expressions". For information about DataWindow painter functions in property expressions, see "Using DataWindow painter and InfoMaker functions" on page 16.

CHAPTER 5

DataWindow Object Properties

About this chapter

This chapter describes the properties that control the appearance and behavior of a DataWindow.

Contents

Overview of DataWindow object properties

There are many ways you can affect the values of DataWindow object properties during execution:

- There are several functions that get and set specific properties, and you can use the general-purpose Describe and Modify functions to get and set property values.
- A subset of the properties can be used with the SyntaxFromSQL function to generate DataWindow source code. You can use that code in the Create function to create new DataWindows.
- A subset of the properties can be addressed in an object's property sheet (Expressions tab page) in the DataWindow painter workspace. You can enter expressions (conditional) to set properties at execution time.

Summary tables in the first part of this chapter

The tables in the first part of this chapter list the properties for each object within a DataWindow object, with short descriptions. There are also tables for SyntaxFromSQL object keywords. After the first table of DataWindow properties, the tables are alphabetical by object and keyword name.

An annotation at the beginning of the table and checkmark columns identifies whether you can use that property with Modify (*M*) or SyntaxFromSQL (*S*). When (*exp*) is included in the description, you can specify a DataWindow painter expression as the value for that property. A DataWindow painter expression lets you specify conditions for determining the property value.

You can get the value of all properties in all tables
You can get the value of (Describe) all properties listed in all tables. The tables do not show checkmarks to account for this capability.

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 painter expression for the value.

FOR INFO For more information and examples of setting properties, see Chapter 4, "Accessing DataWindow Properties in Scripts" and the Describe, Modify, and SyntaxFromSQL functions in the *PowerScript Reference*.

Objects in a DataWindow and their properties

In this section, tables for DataWindow objects and SyntaxFromSQL keywords list the properties that apply to the objects and keywords.

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 SyntaxFromSQL.When (*exp*) is included in the description, you can specify a DataWindow painter expression as the value for that property.

Property for the DataWindow	M	S	Description
Attributes			All general properties
Bands			List of bands
Bandname.property	x		Color, height, and so on for a band, where *bandname* is Detail, Footer, Header, Summary, or Trailer
Bandname.Text	x		Rich text content where *bandname* is Detail, Footer, or Header
Color	x	x	Background color
Column.Count			Number of columns
Crosstab.property	x		Settings for a crosstab DataWindow
Data			Description of data
Data.HTMLTable			Description of the data in the DataWindow in HTML table format
Detail.property	x		Color, height, and so on for the detail band
EditMask.property	x		Settings for EditMask edit style
FirstRowOnPage			The row number of the first displayed row
Font.Bias	x		Treat fonts as display or printer
Footer.property	x		Color, height, and so on for the footer band (see *Bandname.property* in this table)
Grid.ColumnMove	x		Whether the user can drag to reposition columns
Grid.Lines	x		Options for lines in grid DataWindow and crosstab
Header.#.property	x		Color, height, and so on for a group's header band
Header.property	x		Color, height, and so on for the header band
Help.property	x		Help settings for DataWindow actions

Property for the DataWindow	M	S	Description
HorizontalScrollMaximum			Width of scroll box in the horizontal scrollbar
HorizontalScrollMaximum2			Width of second scroll box when scrollbar is split
HorizontalScrollPosition	x		Position of the scroll box in the scrollbar
HorizontalScrollPosition2	x		Position of scroll box in second split scrollbar
HorizontalScrollSplit	x		The position of the split in the scrollbar
HTMLTable.property	x		Settings for the display of DataWindow data when displayed in HTML table format
Label.property	x	x	Settings for the Label presentation style
LastRowOnPage			The last visible row on the page
Message.Title	x	x	The title of the dialog box that displays errors
Nested			Whether the DataWindow has nested reports
Objects			The objects in the DataWindow
OLE.Client.property	x		Settings for the DataWindow as OLE client
Pointer	x		(*exp*) The pointer when over the DataWindow
Print.Buttons	x		Whether buttons display on the printed output
Print.Preview.Buttons	x		Whether buttons display in print preview
Print.property	x	x	Various settings for printing
Printer			The currently selected printer
Processing			Processing required by the presentation style
QueryMode	x		Whether the DataWindow is in query mode

Property for the DataWindow	M	S	Description
QuerySort	x		Whether to sort the result set from the query
ReadOnly	x		Whether the DataWindow is read-only
Retrieve.AsNeeded	x		Whether to retrieve data only as needed
RichText.property	x		Settings for a RichText DataWindow
Row.Resize	x		Whether user can change the height of rows
Rows_Per_Detail		x	Number of rows in each column of N-Up style
Selected	x		List of selected objects
Selected.Data			List of selected data
Selected.Mouse	x		Whether user can use the mouse to select
ShowDefinition	x		(*exp*) Display column names instead of data
Sparse	x		(*exp*) The repeating columns to be suppressed
Storage			The amount of storage used by DataWindow
Summary.property	x		Color, height, and so on for the summary band
Syntax			The syntax of the DataWindow
Syntax.Data			The data of the DataWindow in parse format
Syntax.Modified	x		Whether the syntax has been modified
Table.property	x		Various settings for the database
Timer_Interval	x	x	The milliseconds between timer events
Trailer.#.property	x		Color, height, and so on for a group's trailer band
Units		x	The unit of measure for the DataWindow
VerticalScrollMaximum			The height of the scroll box in the scrollbar

Property for the DataWindow	M	S	Description
VerticalScrollPosition	x		The position of the scroll box in the scrollbar
Zoom	x		The scaling percentage of the DataWindow

Properties for Bitmap objects

An x in the M (Modify) column means you can change the property. When (*exp*) is included in the description, you can specify a DataWindow painter expression as the value for that property.

Property for a Bitmap	M	Description
Attributes		A list of the properties of the bitmap
Band		The band containing the bitmap
Border	x	(*exp*) The type of border around the bitmap
Filename	x	(*exp*) The file containing the bitmap
Height	x	(*exp*) The height of the bitmap
HideSnaked	x	Whether the object appears once per page when printing newspaper columns
Invert	x	(*exp*) Whether the colors are displayed inverted
Moveable	x	Whether the user can move the bitmap
Name		The name of the bitmap object
Pointer	x	(*exp*) The pointer image when it is over the bitmap
Resizeable	x	Whether the user can resize the bitmap
SlideLeft	x	(*exp*) Whether the bitmap moves left to fill in empty space
SlideUp	x	(*exp*) How the bitmap moves up to fill in empty space
Tag	x	(*exp*) The tag text for the bitmap
Type		The object's type, which is bitmap

Property for a Bitmap	M	Description
Visible	x	(*exp*) Whether the bitmap object is visible
Width	x	(*exp*) The width of the bitmap
X	x	(*exp*) The x coordinate of the bitmap
Y	x	(*exp*) The y coordinate of the bitmap

Properties for Button objects

An x in the M (Modify) column means you can change the property. When (*exp*) is included in the description, you can specify a DataWindow painter expression as the value for that property.

Property for a Button	M	Description
Action	x	(*exp*) The action a user can assign to the button
Attributes		A list of the properties of the button object
Background.property	x	(*exp*) Background settings for the button object
Band		The band containing the button object
Color	x	(*exp*) The text color
DefaultPicture	x	Whether or not the action's default picture is to be used on the button (user defined action has no picture)
Font.property	x	(*exp*) Font settings for the text
HTextAlign	x	(*exp*) How the text in the button is horizontally aligned. Values are: 0 (center), 1 (left), 2 (right)
Height	x	(*exp*) The height of the button object
HideSnaked	x	Whether the button object appears once per page when printing newspaper columns
Moveable	x	Whether the user can move the button object
Name		The name of the button object
PictureName	x	(*exp*) Name of the file containing the picture to be used on the button (if not specified, just the text is used)

Property for a Button	M	Description
Pointer	x	(*exp*) The pointer image when it is over the button object
Resizeable	x	Whether the user can resize the button object
SlideLeft	x	(*exp*) Whether the button object moves left to fill in empty space
SlideUp	x	(*exp*) How the button object moves up to fill in empty space
SuppressEventProcessing	x	Whether or not ButtonClicked and ButtonClicking events are fired for this particular button
Tag	x	(*exp*) The tag text for the button object
Text	x	(*exp*) The displayed text
Type		The object's type, which is button
VTextAlign	x	(*exp*) How the text in the button is vertically aligned. Values are: 0 (center), 1 (top), 2 (bottom), 3 (multiline)
Visible	x	(*exp*) Whether the button object is visible
Width	x	(*exp*) The width of the button object
X	x	(*exp*) The x coordinate of the button object
Y	x	(*exp*) The y coordinate of the button object

Properties for Column objects

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 SyntaxFromSQL. When (*exp*) is included in the description, you can specify a DataWindow painter expression as the value for that property.

Property for a Column	M	S	Description
Accelerator	x		(*exp*) The accelerator key for the column
Alignment	x		(*exp*) The alignment of the column's text
Attributes			A list of the properties of the column

Property for a Column	M	S	Description
Background.property	x	x	(*exp*) Background settings for the column
Band			The band containing the column
BitmapName			Whether the column's contents names a bitmap that will be displayed instead of the text
Border	x	x	(*exp*) The type of border around the column
CheckBox.property	x		Settings for CheckBox edit style
Color	x	x	(*exp*) The text color
ColType			The column's data type
Criteria.property	x		Settings for column in Prompt for Criteria dialog box
dbName	x		The name of the database column
dddw.property	x		Settings for DropDownDataWindow edit style
ddlb.property	x		Settings for DropDownListBox edit style
Edit.property	x	x	Settings for Edit edit style
EditMask.property	x		Settings for EditMask edit style
Font.property	x	x	(*exp*) Font settings for the column text
Format	x		(*exp*) The column's display format
Height	x		(*exp*) The height of the column
Height.AutoSize	x		Whether column height is adjusted to fit the data
HideSnaked	x		Whether the object appears once per page when printing newspaper columns
Identity	x		Whether the DBMS sets the column's value
ID			The number of the column
Initial	x		The initial value in the column for a new row
Key	x		Whether column is part of the table's primary key
LineRemove	x		Whether to remove line of text when the column is not visible

Property for a Column	M	S	Description
Moveable	x		Whether the user can move the column
Multiline	x		Whether the column can be multiline
Name			The name of the column
Pointer	x		(*exp*) The pointer's image when it is over column
Protect	x		(*exp*) Whether column is protected from changes
RadioButtons.property	x		Settings for RadioButton edit style
Resizeable	x		Whether the user can resize the column
SlideLeft	x		(*exp*) Whether column moves left to fill in space
SlideUp	x		(*exp*) How the column moves up to fill in space
TabSequence	x		The position of the column in the tab order
Tag	x		(*exp*) The tag text for the column
Type			The object's type, which is Column
Update	x		Whether the column is updatable
Validation	x		(*exp*) The validation expression for the column
ValidationMsg	x		(*exp*) The message displayed when validation fails
Values (for columns)	x		The values in the column's code table
Visible	x		(*exp*) Whether the column object is visible
Width	x		(*exp*) The width of the column
Width.Autosize	x		Whether width adjusts to the data
X	x		(*exp*) The x coordinate of the column
Y	x		(*exp*) The y coordinate of the column

Properties for Computed Field objects

An x in the M (Modify) column means you can change the property. When (*exp*) is included in the description, you can specify a DataWindow painter expression as the value for that property.

Property for a computed field	M	Description
Alignment	x	(*exp*) The alignment of the computed field's text
Attributes		A list of the properties of the computed field
Background.property	x	(*exp*) Background settings for the computed field
Band		The band containing the computed field
Border	x	(*exp*) The type of border around the computed field
Color	x	(*exp*) The text color
ColType		The column's data type
Expression	x	The expression for the computed field
Footer.property	x	(*exp*) Font settings for the computed field
Format	x	(*exp*) The computed field's display format
Height	x	(*exp*) The height of the computed field
Height.AutoSize	x	Whether the computed field's height is adjusted to fit the data
HideSnaked	x	Whether the object appears once per page when printing newspaper columns
LineRemove	x	Whether to remove line of text when the computed field is not visible
Moveable	x	Whether the user can move the computed field
Multiline	x	Whether the column can be multiline
Name		The name of the computed field
Pointer	x	(*exp*) The pointer image when it is over the computed field
Resizeable	x	Whether the user can resize the computed field
SlideLeft	x	(*exp*) Whether the computed field moves left to fill in space

Property for a computed field	M	Description
SlideUp	x	(*exp*) How the computed field moves up to fill in empty space
Tag	x	(*exp*) The tag text for the computed field
Type		The object's type, which is Compute
Visible	x	(*exp*) Whether the computed field object is visible
Width	x	(*exp*) The width of the computed field
Width.Autosize	x	Whether width adjusts to the data
X	x	(*exp*) The x coordinate of the computed field
Y	x	(*exp*) The y coordinate of the computed field

Properties for Graph objects

An x in the M (Modify) column means you can change the property. When (*exp*) is included in the description, you can specify a DataWindow painter expression as the value for that property.

Property for a Graph	M	Description
Attributes		A list of the properties of the graph
Axis	x	(*exp*) List of items (categories, series, or values) for the axis
Axis.property	x	(*exp*) Properties for a graph axis
Axis.DispAttr	x	(*exp*) Display properties for an axis (see DispAttr.*fontproperty* in this table)
BackColor	x	(*exp*) The background color of the graph
Band		The band containing the graph
Border	x	(*exp*) The type of border around the graph
Category	x	(*exp*) List of categories for the axis (see *Axis* in this table)
Category.property	x	(*exp*) Properties for the Category axis (see *Axis.property* in this table)

Property for a Graph	M	Description
Category.DispAttr	x	(*exp*) Display properties for the Category axis (see DispAttr.*fontproperty* in this table)
Color	x	(*exp*) The text color
Depth	x	(*exp*) The depth of a 3D graph
DispAttr.fontproperty	x	Font settings for various components of the graph
Elevation	x	(*exp*) The elevation of a 3D graph
GraphType	x	(*exp*) The type of graph (pie, bar, and so on)
Height	x	(*exp*) The height of the graph
HideSnaked	x	Whether the object appears once per page when printing newspaper columns
Legend	x	(*exp*) The location of the legend
Legend.DispAttr.fontproperty	x	(*exp*) Display properties for the legend
Moveable	x	Whether the user can move the graph
Name		The name of the graph object
OverlapPercent	x	(*exp*) The overlap between data markers in different series
Perspective	x	(*exp*) The distance of the graph from the front of the window
Pie.DispAttr.fontproperty	x	(*exp*) Display properties for the pie slice labels
Pointer	x	(*exp*) The pointer image when it is over the graph
Range		The rows in the DataWindow that are included in the graph
Resizeable	x	Whether the user can resize the graph
Rotation	x	(*exp*) The left-to-right rotation of a 3D graph
Series	x	(*exp*) List of series for the axis (see *Axis* in the table)
Series.property	x	(*exp*) Properties for the Series axis (see *Axis.property* in this table)

Property for a Graph	M	Description
Series.DispAttr	x	(*exp*) Display properties for the Series axis (see DispAttr.*fontproperty* in this table)
ShadeColor	x	(*exp*) The color of the back edge for 3D data markers
SizeToDisplay	x	(*exp*) Whether to size the graph to the display area
SlideLeft	x	(*exp*) Whether the graph moves left to fill in empty space
SlideUp	x	(*exp*) How the graph moves up to fill in empty space
Spacing	x	(*exp*) The gap between categories
Tag	x	(*exp*) The tag text for the graph
Title	x	(*exp*) The graph's title
Title.DispAttr.fontproperty	x	(*exp*) Display properties for the title
Type		The object's type, which is graph
Values	x	(*exp*) List of values for the axis (see *Axis* in the table)
Values.property	x	(*exp*) Properties for the Values axis (see *Axis.property* in the table)
Values.DispAttr	x	(*exp*) Display properties for the Values axis (see DispAttr.*fontproperty* in the table)
Visible	x	(*exp*) Whether the graph object is visible
Width	x	(*exp*) The width of the graph
X	x	(*exp*) The x coordinate of the graph
Y	x	(*exp*) The y coordinate of the graph

Properties for GroupBox objects

An x in the M (Modify) column means you can change the property. When (*exp*) is included in the description, you can specify a DataWindow painter expression as the value for that property.

Property for a GroupBox	M	Description
Attributes		A list of the properties of the GroupBox object
Background.property	x	(*exp*) Background settings for the GroupBox object
Band		The band containing the GroupBox object
Border	x	(exp) Border style: 2 (box), 5 (3D lowered), 6 (3D raised)
Color	x	(*exp*) The text color
Font.property	x	(*exp*) Font settings for the text
Height	x	(*exp*) The height of the GroupBox object
HideSnaked	x	Whether the GroupBox object appears once per page when printing newspaper columns
Moveable	x	Whether the user can move the GroupBox object
Name		The name of the GroupBox object
Pointer	x	(*exp*) The pointer image when it is over the GroupBox object
Resizeable	x	Whether the user can resize the GroupBox object
SlideLeft	x	(*exp*) Whether the GroupBox object moves left to fill in empty space
SlideUp	x	(*exp*) How the GroupBox object moves up to fill in empty space
Tag	x	(*exp*) The tag text for the GroupBox object
Text	x	(*exp*) The displayed text
Type		The object's type, which is GroupBox
Visible	x	(*exp*) Whether the GroupBox object is visible
Width	x	(*exp*) The width of the GroupBox object
X	x	(*exp*) The x coordinate of the GroupBox object
Y	x	(*exp*) The y coordinate of the GroupBox object

Properties for the Group keyword

You can use these properties with SyntaxFrom SQL.

Property	Description
NewPage (Group keywords)	Whether a change in a group column's value causes a page break
ResetPageCount	Whether a new value in a group column restarts page numbering

Properties for Line objects

An x in the M (Modify) column means you can change the property. When (*exp*) is included in the description, you can specify a DataWindow painter expression as the value for that property.

Property for a Line	M	Description
Attributes		A list of the properties of the line
Background.property	x	(*exp*) Background settings for the line
Band		The band containing the line
HideSnaked	x	Whether the object appears once per page when printing newspaper columns
Moveable	x	Whether the user can move the line
Name		The name of the line object
Pen.property	x	(*exp*) Appearance settings of the line
Pointer	x	(*exp*) The pointer image when it is over the line
Resizeable	x	Whether the user can resize the line
SlideLeft	x	(*exp*) Whether the line moves left to fill empty space
SlideUp	x	(*exp*) How the line moves up to fill empty space
Tag	x	(*exp*) The tag text for the line
Type		The object's type, which is Line
Visible	x	(*exp*) Whether the Line object is visible
X1, X2	x	(*exp*) The x coordinate of each end of the line

Property for a Line	M	Description
Y1, Y2	x	(*exp*) The y coordinate of each end of the line

Properties for OLE container objects

An x in the M (Modify) column means you can change the property. When (*exp*) is included in the description, you can specify a DataWindow painter expression as the value for that property.

Property for an OLE container	M	Description
Activation	x	The way the OLE object is activated
Attributes		A list of the properties of the OLE container object
Band		The band containing the OLE container object
BinaryIndex		An internal pointer
Border	x	(*exp*) The type of border around the OLE container object
ClientName	x	The name of the OLE client in the server window
ContentsAllowed	x	Whether object can be embedded, linked, or both
DisplayType	x	Whether the container displays an icon or contents
GroupBy	x	(*exp*) The grouping columns for the transferred data
Height	x	(*exp*) The height of the OLE container object
HideSnaked	x	Whether the object appears once per page when printing newspaper columns
LinkUpdateOptions	x	How a linked object is updated
Moveable	x	Whether the user can move the OLE container object
Name		The name of the OLE container object
Pointer	x	(*exp*) The pointer image when it is over the object
Range		Method for choosing the rows transferred to the OLE object
Resizeable	x	Whether the user can resize the OLE container object
SizeToDisplay	x	(*exp*) Whether the OLE container is automatically sized to the display area

Property for an OLE container	M	Description
SlideLeft	x	(*exp*) Whether the OLE container moves left to fill in space
SlideUp	x	(*exp*) How the OLE container moves up to fill in space
Tag	x	(*exp*) The tag text for the OLE container object
Target	x	(*exp*) The columns or expressions whose data is to the OLE object
Type		The object's type, which is OLE
Visible	x	(*exp*) Whether the OLE container object object is visible
Width	x	(*exp*) The width of the OLE container object
X	x	(*exp*) The x coordinate of the OLE container object
Y	x	(*exp*) The y coordinate of the OLE container object

Properties for Oval, Rectangle, and RoundRectangle objects

An x in the M (Modify) column means you can change the property. When (*exp*) is included in the description, you can specify a DataWindow painter expression as the value for that property.

Property	M	Description
Attributes		A list of the properties of the object
Background.property	x	(*exp*) Background settings for the object
Band		The band containing the object
Brush.property	x	(*exp*) Settings for fill pattern and color
Height	x	(*exp*) The height of the object
HideSnaked	x	Whether the object appears once per page when printing newspaper columns
Moveable	x	Whether the user can move the object
Name		The name of the object
Pen.property	x	(*exp*) Appearance settings of the object

Property	M	Description
Pointer	x	(*exp*) The pointer image when it is over the object
Resizeable	x	Whether the user can resize the object
SlideLeft	x	(*exp*) Whether the object moves left to fill empty space
SlideUp	x	(*exp*) How the object moves up to fill empty space
Tag	x	(*exp*) The tag text for the object
Type		The object's type, which is ellipse, rectangle, or roundrectangle
Visible	x	(*exp*) Whether the object is visible
X	x	(*exp*) The x coordinate of the object
Y	x	(*exp*) The y coordinate of the object

Additional properties for RoundRectangle objects

An x in the M (Modify) column means you can change the property. When (*exp*) is included in the description, you can specify a DataWindow painter expression as the value for that property.

Property	M	Description
EllipseHeight	x	(*exp*) The radius of the vertical part of the rounded corner
EllipseWidth	x	(*exp*) The radius of the horizontal part of the rounded corner

Properties for Report objects

An x in the M (Modify) column means you can change the property. When (*exp*) is included in the description, you can specify a DataWindow painter expression as the value for that property.

Property for a Report	M	Description
Attributes		A list of the properties of the report
Band		The band containing the report

Property for a Report	M	Description
Border	x	(*exp*) The type of border around the report
Criteria	x	The WHERE clause that relates the report to the main DataWindow
DataObject	x	The name of the DataWindow that is the nested report
Height	x	(*exp*) The height of the report
HideSnaked	x	Whether the object appears once per page when printing newspaper columns
Moveable	x	Whether the user can move the report
Name		The name of the Report object
Nest_Arguments	x	Retrieval arguments for the report
NewPage (Report objects)	x	Whether to start the report on a new page (composite only)
Pointer	x	(*exp*) The pointer image when it is over the report
Resizeable	x	Whether the user can resize the report
SlideLeft	x	(*exp*) Whether the report moves left to fill in empty space
SlideUp	x	(*exp*) How the report moves up to fill in empty space
Tag	x	(*exp*) The tag text for the report
Trail_Footer	x	Where to print the footer (composite only)
Type		The object's type, which is report
Visible	x	(*exp*) Whether the Report object is visible
Width	x	(*exp*) The width of the report
X	x	(*exp*) The x coordinate of the report
Y	x	(*exp*) The y coordinate of the report

Properties for the Style keyword

You can use these properties with SyntaxFromSQL.

Property	Description
Detail_Bottom_Margin	Bottom margin of the detail area
Detail_Top_Margin	Top margin of the detail area
Header_Bottom_Margin	Bottom margin of the header area
Header_Top_Margin	Top margin of the header area
Horizontal_Spread	Horizontal space between columns in the detail area
Left_Margin	The left margin of the DataWindow
Report	Whether the DataWindow is a read-only report
Type	The presentation style
Vertical_Size	The height of the columns in the detail area
Vertical_Spread	The vertical space between columns in the detail area

Properties for TableBlob objects

An x in the M (Modify) column means you can change the property. When (*exp*) is included in the description, you can specify a DataWindow painter expression as the value for that property.

Property for a TableBlob	M	Description
Attributes		A list of the properties of the TableBlob
Band		The band containing the TableBlob
Border	x	(*exp*) The type of border around the TableBlob
ClientName	x	The name of the OLE client in the server window
Height	x	(*exp*) The height of the TableBlob
HideSnaked	x	Whether the object appears once per page when printing newspaper columns
ID		The number of the TableBlob
KeyClause	x	(*exp*) The key clause used when retrieving the blob
Moveable	x	Whether the user can move the TableBlob
Name		The name of the TableBlob

Property for a TableBlob	M	Description
OLEClass	x	(*exp*) The name of the TableBlob's OLE column
Pointer	x	(*exp*) The pointer image when it is over the TableBlob
Resizeable	x	Whether the user can resize the TableBlob
SlideLeft	x	(*exp*) Whether the TableBlob moves left to fill empty space
SlideUp	x	(*exp*) How the TableBlob moves up to fill empty space
Tag	x	(*exp*) The tag text for the object
Template	x	(*exp*) The file used to start the OLE application
Type		The object's type, which is TableBlob
Visible	x	(*exp*) Whether the TableBlob is visible
Width	x	(*exp*) The width of the TableBlob
X	x	(*exp*) The x coordinate of the TableBlob
Y	x	(*exp*) The y coordinate of the TableBlob

Properties for Text objects

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 SyntaxFromSQL. When (*exp*) is included in the description, you can specify a DataWindow painter expression as the value for that property.

Property for text	M	S	Description
Alignment	x	x	The alignment of the text
Attributes			A list of the properties of the text object
Background.property	x	x	(*exp*) Background settings for the text object
Band			The band containing the text object
Border	x	x	(*exp*) The type of border around the text object
Color	x	x	(*exp*) The text color
Font.property	x	x	(*exp*) Font settings for the text
Height	x		(*exp*) The height of the text object

Property for text	M	S	Description
Height.AutoSize	x		Whether the object's height is adjusted to fit the data
HideSnaked	x		Whether the object appears once per page when printing newspaper columns
Moveable	x		Whether the user can move the text object
Name			The name of the text object
Pointer	x		(*exp*) The pointer image when it is over the text object
Resizeable	x		Whether the user can resize the text object
SlideLeft	x		(*exp*) Whether the text object moves left to fill space
SlideUp	x		(*exp*) How the text object moves up to fill empty space
Tag	x		(*exp*) The tag text for the text object
Text	x		(*exp*) The displayed text
Type			The object's type, which is Text
Visible	x		(*exp*) Whether the object is visible
Width	x		(*exp*) The width of the text object
X	x		(*exp*) The x coordinate of the text object
Y	x		(*exp*) The y coordinate of the text object

Title keyword

You can use this property with SyntaxFromSQL.

Property	Description
Title("string")	The title for the DataWindow

Alphabetical list of properties

Accelerator

Description

The accelerator key that a user can press to select a column in the DataWindow object.

Applies to

Column objects

Syntax

Dot notation:

dw_control.Object.*columnname*.Accelerator

Describe and Modify argument:

"*columnname*.Accelerator { = ' *acceleratorkey* ' }"

Parameter	Description
columnname	The name of the column for which you want to get or set the accelerator key
acceleratorkey	(*exp*) A string expression whose value is the letter that will be the accelerator key for *columnname*. Acceleratorkey can be a quoted DataWindow painter expression

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 Set the value on the Column object property sheet, Edit tab.

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 object that labels the column.When you enter the text object's label, precede the character you want underlined with an ampersand (&).

Accelerator keys and edit styles To use an accelerator key with the CheckBox or RadioButton edit style, select the Edit edit style and specify the accelerator there.

Examples

```
dw_1.Object.emp_name.Accelerator = 'A'

ls_data = dw_1.Describe("emp_name.Accelerator")

dw_1.Modify("emp_name.Accelerator='A'")
```

Action

Description

The action a user can assign to a button object.

Applies to

Button objects

Syntax

Dot notation:

dw_control.Object.*buttonname*.Action

Describe and Modify argument:

"*buttonname*.Action { = ' *value* ' }"

Parameter	Description
buttonname	The name of the button for which you want to assign an action
value	The action value assigned to the button. Values are listed in the following table

Value	Action	Description	Value returned to ButtonClicked event
11	AppendRow	Inserts row at the end	Row number of newly inserted row
3	Cancel	Cancels a retrieval that has been started with the option to yield	0
10	DeleteRow	If button is in detail band, deletes row associated with button; otherwise, deletes the current row	1 if successful -1 if an error occurs
9	Filter	Displays Filter dialog box and filters as specified	Number of rows filtered Number < 0 if an error occurs

Value	Action	Description	Value returned to ButtonClicked event
12	InsertRow	If button is in detail band, inserts row using row number associated with the button; otherwise, inserts row using the current row	Row number of newly inserted row
6	PageFirst	Scrolls to the first page	1 if successful -1 if an error occurs
7	PageLast	Scrolls to the last page	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
4	PageNext	Scrolls to the next page	The row displayed at the top of the DataWindow control when the scrolling is complete or attempts to go past the first row -1 is an error occurs
5	PagePrior	Scrolls to the prior page	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
16	Preview	Toggles between preview and print preview	0
17	PreviewWith Rulers	Toggles between rulers on and off	0
15	Print	Prints one copy of the DataWindow object	0

Value	Action	Description	Value returned to ButtonClicked event
20	QueryClear	Removes the WHERE clause from a query (if one was defined)	0
18	QueryMode	Toggles between query mode on and off	0
19	QuerySort	Specifies sorting criteria (forces query mode on)	0
2	Retrieve	Retrieves rows from the database. The option to yield is not automatically turned on	Number of rows retrieved
1	Retrieve (Yield)	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	Number of rows retrieved
14	SaveRowsAs	Displays Save As dialog box and saves rows in the format specified	Number of rows filtered
8	Sort	Displays Sort dialog box and sorts as specified	1 if successful -1 if an error occurs
13	Update	Saves changes to the database. If the update is successful, a COMMIT is issued. If the update fails, a ROLLBACK is issued	1 if successful -1 if an error occurs
0	UserDefined	(Default) Allows for programming of the ButtonClicked and ButtonClicking events with no intervening action occurring	Return code from the user's coded event script

Usage

In the painter Set the value on the Button object property sheet, General tab.

Examples

```
dw_1.Object.b_retrieve.Action = "2"

setting = dw_1.Describe("b_retrieve.Action")

dw_1.Modify("b_retrieve.Action = '2'")
```

Activation

Description

The way the server for the OLE object in the container is activated. Choices include letting the user activate the object by double-clicking or putting activation under program control.

Applies to

OLE objects

Syntax

Dot notation:

dw_control.Object.*oleobjectname*.Activation

Describe and Modify argument:

"*oleobjectname*.Activation { = ' *activationtype* ' }"

Parameter	Description
oleobjectname	The name of the OLE container object for which you want to get or set the activation method
activationtype	(*exp*) A number specifying the method of activation for the OLE object. *Activationtype* can be a quoted DataWindow painter expression. Values are: 0 — The object has to be activated with the Activate function 1 — The user can activate the object by double-clicking on it 2 — The object activates when the container gets focus

Usage

In the painter Set the value using the Object property sheet, Options tab.

Examples

```
dw_1.Object.ole_report.Activation

ls_data = dw_1.Describe("ole_report.Activation")

dw_1.Modify("ole_report.Activation='2'")
```

Alignment

Description

The alignment of the object's text within its borders.

Applies to

Column, Computed Field, and Text objects

Syntax

Dot notation:

dw_control.Object.*objectname*.Alignment

Describe and Modify argument:

"*objectname*.Alignment { = ' *alignmentvalue* ' }"

SyntaxFromSQL:

Text (... Alignment = *value* ...)

Parameter	Description
objectname	The name of the object for which you want to get or set the alignment
alignmentvalue	(*exp*) A number specifying the type of alignment for the text of *objectname*. *Alignmentvalue* can be a quoted DataWindow painter expression. Values are: 0 — (Default) Left 1 — Right 2 — Center 3 — Justified When generating DataWindow syntax with SyntaxFromSQL, the setting for Alignment applies to all text objects used as column labels

Usage

When you select justified, the last line of text is not stretched to fill the line. Objects with only one line of text look left-aligned.

In the painter Set the value using:

- Object property sheet, General tab
- StyleBar (applies to all selected objects)

Examples

```
dw_1.Object.emp_name_t.Alignment = 2

ls_data = dw_1.Describe("emp_name.Alignment")

dw_1.Modify("emp_name_t.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 function.

Applies to

Database table for the DataWindow object

Not settable in PowerScript. Used in DataWindow syntax.

Syntax

Table(Arguments = ((*name1*, *type*), (*name2*, *type*) ...) ...)

Parameter	Description
name	The name of the retrieval argument
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 painter.

Open the SQL painter by selecting Design>Data Source from the menu bar. Then select Design>Retrieval Arguments.

Attributes

Description

A tab-separated list of all the properties that apply to an object.

Applies to

DataWindow, Bitmap, Button, Column, Computed Field, Graph, GroupBox, Line, OLE, Oval, Rectangle, Report, RoundRectangle, TableBlob, and Text objects

Syntax

Dot notation:

dw_control.Object.*objectname*.Attributes

Describe argument:

"*objectname*.Attributes"

Examples

```
ls_data = dw_1.Object.emp_name_t.Attributes
ls_data = dw_1.Describe("DataWindow.Attributes")
ls_data = dw_1.Describe("emp_name_t.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 objects

Syntax

Dot notation:

dw_control.Object.*graphname.axis*

Describe and Modify argument:

"*graphname.axis* { = ' *list* ' }"

Parameter	Description
graphname	The name of the graph within the DataWindow object for which you want to get or set the list of items for *axis*
axis	An axis name. Values are: ♦ Category ♦ Series ♦ Values
list	A string listing the categories, series, or values for the graph. The contents of the list depends on the axis you specify. The items in the list are separated by commas. List is quoted

Usage

In the painter Set the value by selecting a column or expression for each axis on the Graph Object property sheet, Data tab.

Examples

```
ls_data = dw_1.Object.gr_1.Values

dw_1.Object.gr_1.Series = "Actual, Budget"

ls_data = dw_1.Describe("gr_1.Category")

ls_data = dw_1.Describe("gr_1.Series")

ls_data = dw_1.Describe("gr_1.Values")

dw_1.Modify("gr_1.Series='Actual, Budget'")
```

Axis.property

Description

Settings that control the appearance of an axis on a graph.

Applies to

Graph objects

Syntax

Dot notation:

dw_control.Object.*graphname.axis.property*

Describe and Modify argument:

"*graphname.axis.property* { = *value* }"

Parameter	Description
graphname	The name of the graph within the DataWindow object for which you want to get or set a property value for an axis
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 painter expression

Property for Axis	Value
AutoScale	(*exp*) A boolean number specifying whether PowerBuilder scales the axis automatically. Values are: 0 — No, do not automatically scale the axis 1 — Yes, automatically scale the axis Painter: Set the value on an axis tab, Scale group, AutoScale option. Enabled when the axis displays nonstring data
DispAttr. *fontproperty*	(*exp*) Properties that control the appearance of the text that labels the axis divisions FOR INFO For a list of font properties, see DispAttr.*fontproperty*. Painter: Text tab. Choose Category Axis Text, Series Axis Text, or Values Axis Text and set font properties

Property for Axis	Value
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 Painter: An axis tab, Major Divisions group, Label Every option (not available for all graph types)
DropLines	(*exp*) 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 Painter: An axis tab, Major Divisions group, DropLines option (not available for all graph types)
Frame	(*exp*) An integer indicating the type of line used for the frame. Values are 0–5. See DropLines in this table for their meaning Painter: An axis tab, Line Style group, Frame option (available for 3D graph types)
Label	(*exp*) A string whose value is the axis label. Painter: An axis tab, Label text box
LabelDispAttr.*fontproperty*	(*exp*) Properties that control the appearance of the axis label FOR INFO For a list of font properties, see DispAttr.*fontproperty* Painter: Text tab. Choose Category Axis Label, Series Axis Label, or Values Axis Label and set font properties
MajorDivisions	(*exp*) An integer specifying the number of major divisions on the axis Painter: An axis tab, Major Divisions group, Number option
MajorGridLine	(*exp*) An integer specifying the type of line for the major grid. Values are 0–5. See DropLines in this table for their meaning Painter: An axis tab, Major Divisions group, Grid option

Property for Axis	Value
MajorTic	(*exp*) An integer specifying the type of the major tick marks. Values are: 1 — None 2 — Inside 3 — Outside 4 — Straddle Painter: An axis tab, Major Divisions group, Ticks option
MaximumValue	(*exp*) A double specifying the maximum value for the axis Painter: An axis tab, Scale group, Maximum option
MinimumValue	(*exp*) A double specifying the minimum value for the axis Painter: An axis tab, Scale group, Minimum option
MinorDivisions	(*exp*) An integer specifying the number of minor divisions on the axis Painter: An axis tab, Minor Divisions group, Number option
MinorGridLine	(*exp*) An integer specifying the type of line for the minor grid. Values are 0–5. See DropLines in this table for their meaning Painter: An axis tab, Minor Divisions group, Grid option
MinorTic	(*exp*) An integer specifying the type of the minor tick marks Values are: 1 — None 2 — Inside 3 — Outside 4 — Straddle Painter: An axis tab, Minor Divisions group, Ticks option
OriginLine	(*exp*) An integer specifying the type of origin line for the axis. Values are 0–5. See DropLines in this table for their meaning Painter: An axis tab, Line Style group. Enabled for numeric data axes.
PrimaryLine	(*exp*) An integer specifying the type of primary line for the axis. Values are 0–5. See DropLines in this table for their meaning Painter: An axis tab, Line Style group

Property for Axis	Value
RoundTo	(*exp*) A double specifying the value to which you want to round the axis values. Specify both a value and a unit (described next) Painter: An axis tab, Scale group, Round textbox
RoundToUnit	(*exp*) An integer specifying the units for the rounding value. The units must be appropriate for the axis data type. Values are: 0 — Default, for an axis of any data type 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 Painter: An axis tab, Scale group, Round To Maximum dropdown listbox
ScaleType	(*exp*) An integer specifying the type of scale used for the axis. Values are: 1 — Scale_Linear 2 — Scale_Log10 3 — Scale_Loge Painter: An axis tab, Scale group, Scale option
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 Painter: An axis tab, Line Style group, Secondary option
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 Painter: An axis tab. Enabled for 3D graphs only

Property for Axis	Value
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 Painter: Graph tab, Series Sort and Category Sort options

Usage

In the painter Set the value using the Graph Object property sheet, various tabs.

Examples

```
ls_data = dw_1.Object.gr_1.Category.AutoScale

dw_1.Object.Category.LabelDispAttr.Alignment = 2

ls_data = dw_1.Describe("gr_1.Category.AutoScale")

dw_1.Modify("gr_1.Series.AutoScale=0")

dw_1.Modify("gr_1.Values.Label='Cities'")

dw_1.Modify("gr_1.Category.LabelDispAttr.Alignment=2")
```

BackColor

Description

The background color of a graph in a DataWindow.

Applies to

Graph objects

Syntax

Dot notation:

dw_control.Object.*graphname*.BackColor

Describe and Modify argument:

"*graphname*.BackColor { = *long* }"

Parameter	Description
graphname	The graph whose background color you want to get or set
long	(*exp*) A long expression specifying the color (red, green, and blue values) to be used as the graph's background color. *Long* can be a quoted DataWindow painter expression

Usage

In the painter Set the value using the Graph Object property sheet, General tab.

Examples

```
dw_1.Object.graph_1.BackColor = 250
setting = dw_1.Describe("graph_1.BackColor")
dw_1.Modify("graph_1.BackColor=250")
```

Background.*property*

Description

Settings for the color and transparency of an object.

Applies to

Button, Column, Computed Field, GroupBox, Line, Oval, Rectangle, RoundRectangle, and Text objects

Syntax

Dot notation:

dw_control.Object.*objectname*.Background.*property*

Describe and Modify argument:

"*objectname*.Background.*property* { = ' *value* ' }"

SyntaxFromSQL:

Column (Background.*property* = *value*)

Text (Background.*property* = *value*)

Parameter	Description
objectname	The object whose Background properties you want to get or set When generating DataWindow syntax with SyntaxFromSQL, the Background settings apply to all columns or all text objects
property	A property that applies to the background of an object, as listed in the Property table below
value	Values for the properties are shown below. *Value* can be a quoted DataWindow painter expression

Property for Background	Value
Color	(*exp*) A long expression specifying the color (the red, green, and blue values) to be used as the object's background color
Mode	(*exp*) A number expression specifying the mode of the background of objectname. Values are: 0 — Make the object's background opaque 1 — Make the object's background transparent

Usage

In the painter Set the value using:

- Object property sheet, General tab, Fill option (choose Transparent or a color)

  When you create an Oval, Rectangle, or RoundRectangle object, the background and foreground colors of the fill pattern are the same, taken from the Foreground Color on the Stylebar. You can't change the background color after the object has been created. The Fill option changes the property Brush.Color.

- Object property sheet, Expressions tab (use a conditional expression)

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

```
dw_1.Object.oval_1.Background.Color = RGB(255, 0, 128)

ls_data = dw_1.Describe("oval_1.Background.Color")

dw_1.Modify("emp_name.Background.Color='11665407'")

ls_data = dw_1.Describe("emp_name.Background.Mode")

dw_1.Modify("emp_name.Background.Mode='1'")

dw_1.Modify("rndrect_1.Background.Mode='0'")
```

```
SQLCA.SyntaxFromSQL(sql_syntax, &
    "Style(...) Column(Background.Mode=1 ...) ...", &
    ls_Errors)

SQLCA.SyntaxFromSQL(sql_syntax, &
    "Style(...) Column(Background.Color=11665407 ...)", &
    ls_Errors)
```

Band

Description

The band or layer in the DataWindow object that contains the object. The returned text is one of the following, where # is the level number of a group: detail, footer, header, header.#, summary, trailer.#, foreground, background.

Changing an object's band
Use the SetPosition function to change an object's band during execution.

Applies to

Bitmap, Button, Column, Computed Field, Graph, GroupBox, Line, OLE, Oval, Rectangle, Report, RoundRectangle, TableBlob, and Text objects

Syntax

Dot notation:

dw_control.Object.*objectname*.Band

Describe and Modify argument:

"*objectname*.Band"

Parameter	Description
objectname	The name of the object within the DataWindow for which you want the band it occupies

Usage

In the painter Set the value using:

- When the object's layer is Band, drag the object into another band
- Object property sheet, Position tab, Layer option

Examples

```
ls_data = dw_1.Object.emp_title.Band

ls_data = dw_1.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

Dot notation:

dw_control.Object.DataWindow.*bandname*{.#}.*property*

Describe and Modify argument:

"DataWindow.*bandname*{.#}.*property* { = *value* }"

Parameter	Description
bandname	The identifier of a band in the DataWindow object. Values are: ♦ Detail ♦ Header ♦ Footer ♦ Summary ♦ Trailer
#	The number of the group you want when *bandname* is Header or Trailer. 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

Property for Bandname	Value
Color	(*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 painter expression Painter: General tab, Color option, or Expressions tab
Height	An integer specifying the height of the detail area in the unit of measure specified for the DataWindow Painter: General tab, Height option For another way of setting the height of the detail band, see the SetDetailHeight function

Property for Bandname	Value
Height.AutoSize	(Only when *bandname* is Detail) Whether the height of the band in the DataWindow object should be calculated by looking at all the objects that are autosized. Values are: No — Make all the heights the same Yes — Calculate the height according to the formula: Height = height of the largest painter row + painted row height – row height The height of the detail band will never be less than the height selected in the painter Painter: General tab, Autosize Height option
Pointer	(*exp*) A string specifying a value of the Pointer enumerated data type or the name of a cursor file (.CUR) to be used for the pointer. See the SetPointer function for a list of Pointer values. *Pointername* can be a quoted DataWindow painter expression. Painter: Pointer tab or Expressions tab

Usage

In the painter Set the value using the Band Object property sheet. To display the property sheet, right-click the gray divider for the band and choose Properties.

Examples

```
ls_data = dw_1.Object.DataWindow.Detail.Height

dw_1.Object.DataWindow.Detail.Pointer = "hand.cur"

dw_1.Object.DataWindow.Trailer.2.Height = 500

ls_data = dw_1.Describe("DataWindow.Detail.Height")

ls_data = &
    dw_1.Describe("DataWindow.Detail.Height.AutoSize")

dw_1.Modify("DataWindow.Detail.Pointer='hand.cur'")

dw_1.Modify("DataWindow.Detail.Pointer=' ~"Cross!~" ~t &
if(emp_status=~"a~", ~"HourGlass!~", ~"Cross!~")'")

dw_1.Modify("DataWindow.Footer.Height=250")

ll_color = RGB(200, 200, 500)
dw_1.Modify("DataWindow.Header.2.Color=" &
    + String(ll_color))

dw_1.Modify("DataWindow.Trailer.2.Height=500")
```

```
dw_1.Modify( &
"DataWindow.Summary.Pointer='c:\pb\total.cur'")
```

Bandname.Text

Description

(RichText presentation style only) The rich text content of the specified band as an ASCII string.

When you use Describe or dot notation, nested quotes are converted to tilde-quote combinations. To get pure RTF data, use the CopyRTF function.

Applies to

DataWindows in the RichText presentation style

Syntax

Dot notation:

dw_control.Object.DataWindow.*bandname*.Text

Describe and Modify argument:

"DataWindow.*bandname*.Text { = *rtfstring* }"

Parameter	Description
bandname	The identifier of a band in the DataWindow object that has the RichText presentation style. Values are: ♦ Detail ♦ Header ♦ Footer
rtfstring	A string whose value is the rich text content of the band. The string includes the rich text formatting codes, text, and input fields Text assigned to the header or footer band is ignored if RichText.HeaderFooter is set to no When you assign text using the Modify function or dot notation, nested quotes must be represented with tildes and quotes. If your data is a pure RTF string, use the PasteRTF function

Usage

In the painter Set the value by editing the content of each band in the painter workspace.

Examples

```
ls_footertext = dw_1.Object.DataWindow.Footer.Text
ls_data = dw_1.Describe("DataWindow.Detail.Text")
```

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.#, Summary, Trailer.#. The items in the list are separated by tabs.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.Bands

Describe argument:

"DataWindow.Bands"

Examples

```
ls_data = dw_1.Object.DataWindow.Bands
ls_data = dw_1.Describe("DataWindow.Bands")
```

BinaryIndex

Description

An internal index that PowerBuilder uses to manage the OLE object in the library. But there is no reason to get this value; the value has no external significance.

Applies to

OLE objects

Syntax

"*oleobjectname*.BinaryIndex"

BitmapName

Description

Whether PowerBuilder interprets the column's value as the name of a bitmap file and displays the bitmap instead of the text. BitmapName's value is either Yes or No.

Applies to

Column objects

Syntax

Dot notation:

dw_control.Object.*columnname*.BitmapName

Describe argument:

"*columnname*.BitmapName"

Usage

In the painter Set the value using the Object property sheet, General tab, Display As Picture.

Examples

```
ls_data = dw_1.Object.emp_name.BitmapName
ls_data = dw_1.Describe("emp_name.BitmapName")
```

Border

Description

The type of border for the object.

Applies to

Bitmap, Column, Computed Field, Graph, GroupBox, OLE, Report, TableBlob, and Text objects

Syntax

Dot notation:

dw_control.Object.*objectname*.Border

Describe and Modify argument:

"*objectname*.Border { = ' *value* ' }"

SyntaxFromSQL:

Column (... Border = *value* ...)

Text (... Border = *value* ...)

Parameter	Description
objectname	The name of the object whose border you want to get or set When generating DataWindow syntax with SyntaxFromSQL, the Border setting applies to all columns or all text objects

Parameter	Description
value	(*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 Integer can be a DataWindow quoted painter expression When you change between Resize and another border, change the Resizeable property too so that the object's appearance and behavior match For columns, you can access the Border property with the GetBorderStyle and SetBorderStyle functions

Usage

In the painter Set the value using:

- Object property sheet, General tab

  Changing the Border setting between Resize and another border affects the Resizeable option on the Position tab. To make another border resizable, choose the border. Close and then redisplay the property sheet and check Resizeable on the Position tab.

- Object property sheet, Expressions tab (use a conditional expression)

Examples

```
ls_data = dw_1.Object.emp_name_t.Border

dw_1.Object.emp_name_t.Border='6'

ls_data = dw_1.Describe("emp_name_t.Border")

dw_1.Modify("emp_name_t.Border='6'")

SQLCA.SyntaxFromSQL(sql_syntax, &
    "Style(...) Column(Border=5 ...) ...", ls_Errors)
```

Brush.*property*

Description

Settings for the fill pattern and color of a graphic object.

Applies to

Oval, Rectangle, and RoundRectangle objects

Syntax

Dot notation:

dw_control.Object.*objectname*.Brush.*property*

Describe and Modify argument:

"*objectname*.Brush.*property* { = ' *value* ' }"

Parameter	Description
objectname	The name of the Line, Oval, Rectangle, RoundRectangle, or Text object whose Brush property you want to get or set
property	A property that applies to the Brush characteristics of an object, as listed in the table below
value	Values for the properties are shown below. Value can be a quoted DataWindow painter expression

Property for Brush	Value
Color	(*exp*) A long expression specifying the color (the red, green, and blue values) to be used to fill the object Painter: General tab, Fill option
Hatch	(*exp*) A number expression specifying the fill pattern of objectname. 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 Painter: General tab, Pattern option

Usage

In the painter Set the value using:

- Object property sheet, General tab
- Object property sheet, Expressions tab (use a conditional expression)

Examples

```
ls_data = dw_1.Object.oval_1.Brush.Hatch

dw_1.Object.oval_1.Brush.Hatch = 5

ls_data = dw_1.Describe("oval_1.Brush.Hatch")

dw_1.Modify("oval_1.Brush.Hatch='5'")
```

```
dw_1.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 objects

Syntax

Dot notation:

dw_control.Object.*columnname*.CheckBox.*property*

Describe and Modify argument:

"*columnname*.CheckBox.*property* { = *value* }"

Parameter	Description
columnname	The column whose edit style is CheckBox for which you want to get or set property values
property	A property for the CheckBox edit style, as listed in the table below
value	Values for the properties are shown in the table below. For CheckBox properties, *value* cannot be a DataWindow painter expression

Property for CheckBox	Value
3D	Whether the CheckBox should be 3D. Values are: Yes — Make the CheckBox 3D No — Do not make the CheckBox 3D Painter: 3D Look option

Property for CheckBox	Value
LeftText	Whether the CheckBox label is to the left or right of the CheckBox. Values are: Yes — Display the label on the left No — Display the label on the right Painter: Left Text option
Off	A string constant specifying the column value when the CheckBox is off (unchecked). The resulting value must be the same data type as the column Painter: Off option
On	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 data type as the column Painter: On option
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 data type as the column Painter: Other option (available when 3 States is checked)
Scale	Whether you want to scale the 2D CheckBox. Takes effect only when the 3D property is No. Values are: Yes — Scale the CheckBox No — Do not scale the CheckBox Painter: Scale option
Text	A string specifying the CheckBox's label text Painter: Text option

Usage

In the painter Set values using the Object property sheet, Edit tab, when Style option is CheckBox.

Examples

```
dw_1.Object.emp_gender.CheckBox.3D = "no"

IF dw_1.Object.emp_status.CheckBox.LeftText = "yes" THEN
dw_1.Object.emp_status2.CheckBox.LeftText = "yes"
END IF

dw_1.Modify("emp_gender.CheckBox.3D=no")

IF dw_1.Describe("emp_status.CheckBox.LeftText") &
    = "yes" THEN
```

```
dw_1.Modify("emp_status2.CheckBox.LeftText=yes")
END IF

dw_1.Modify("emp_status.CheckBox.Off='Terminated'")

dw_1.Modify("emp_status.CheckBox.On='Active'")

dw_1.Modify("emp_status.CheckBox.Other='Unknown'")
```

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 and TableBlob objects

Syntax

Dot notation:

dw_control.Object.*objectname*.ClientName

Describe and Modify argument:

"*objectname*.ClientName { = ' *clientname* ' }"

Parameter	Description
objectname	The name of a blob column or an OLE container object
clientname	(*exp*) A string expression to be used in the title of the server application's window. For a blob, the string usually includes data from the current row so that the window title can identify the blob's row Begin the string with a tab (~t) when you modify the value so that PowerBuilder evaluates the expression instead of displaying it

Usage

In the painter Set the value using the Object property sheet, Options tab.

Examples

```
cname = dw_1.Object.emppict_blob.ClientName

dw_1.Object.emppict_blob.ClientName = &
    "~t'Data for ' String(emp_id)"

cname = dw_1.Describe("emppict_blob.ClientName")

dw_1.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 object:

- 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, tickmarks, and the outline of the box for 3D graphs

Applies to

DataWindow, Button, Column, Graph, and GroupBox objects

Syntax

Dot notation:

dw_control.Object.DataWindow.Color

dw_control.Object.*objectname*.Color

Describe and Modify argument:

"DataWindow.Color { = *long* }"

"*objectname*.Color { = *long* }"

SyntaxFromSQL:

DataWindow (Color = *long*)

Column (Color = *long*)

Parameter	Description
objectname	The column whose text color you want to set or the graph whose line color you want to set
long	(*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 painter expression in quotes. You cannot specify an expression for the DataWindow background color When generating DataWindow syntax with SyntaxFromSQL, the Color setting for Column applies to all columns

Usage

In the painter For the DataWindow background, set the value using:

- DataWindow Object property sheet, General tab, Color option

For a column's text color, set the value using:

- Object property sheet, Font tab, Text Color option
- Object property sheet, Expressions tab

For a graph's line color, set the value using:

- Object property sheet, General tab, Line Color option

Examples

```
column_text_color = dw_1.Object.emp_name.Color

dw_1.Object.salary.Color = &
    "0~tIf(salary>90000,255,65280)"

dw_back_color = dw_1.Describe("DataWindow.Color")

column_text_color = dw_1.Describe("emp_name.Color")

dw_1.Modify( &
    "salary.Color='0~tIf(salary>90000,255,65280)'")
```

See also

Background, BackColor

ColType

Description

The data type of the column or computed field.

Applies to

Column and Computed Field objects

Syntax

Dot notation:

dw_control.Object.*objectname*.ColType

Describe argument:

"*objectname*.ColType"

Parameter	Description
objectname	The column for which you want the data type. Possible data types are: ♦ Char (*n*) — *n* is the number of characters ♦ Date ♦ DateTime ♦ Decimal (*n*) — *n* is the number of decimal places ♦ Int ♦ Long ♦ Number ♦ Time ♦ Timestamp

Usage

In the painter The value of ColType is derived from the data or expression you specify for the object.

Examples

```
ls_coltype = dw_1.Object.emp_id.ColType
ls_coltype = dw_1.Describe("emp_id.ColType")
```

Column.Count

Description

The number of columns in the DataWindow object.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.Column.Count

Describe argument:

"DataWindow.Column.Count"

Usage

In the painter The value is determined by the number of columns you select in the Select painter, whether or not they are displayed.

Examples

```
ls_colcount = dw_1.Object.DataWindow.Column.Count
ls_colcount = dw_1.Describe("DataWindow.Column.Count")
```

ContentsAllowed

Description

The way the OLE container 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 objects

Syntax

Dot notation:

dw_control.Object.*oleobjectname*.ContentsAllowed

Describe and Modify argument:

"*oleobjectname*.ContentsAllowed { = ' *contentstype* ' }"

Parameter	Description
oleobjectname	The name of the OLE container object for which you want to get or set the type of contents
contentstype	A number specifying whether the OLE object in the container has to be embedded, has to be linked, or can be either embedded or linked. Values are: 0 — Embedded 1 — Linked 2 — Any

Usage

In the painter Set the value using the Object property sheet, Options tab, Contents option.

Examples

```
ls_data = dw_1.Object.ole_report.ContentsAllowed

dw_1.Object.ole_report.ContentsAllowed = 2

ls_data = dw_1.Describe("ole_report.ContentsAllowed")

dw_1.Modify("ole_report.ContentsAllowed='2'")
```

Criteria

Description

The WHERE clause for a related report. The Criteria property defines the connection between the related report and the DataWindow.

Applies to

Report objects

Syntax

Dot notation:

dw_control.Object.*reportname*.Criteria

Describe and Modify argument:

"*reportname*.Criteria { = *string* }"

Parameter	Description
reportname	The name of the report object for which you want to get or set Criteria
string	An expression that will be the WHERE clause for the related report

Examples

```
ls_colcount = dw_1.Object.rpt_1.Criteria
dw_1.Object.rpt_1.Criteria = "emp_id=:emp_id"
ls_colcount = dw_1.Describe("rpt_1.Criteria")
dw_1.Modify("rpt_1.Criteria='emp_id=:emp_id'")
```

See also

Nest_Arguments DataWindow object property

Criteria.*property*

Description

Settings for the Prompt for Criteria dialog box. When Prompt for Criteria is enabled, PowerBuilder prompts the user to specify criteria for retrieving data whenever the Retrieve function is called. Note that the Required property also affects query mode.

Applies to

Column objects

Syntax

Dot notation:

dw_control.Object.*columnname*.Criteria.*property*

Describe and Modify argument:

"*columnname*.Criteria.*property* { = *value* }"

Parameter	Description
columnname	The name of the column for which you want to get or set Prompt for Criteria properties
property	A property for the Prompt for Criteria dialog. Properties and their settings are listed in the table below
value	A Yes or No value to be assigned to the property. For Criteria properties, *value* cannot be a DataWindow painter expression

Property for Criteria	Value
Dialog	Whether Prompt for Criteria is on for *columnname*. Values are: Yes — Include *columnname* in the Prompt for Criteria dialog box No — (Default) Do not include *columnname* in the Prompt for Criteria dialog box If the Dialog property is Yes for at least one column in the DataWindow, then PowerBuilder displays the Prompt for Criteria dialog box when the Retrieve function is called Painter: Rows menu, Prompt for Criteria dialog box, select each column
Override_Edit	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. Values are: Yes — Allow the user to override the column's edit style and enter data in a standard edit control No — (Default) Constrain the user to the edit style for the column Painter: Object property sheet, General tab, Query Criteria group, Override Edit option
Required	Whether the user is restricted to the equality operator (=) when specifying criteria in query mode and in the Prompt for Criteria dialog box. Values are: Yes — Require the user to use the equality operator only No — (Default) Allow the user to use any relational operator, including =, <>, <, >, >=, and <= Painter: Object property sheet, General tab, Query Criteria group, Equality Required option

Usage

In the painter Set the values using the menus and Object property sheet as described in the table above.

Examples

```
setting = dw_1.Object.empname.Criteria.Dialog

dw_1.Object.empname.Criteria.Dialog= "Yes"

setting = dw_1.Describe("empname.Criteria.Dialog")

dw_1.Modify("empname.Criteria.Dialog=Yes")

dw_1.Modify("empname.Criteria.Override_Edit=Yes")
```

```
dw_1.Modify("empname.Criteria.Required=No")

IF dw_1.Describe("empname.Criteria.Edit.Style") &
    = "dddw" THEN
dw_1.Modify("empname.Criteria.Override_Edit=Yes")
END IF
```

Crosstab.*property*

Description

Settings for a DataWindow object whose presentation style is Crosstab.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.Crosstab.*property*

Describe and Modify argument:

"DataWindow.Crosstab.*property* { = *value* }"

Parameter	Description
property	A property for a Crosstab DataWindow. Properties and their settings are listed in the table below
value	A string expression listing the items to be assigned to the property. For Crosstab properties, *value* is always quoted and can be a DataWindow painter expression

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 Painter: Columns option
Rows	(*exp*) A string containing a comma- or tab-separated list of the names of columns that make up the rows of the crosstab Painter: rows option

Property for Crosstab	Value
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 Painter: Source Data option
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 are: Yes — StaticMode is enabled No — (Default) StaticMode is disabled Painter: Not set in painter
Values	(*exp*) A string containing a comma- or tab-separated list of expressions that will be used to calculate the values of the crosstab Painter: Values option

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 workspace to display the popup menu and select Crosstab.

Examples

```
setting = dw_1.Object.DataWindow.Crosstab.Columns

dw_1.Object.DataWindow.Crosstab.Columns = "dept_id"

setting = dw_1.Describe("DataWindow.Crosstab.Columns")

dw_1.Modify("DataWindow.Crosstab.Columns='dept_id'")

dw_1.Modify("DataWindow.Crosstab.Rows='salary'")

dw_1.Modify("DataWindow.Crosstab.SourceNames=" &
+ "'Order Number, Item Number, Price'")

dw_1.Modify("DataWindow.Crosstab.Values='empname'")

dw_1.Modify("DataWindow.Crosstab.StaticMode='yes'")
```

See also

CrosstabDialog function in the *PowerScript Reference*

Data

Description

A tab-separated list describing the data in the DataWindow object.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.Data

Describe argument:

"DataWindow.Data"

Examples

```
setting = dw_1.Object.DataWindow.Data
setting = dw_1.Describe("DataWindow.Data")
```

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

Dot notation:

dw_control.Object.DataWindow.Data.HtmlTable

Describe argument:

"DataWindow.Data.HtmlTable"

Usage

Some presentation styles translate into better HTML than others. The Tabular, Group, Freeform, Crosstab, and Grid presentation styles produce good results. The Composite, RichText, Graph, and OLE 2.0 presentation styles and nested reports produce HTML tables based on the result set only and not on the presentation style. DataWindows with overlapping objects in them may not produce the desired results.

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 function.

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 Get the value using Design>Preview, Save Rows As with Save As Type set to HTMLTable.

Examples

```
ls_html = dw_1.Object.DataWindow.Data.HTMLTable
ls_html = dw_1.Describe("DataWindow.Data.HTMLTable")
```

DataObject

Description

The name of the DataWindow that is the nested report within the main DataWindow.

Applies to

Report objects

Syntax

Dot notation:

dw_control.Object.*reportname*.DataObject

Describe and Modify argument:

"*reportname*.DataObject = ' *dwname* ' "

Parameter	Description
reportname	The name of the Report object in the main DataWindow for which you want to get or set the nested DataWindow
dwname	A string naming a DataWindow object in the application's libraries that is the DataWindow for the report within the main DataWindow

Usage

In the painter Set the value using the Object property sheet, Select Report tab.

Examples

```
setting = dw_1.Object.rpt_1.DataObject
dw_1.Object.rpt_1.DataObject = "d_empdata"
setting = dw_1.Describe("rpt_1.DataObject")
dw_1.Modify("rpt_1.DataObject='d_empdata'")
```

dbName

Description

The name of the database column. PowerBuilder uses this value to construct the update syntax.

Applies to

Column objects

Syntax

Dot notation:

dw_control.Object.*columnname*.dbName

Describe and Modify argument:

"*columnname*.dbName { = ' *dbcolumnname* ' }"

Parameter	Description
columnname	The name of the column for which you want the name of the corresponding database column
dbcolumnname	The name of the database column associated with *columnname*

Usage

DbName is the name of the database column in the format *tablename.columnname*. The value of dbName does not include the quotes that may be part of the SQL syntax.

In the painter On the Syntax tab of the Select painter's SQL Toolbox, you can see the database column names (they may be shown with quotes).

Examples

```
dbcol = dw_1.Object.emp_id.dbName

dw_1.Object.emp_id.dbName = "emp_id"

dbcol = dw_1.Describe("emp_id.dbName")

dw_1.Modify("emp_id.dbName='emp_id'")
```

dddw.*property*

Description

Properties that control the appearance and behavior of a column with the DropDownDataWindow edit style.

Applies to

Column objects

Syntax

Dot notation:

dw_control.Object.*columnname*.dddw.*property*

Describe and Modify argument:

"*columnname*.dddw.*property* { = *value* }"

Parameter	Description
columnname	The name of a column that has the DropDownDataWindow edit style
property	A property for the DropDownDataWindow column. Properties and their settings are listed in the table below
value	The value to be assigned to the property. For dddw properties, *value* cannot be a DataWindow painter expression

Property for dddw	Value
AllowEdit	Whether the user can type a value as well as choose from the DropDownDataWindow's list. Values are: Yes — Typing is allowed No — (Default) Typing is not allowed Painter: Allow Editing option
AutoHScroll	Whether the DropDownDataWindow automatically scrolls horizontally when the user enters or deletes data. Values are: Yes — (Default) Scroll horizontally automatically No — Do not scroll automatically Painter: Auto Horz Scroll option
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 Painter: Case option
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 Painter: Data Column option

Property for dddw	Value
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 Painter: Display Column option
HScrollBar	Whether a horizontal scrollbar displays in the DropDownDataWindow. Values are: Yes — Display a horizontal scrollbar No — Do not display a horizontal scrollbar Painter: H Scroll Bar option
HSplitScroll	Whether the horizontal scrollbar is split. The user can adjust the split position. Values are: Yes — Split the horizontal scrollbar so the user can scroll the display and data columns separately No — The horizontal scrollbar is not split Painter: Split Horz Scroll Bar option
Limit	An integer from 0 to 32767 specifying the maximum number of characters that can be entered in the DropDownDataWindow. Zero means unlimited Painter: Limit option
Line	An integer from 0 to 32767 specifying the number of lines (values) to display in the DropDownDataWindow Painter: Lines in DropDown option
Name	A string whose value is the name of the predefined DropDownDataWindow 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 Call GetChild *after* setting dddw.Name to get a valid reference to the column's DropDownDataWindow Painter: Name option
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 Painter: Empty String is NULL option

Property for dddw	Value
PercentWidth	An integer specifying the width of the dropdown portion of the DropDownDataWindow as a percentage of the column's width Call GetChild *after* setting dddw.PercentWidth to get a valid reference to the column's DropDownDataWindow Painter: Width of DropDown option
Required	Whether the column is required. Values are: Yes — Required No — (Default) Not required Painter: Required option
ShowList	Whether the ListBox portion of the DropDownDataWindow displays when the column has focus. A down arrow does not display at the right end of the DropDownDataWindow when dddw.ShowList is yes. Values are: Yes —Display the list whenever the column has the focus No — Do not display the list until the user selects the column Painter: Always Show List option
UseAsBorder	Whether a down arrow displays at the right end of the DropDownDataWindow. Values are: Yes — Display the arrow No — Do not display the arrow Note that if ShowList is set to Yes, the column ignores the UseAsBorder property and the arrow never displays Painter: Always Show Arrow option
VScrollBar	Whether a vertical scrollbar displays in the DropDownDataWindow for long lists. Values are: Yes — Display a vertical scrollbar No — Do not display a vertical scrollbar Painter: V Scroll Bar option

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 again after changing these properties to obtain a valid reference.

In the painter Set values using the Object property sheet, Edit tab, when Style is DropDownDW.

Examples

```
ls_data = dw_1.Object.emp_status.dddw.AllowEdit")
dw_1.Object.emp_status.dddw.Case = "Any"
ls_data = dw_1.Describe("emp_status.dddw.AllowEdit")
dw_1.Modify("emp_status.dddw.Case='Any'")
dw_1.Modify("emp_status.dddw.DataColumn='status_id'")
dw_1.Modify("emp_status.dddw.Limit=30")
dw_1.Modify("emp_status.dddw.Name='d_status'")
dw_1.Modify("emp_status.dddw.PercentWidth=120")
```

ddlb.*property*

Description

Properties that control the appearance and behavior of a column with the DropDownListBox edit style.

Applies to

Column objects

Syntax

Dot notation:

dw_control.Object.*columnname*.ddlb.*property*

Describe and Modify argument:

"*columnname*.ddlb.*property* { = *value* }"

Parameter	Description
columnname	The name of a column that has the DropDownListBox edit style
property	A property for the DropDownListBox column. Properties and their settings are listed in the table below
value	The value to be assigned to the property. For ddlb properties, value cannot be a DataWindow painter expression

Property for ddlb	Value
AllowEdit	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 Painter: Allow Editing option
AutoHScroll	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 Painter: Auto Horz Scroll option
Case	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 Painter: Case option
Limit	An integer from 0–32767 specifying the maximum number of characters that can be entered in the DropDownListBox. Zero means unlimited Painter: Limit option
NilIsNull	Whether to set the data value of the DropDownListBox 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 Painter: Empty string is NULL option
Required	Whether the column is required. Values are: Yes — Required No — (Default) Not required Painter: Required option
ShowList	Whether the ListBox portion of the DropDownListBox displays when the column has focus. A down arrow does not display at the right end of the DropDownListBox when ddlb.ShowList is yes. Values are: Yes — Display the list whenever the column has focus No — Do not display the list until the user selects the column Painter: Always Show List option

Property for ddlb	Value
Sorted	Whether the list in the DropDownListBox is sorted. Values are: Yes — The list is sorted No — The list is not sorted Painter: Sorted option
UseAsBorder	Whether a down arrow displays at the right end of the DropDownListBox. Values are: Yes — Display the arrow No — Do not display the arrow Note that if ShowList is set to Yes, the column ignores the UseAsBorder property and the arrow never displays Painter: Always Show Arrow option
VScrollBar	Whether a vertical scrollbar displays in the DropDownListBox for long lists. Values are: Yes — Display a vertical scrollbar No — Do not display a vertical scrollbar Painter: Vert Scroll Bar option

Usage

In the painter Set values using the Object property sheet, Edit tab, when Style is DropDownListBox.

Examples

```
ls_data = dw_1.Object.emp_status.ddlb.AllowEdit

dw_1.Object.emp_status.ddlb.Case = "Any"

ls_data = dw_1.Describe("emp_status.ddlb.AllowEdit")

dw_1.Modify("emp_status.ddlb.Case='Any'")

dw_1.Modify("emp_status.ddlb.Limit=30")
```

DefaultPicture

Description

Whether the action's default picture is to be used on the button (a user defined action does not have a picture).

Applies to

Button objects

Syntax

Dot notation:

dw_control.Object.*buttonname*.DefaultPicture

Describe and Modify argument:

"*buttonname*.DefaultPicture { = ' *value* ' }"

Parameter	Description
buttonname	The name of the button to which you want to assign an action
value	Whether the action's default picture is used. Values are: Yes — Use the default picture No — Do not use the default picture

Usage

In the painter Set the value using the Button object property sheet, General tab.

Examples

```
dw_1.Object.b_name.DefaultPicture = "Yes"

setting = dw_1.Describe("b_name.DefaultPicture")

dw_1.Modify("b_name.DefaultPicture ='No'")
```

Depth

Description

The depth of a 3D graph.

Applies to

Graph objects

Syntax

Dot notation:

dw_control.Object.*graphname*.Depth

Describe and Modify argument:

"*graphname*.Depth { = ' *depthpercent* ' }"

Parameter	Description
graphname	The graph object within the DataWindow for which you want to set the depth
depthpercent	(*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 painter expression

Usage

In the painter Set the value using the Object property sheet, Graph tab, Depth (% width) option.

Examples

```
setting = dw_1.Object.graph_1.Depth

dw_1.Object.graph_1.Depth = 70

setting = dw_1.Describe("graph_1.Depth")

dw_1.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

SyntaxFromSQL:

Style (Detail_Bottom_Margin = *value*)

Parameter	Description
value	An integer specifying the size of the bottom margin of the detail area in the units specified for the DataWindow

Examples

```
SQLCA.SyntaxFromSQL(sqlstring, &
'Style(...Detail_Bottom_Margin = 25 ...)', &
errstring)
```

Detail_Top_Margin

Description

The size of the top margin of the DataWindow's detail area.

Applies to

Style keywords

Syntax

SyntaxFromSQL:

Style (Detail_Top_Margin = *value*)

Parameter	Description
value	An integer specifying the size of the top margin of the detail area in the units specified for the DataWindow

Examples

```
SQLCA.SyntaxFromSQL(sqlstring, &
'Style(...Detail_Top_Margin = 25 ...)', &
errstring)
```

Detail.*property*

See Bandname.property.

DispAttr.*fontproperty*

Description

Settings for the appearance of various text components of a graph.

Applies to

Properties of Graph objects, as noted throughout this discussion

Syntax

Dot notation:

dw_control.Object.*graphname.property*.DispAttr.*fontproperty*

Describe and Modify argument:

"*graphname.property*.DispAttr.*fontproperty* { = *value* }"

Parameter	Description
graphname	The Graph object in a DataWindow for which you want to get or set font appearance values
property	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: "*graphname.axis*.LabelDispAttr.*fontproperty* { = *value* }"

Parameter	Description
fontproperty	A property that controls the appearance of text in the graph. Properties and their settings are listed in the table below
value	The value to be assigned to *fontproperty*. *Value* can be a quoted DataWindow painter expression

Property for DispAttr	Value
Alignment	(*exp*) The alignment of the text. Values are: 0 — Left 1 — Right 2 — Center Painter: Alignment option
AutoSize	(*exp*) Whether the text element should be autosized according to the amount of text being displayed. Values are: 0 — Do not autosize 1 — Autosize Painter: AutoSize checkbox
BackColor	(*exp*) A long value specifying the background color of the text Painter: Background Color option
DisplayExpression	An expression whose value is the label for the graph component. The default expression is the property containing the text for the graph component. The expression can include the text property and add other variable text Painter: Display Expression option
Font.CharSet	(*exp*) 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 Painter: Set indirectly via font selection
Font.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 Painter: Rotation option

Property for DispAttr	Value
Font.Face	(*exp*) A string specifying the name of the font face, such as Arial or Courier Painter: Font option
Font.Family	(*exp*) 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 Painter: Set indirectly via font selection
Font.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 Painter: Height option, specified in points (not available when AutoSize is checked)
Font.Italic	(*exp*) Whether the text should be italic. Values are: 0 — Not italic (default) 1 — Italic Painter: Font Style option
Font.Orientation	Same as Escapement
Font.Pitch	(*exp*) The pitch of the font. Values are: 0 — The default pitch for your system 1 — Fixed 2 — Variable Painter: Set indirectly via font selection
Font.Strikethrough	(*exp*) Whether the text should be crossed out. Values are: 0 — Not crossed out (default) 1 — Crossed out Painter: Not settable for graph text in painter
Font.Underline	(*exp*) Whether the text should be underlined. Values are: 0 — Not underlined (default) 1 — Underlined Painter: Underline checkbox

Property for DispAttr	Value
Font.Weight	(*exp*) An integer specifying the weight of the text—for example, 400 for normal or 700 for bold Painter: Set indirectly via Font Style option
Font.Width	(*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
Format	(*exp*) A string containing the display format for the text Painter: Display Format option
TextColor	(*exp*) A long specifying the color to be used for the text Painter: Text Color option

Usage

In the painter Set values using the Object property sheet, Text tab. Settings apply to the selected item in the Text Object listbox.

Examples

```
setting = &
dw_1.Object.Category.LabelDispAttr.Font.Face

dw_1.Object.Category.LabelDispAttr.Font.Face = "Arial"

setting = &
dw_1.Describe("Category.LabelDispAttr.Font.Face")

dw_1.Modify("Category.LabelDispAttr.Font.Face='Arial'")

dw_1.Modify("Title.DispAttr.DisplayExpression=" &
+ "'Title + ~"~n~" + Today()'")
```

DisplayType

Description

The way the OLE container 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, PowerBuilder displays an icon even if DisplayType is set to contents.

Applies to

OLE objects

Syntax

Dot notation:

dw_control.Object.*oleobjectname*.DisplayType

Describe and Modify argument:

"*oleobjectname*.DisplayType { = ' *type* ' }"

Parameter	Description
oleobjectname	The name of the OLE container object for which you want to get or set the type of display
type	A number specifying whether the user will see an icon or an image of the OLE object's contents. *Type* can be a quoted DataWindow painter expression. Values are: 0 — Icon 1 — Content

Usage

In the painter Set the value on the Object property sheet, Options tab.

Examples

```
ls_data = dw_1.Object.ole_report.DisplayType

dw_1.Object.ole_report.DisplayType = 1

ls_data = dw_1.Describe("ole_report.DisplayType")

dw_1.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 objects

Syntax

Dot notation:

dw_control.Object.*columnname*.Edit.*property*

Describe and Modify argument:

"*columnname*.Edit.*property* { = *value* }"

SyntaxFromSQL:

Column (Edit.*property* = *value*)

Parameter	Description
columnname	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
property	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 SyntaxFromSQL
value	The value to be assigned to the property. For most Edit properties, you cannot specify a DataWindow painter expression. The exception is Edit.Format

Property for Edit	Value
AutoHScroll	Whether the edit control scrolls horizontally automatically when data is entered or deleted. Values are: Yes — Scroll horizontally automatically No — Do not scroll horizontally automatically You can use AutoHScroll with SyntaxFromSQL. The setting applies to all the columns in the generated syntax Painter: Auto Horz Scroll option
AutoSelect	Whether to select the contents of the edit control automatically when it receives focus. Values are: Yes — Select automatically No — Do not select automatically You can use AutoSelect with SyntaxFromSQL. The setting applies to all the columns in the generated syntax Painter: Auto Selection option
AutoVScroll	Whether the edit box scrolls vertically automatically when data is entered or deleted. Values are: Yes — Scroll vertically automatically No — Do not scroll vertically automatically You can use AutoVScroll with SyntaxFromSQL. The setting applies to all the columns in the generated syntax Painter: Auto Vert Scroll option

Property for Edit	Value
Case	The case of the text in the edit control. Values are: Any — Character of any case allowed Upper — Characters converted to uppercase Lower — Characters converted to lowercase Painter: Case option
CodeTable	Whether the column has a code table. Values are: Yes — Code table defined No — No code table defined Painter: Use Code Table option
DisplayOnly	Whether the column is display only. Values are: Yes — Do not allow the user to enter data; make the column display only No — (Default) Allow the user to enter data Painter: Display Only option For conditional control over column editing, uae the Protect property
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 SyntaxFromSQL. The setting applies to all the columns in the generated syntax Painter: Show Focus Rectangle option
Format	(*exp*) A string containing the display format of the edit control. The value for Format is quoted and can be a DataWindow painter expression Painter: Format option (don't use quotes around the value)
HScrollBar	Whether a horizontal scrollbar displays in the edit control. Values are: Yes — Display the horizontal scrollbar No — Do not display the horizontal scrollbar Painter: Horz Scroll Bar option
Limit	A number specifying the maximum number of characters (0 to 32,767) that the user can enter. 0 means unlimited Painter: Limit option

Property for Edit	Value
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 Painter: Name option
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 Painter: Empty String is NULL option
Password	Whether to assign secure display mode to the column. When the user enters characters, they display as asterisks (*) Values are: Yes — Assign securedisplay 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, *****) Painter: Password option
Required	Whether the column is required. Values are: Yes — It is required No — It is not required Painter: Required option
Style	(*Describe only*) Returns the edit style of the column
ValidateCode	Whether the code table will be used to validate user-entered values. Values are: Yes — Use the code table No — Do not use the code table Painter: Validate Using Code Table option
VScrollBar	Whether a vertical scrollbar displays in the line edit. Values are: Yes — Display vertical scrollbars No — Do not display vertical scrollbars Painter: Vert Scroll Bar option

Usage

In the painter Set values using the Object property sheet, Edit tab, when Style is Edit.

Examples

```
setting = dw_1.Object.emp_name.Edit.AutoHScroll

dw_1.Object.emp_name.Edit.Required = "no"

setting = dw_1.Describe("emp_name.Edit.AutoHScroll")

dw_1.Modify("emp_name.Edit.Required=no")
```

EditMask.*property*

Description

Settings that affect the appearance and behavior of columns with the EditMask edit style.

Applies to

Column objects

Syntax

Dot notation:

dw_control.Object.*columnname*.EditMask.*property*

Describe and Modify argument:

"*columnname*.EditMask.*property* { = *value* }"

Parameter	Description
columnname	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
property	A property for the column's EditMask style. Properties and their settings are listed in the table below
value	The value to be assigned to the property. For EditMask properties, you cannot specify a DataWindow painter expression

Property for EditMask	Value
AutoSkip	Whether the EditMask will automatically skip to the next field when the maximum number of characters have been entered: Yes — Skip automatically No — Do not skip automatically Painter: AutoSkip option
CodeTable	Whether the column has a code table. Values are: Yes — Code table defined No — No code table defined Painter: Code Table option (available when Spin Control is checked)
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 Painter: Focus Rectangle option
Mask	A string containing the edit mask for the column Painter: Mask option
ReadOnly	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 Painter: Read Only option (available when Spin Control is checked)
Required	Whether the column is required. Values are: Yes — It is required No — It is not required Painter: Required option
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 Painter: Spin Control option

Property for EditMask	Value
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 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 Painter: Spin Increment option (available for numeric, date, and time 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 Yes Because the SpinRange string is within another quoted string, the tilde separator becomes four tildes, which reduces to a single tilde when parsed. The format for the string is: "EditMask.SpinRange = ' *minval~~~~maxval* ' " Painter: Spin Range group, Min and Max options (available for numeric, date, and time columns)
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 Painter: Use Format option

Usage

In the painter Set values using the Object property sheet, Edit tab, when Style is EditMask.

Examples

```
setting = dw_1.Object.emp_status.EditMask.Spin

dw_1.Object.emp_bonus.EditMask.SpinIncr = 1000

dw_1.Object.id.EditMask.SpinRange = "0~~~~10"

setting = dw_1.Describe("emp_status.EditMask.Spin")

dw_1.Modify("emp_bonus.EditMask.SpinIncr=1000")

dw_1.Modify("emp_bonus.EditMask.SpinRange='0~~~~5000'")
```

Elevation

Description

The elevation in a 3D graph.

Applies to

Graph objects

Syntax

Dot notation:

dw_control.Object.*graphname*.Elevation

Describe and Modify argument:

"*graphname*.Elevation { = ' *integer* ' }"

Parameter	Description
graphname	The name of the graph object in the DataWindow for which you want to get or set the elevation
integer	(*exp*) An integer specifying the elevation of the graph. Elevation can be a quoted DataWindow painter expression

Usage

In the painter Set the value using the Object property sheet, Graph tab, Elevation scrollbar (available when a 3D graph type is selected).

Examples

```
setting = dw_1.Object.graph_1.Elevation

dw_1.Object.graph_1.Elevation = 35

setting = dw_1.Describe("graph_1.Elevation")

dw_1.Modify("graph_1.Elevation=35")

dw_1.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 objects

Syntax

Dot notation:

dw_control.Object.*rrectname*.EllipseHeight

Describe and Modify argument:

"*rrectname*.EllipseHeight { = ' *integer* ' }"

Parameter	Description
rrectname	The name of the RoundRectangle object in the DataWindow for which you want to get or set the ellipse height
integer	(*exp*) 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 painter expression

Usage

In the painter Settable only in code, not in the painter.

Examples

```
setting = dw_1.Object.rrect_1.EllipseHeight

dw_1.Object.rrect_1.EllipseHeight = 35

setting = dw_1.Describe("rrect_1.EllipseHeight")

dw_1.Modify("rrect_1.EllipseHeight=35")

dw_1.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 objects

Syntax

Dot notation:

dw_control.Object.*rrectname*.EllipseWidth

Describe and Modify argument:

"*rrectname*.EllipseWidth { = ' *integer* ' }"

Parameter	Description
rrectname	The name of the RoundRectangle object 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. EllipseHeight can be a quoted DataWindow painter expression

Usage

In the painter Settable only in code, not in the painter.

Examples

```
setting = dw_1.Object.rrect_1.EllipseWidth

dw_1.Object.rrect_1.EllipseWidth = 35

setting = dw_1.Describe("rrect_1.EllipseWidth")

dw_1.Modify("rrect_1.EllipseWidth=35")

dw_1.Modify("rrect_1.EllipseWidth='10~tIf(...,20,30)'")
```

Expression

Description

The expression for a computed field object in the DataWindow. The expression is made up of calculations and DataWindow painter functions. The DataWindow evaluates the expression to get the value it will display in the computed field.

Applies to

Computed field objects

Syntax

Dot notation:

dw_control.Object.*computename*.Expression

Describe and Modify argument:

"*computename*.Expression { = '*string* ' }"

Parameter	Description
computename	The name of the computed field object in the DataWindow for which you want to get or set the expression
string	A string whose value is the expression for the computed field

Usage

In the painter Set the value using the Object property sheet, General tab, Expression option. The More button displays the Modify Expression dialog, which provides help in specifying the expression. The Verify button tests the expression.

Examples

```
setting = dw_1.Object.comp_1.Expression

dw_1.Object.comp_1.Expression = "avg(salary for all)"

setting = dw_1.Describe("comp_1.Expression")

dw_1.Modify("comp_1.Expression='avg(salary for all)'")
```

Filename

Description

The filename containing the bitmap for a Bitmap object in the DataWindow.

Applies to

Bitmap objects

Syntax

Dot notation:

dw_control.Object.*bitmapname*.Filename

Describe and Modify argument:

"*bitmapname*.Filename { = ' *filestring* ' }"

Parameter	Description
bitmapname	The name of the bitmap object in the DataWindow for which you want to get or set the filename
filestring	(*exp*) A string containing the name of the file that contains the bitmap. *Filestring* can be an quoted DataWindow painter expression If you include the name of the file containing the bitmap in the executable for the application, PowerBuilder will always use that bitmap; you cannot use Modify to change the bitmap

Usage

In the painter Set the value using the Picture Object property sheet, General tab, File Name option.

Examples

```
setting = dw_1.Object.bitmap_1.Filename

dw_1.Object.bitmap_1.Filename = "exclaim.bmp"

setting = dw_1.Describe("bitmap_1.Filename")

dw_1.Modify("bitmap_1.Filename='exclaim.bmp'")
```

FirstRowOnPage

Description

The first row currently visible in the DataWindow.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.FirstRowOnPage

Describe argument:

"DataWindow.FirstRowOnPage"

Examples

```
setting = dw_1.Object.DataWindow.FirstRowOnPage
setting = dw_1.Describe("DataWindow.FirstRowOnPage")
```

Font.Bias

Description

The way fonts are manipulated in the DataWindow during execution.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.Font.Bias

Describe and Modify argument:

"DataWindow.Font.Bias { = *biasvalue* }"

Parameter	Description
biasvalue	An integer indicating how the fonts will be manipulated at execution. *Biasvalue* cannot be a DataWindow painter expression. Values are: 0 — As display fonts 1 — As printer fonts 2 — Neutral; no manipulation will take place

Examples

```
setting = dw_1.Object.DataWindow.Font.Bias
dw_1.Object.DataWindow.Font.Bias = 1
setting = dw_1.Describe("DataWindow.Font.Bias")
dw_1.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 objects

Syntax

Dot notation:

dw_control.Object.*objectname*.Font.*property*

Describe and Modify argument:

"*objectname*.Font.*property* { = ' *value* ' }"

SyntaxFromSQL:

Column(Font.*property* = *value*)

Text(Font.*property* = *value*)

Parameter	Description
objectname	The name of a column, computed field, or text object for which you want to get or set font properties. For a column, you can specify its name or a pound sign (#) followed by the column number When generating DataWindow syntax with SyntaxFromSQL, the Font settings apply to all columns or all text objects
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 painter expression

Property for Font	Value
CharSet	(*exp*) 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 Painter: Set indirectly via font selection
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 Painter: Expressions tab (specify a constant or a conditional expression)

Property for Font	Value
Face	(*exp*) A string specifying the name of the font face, such as Arial or Courier Painter: Font tab (Font option) or StyleBar
Family	(*exp*) 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 Painter: Set indirectly via font selection
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 Painter: Font tab, Size option (specified in points) or StyleBar or Expressions tab
Italic	(*exp*) Whether the text should be italic. The default is no. Painter: Font tab (Font styles option) or StyleBar or Expressions tab
Pitch	(*exp*) The pitch of the font. Values are: 0 — The default pitch for your system 1 — Fixed 2 — Variable Painter: Set indirectly via font selection
Strikethrough	(*exp*) Whether the text should be crossed out. The default is no Painter: Font tab, Effects group, Strikeout checkbox or Expressions tab
Underline	(*exp*) Whether the text should be underlined. The default is no Painter: Font tab, Effects group, Underline checkbox or StyleBar or Expressions tab
Weight	(*exp*) An integer specifying the weight of the text; for example, 400 for normal or 700 for bold Painter: Font styles option or StyleBar or Expressions tab

Property for Font	Value
Width	(*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 Painter: Set indirectly via font selection

Usage

In the painter Set the value using:

- Object property sheet, Font tab
- For some font settings, StyleBar (useful for multiple objects)
- For some font settings, Object property sheet, Expressions tab (use a conditional expression)

Examples

```
dw_1.Object.emp_name_t.Font.Face

dw_1.Object.emp_name_t.Font.Face = "Arial"

dw_1.Describe("emp_name_t.Font.Face")

dw_1.Modify("emp_name_t.Font.Face='Arial'")
```

Footer.*property*

See Bandname.property.

Format

Description

The display format for a column.

You can use the GetFormat and SetFormat functions 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 objects

Syntax

Dot notation:

dw_control.Object.*objectname*.Format

Describe and Modify argument:

"*objectname*.Format { = ' *value* ' }"

Parameter	Description
objectname	The name of the column or computed field for which you want to get or set the display format
value	(*exp*) A string specifying the display format. See the *PowerBuilder User's Guide* for information on constructing display formats. *Value* can be a quoted DataWindow painter expression

Usage

In the painter Set the value using:

- Object property sheet, Format tab, Format option
- Object property sheet, Expressions tab (use a conditional expression)

Examples

```
setting = dw_1.Object.phone.Format

dw_1.Object."phone.Format = "[red](@@@)@@@-@@@@;'None'"

setting = dw_1.Describe("phone.Format")

dw_1.Modify( &
"phone.Format='[red](@@@)@@@-@@@@;~~~'None~~~''")
```

See also

GetFormat function in the *PowerScript Reference*
SetFormat function in the *PowerScript Reference*

GraphType

Description

The type of graph, such as bar, pie, column, and so on.

Applies to

Graph objects

Syntax

Dot notation:

dw_control.Object.*graphname*.GraphType

Describe and Modify argument:

"*graphname*.GraphType { = ' *typeinteger* ' }"

Parameter	Description
graphname	The graph object for which you want to get or change the type
typeinteger	(*exp*) An integer identifying the type of graph in the DataWindow object. *Typeinteger* can be a quoted DataWindow painter expression. Values are: 1 — Area 2 — Bar 3 — Bar3D 4 — Bar3DObj 5 — BarStacked 6 — BarStacked3DObj 7 — Col 8 — Col3D 9 — Col3DObj 10 — ColStacked 11 — ColStacked3DObj 12 — Line 13 — Pie 14 — Scatter 15 — Area3D 16 — Line3D 17 — Pie3D

Usage

In the painter Set the value using the Object property sheet, Graph tab, Graph Type option. Select the picture that represents the graph type.

Examples

```
setting = dw_1.Object.graph_1.GraphType

dw_1.Object.graph_1.GraphType = 17

setting = dw_1.Describe("graph_1.GraphType")

dw_1.Modify("graph_1.GraphType=17")
```

Grid.ColumnMove

Description

Whether the user can rearrange columns by dragging.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.Grid.ColumnMove

Describe and Modify argument:

"DataWindow.Grid.ColumnMove { = *value* } "

Parameter	Description
value	Whether the user can rearrange columns. Values are: Yes — The user can drag columns No — The user cannot drag columns

Usage

In the painter Set the value using the DataWindow Object property sheet, General tab, Grid group, Column Moving checkbox (available when the presentation style is Grid or Crosstab).

Examples

```
setting = dw_1.Object.DataWindow.Grid.ColumnMove

dw_1.Object.DataWindow.Grid.ColumnMove = No

setting = dw_1.Describe("DataWindow.Grid.ColumnMove")

dw_1.Modify("DataWindow.Grid.ColumnMove=No")
```

Grid.Lines

Description

The way grid lines display and print in a DataWindow whose presentation style is Grid or Crosstab.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.Grid.Lines

Describe and Modify argument:

"DataWindow.Grid.Lines { = *value* }"

Parameter	Description
value	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

Usage

In the painter Set the value using the DataWindow Object property sheet, General tab, Grid group, Display option (available when the presentation style is Grid or Crosstab).

Examples

```
setting = dw_1.Object.DataWindow.Grid.Lines

dw_1.Object.DataWindow.Grid.Lines = 2

setting = dw_1.Describe("DataWindow.Grid.Lines")

dw_1.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 objects

Syntax

Dot notation:

dw_control.Object.*oleobjectname*.GroupBy

Describe and Modify argument:

"*oleobjectname*.GroupBy { = ' *columnlist* ' }"

Parameter	Description
oleobjectname	The name of the OLE container object 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 painter expression

Usage

Target and Range also affect the data that is transferred to the OLE object.

In the painter Set the value using the Object property sheet, Data tab, Group By option.

Examples

```
ls_data = dw_1.Object.ole_report.GroupBy

dw_1.Object.ole_report.GroupBy = "emp_state, emp_office"
```

```
dw_1.Object.ole_report.GroupBy = "year"

ls_data = dw_1.Describe("ole_report.GroupBy")

dw_1.Modify(" &
    ole_report.GroupBy='emp_state, emp_office'")

dw_1.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

SyntaxFromSQL:

Style (Header_Bottom_Margin = *value*)

Parameter	Description
value	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

Examples

```
SQLCA.SyntaxFromSQL(sqlstring, &
'Style(...Header_Bottom_Margin = 25 ...)', &
errstring)
```

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

SyntaxFromSQL:

Style (Header_Top_Margin = *value*)

Parameter	Description
value	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

Examples

```
SQLCA.SyntaxFromSQL(sqlstring, &
'Style(...Header_Top_Margin = 500 ...)', errstring)
```

Header.*property*

See Bandname.property.

Header.*#.property*

See Bandname.property.

Height

Description

The height of an object in the DataWindow.

Applies to

Bitmap, Button, Column, Computed Field, Graph, GroupBox, OLE, Oval, Rectangle, Report, RoundRectangle, TableBlob, and Text objects

Syntax

Dot notation:

dw_control.Object.*objectname*.Height

Describe and Modify argument:

"*objectname*.Height { = ' *value* ' }"

Parameter	Description
objectname	The object within the DataWindow whose height you want to get or set

Parameter	Description
value	(*exp*) An integer specifying the height of the object in the unit of measure specified for the DataWindow. *Value* can be a quoted DataWindow painter expression

Usage

In the painter Set the value using:

- Object property sheet, Position tab, Height option
- Object property sheet, Expressions tab (use a conditional expression)

Examples

```
setting = dw_1.Object.empname.Height

dw_1.Object.empname.Height = 50

setting = dw_1.Describe("empname.Height")

dw_1.Modify("empname.Height=50")
```

Height.AutoSize

Description

Whether the object's width should be held constant and its height adjusted so that all the data is visible.

Applies to

Column, Computed Field, and Text objects

Syntax

Dot notation:

dw_control.Object.*objectname*.Height.AutoSize

Describe and Modify argument:

"*objectname*.Height.AutoSize { = *value* }"

Parameter	Description
objectname	The object for which you want to get or set the AutoSize property
value	Whether the width or height of the object 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 and change the width so that all data is visible Yes — Use the width defined in the painter and calculate the height so that all the data is visible

Usage

In the painter Set the value using the Object property sheet, Position tab, Autosize Height checkbox.

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).

Examples

```
setting = dw_1.Object.empname.Height.AutoSize

dw_1.Object.empname.Height.AutoSize = "Yes"

setting = dw_1.Describe("empname.Height.AutoSize")

dw_1.Modify("empname.Height.AutoSize=Yes")
```

Help.*property*

Description

Settings for customizing the Help topics associated with DataWindow dialog boxes.

FOR INFO For more information about Help, see the ShowHelp function in the *PowerScript Reference*.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.Help.*property*

Describe and Modify argument:

"DataWindow.Help.*property* { = *value* }"

Parameter	Description
property	A property for specifying DataWindow Help. Help properties and their settings are listed in the table below. The File property must have a valid filename before the rest of the Help property settings are valid
value	The value to be assigned to the property. For Help properties, *value* cannot be a DataWindow painter expression

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:\PB30\MYHELP.HLP). When this property has a value, Help buttons display on the DataWindow dialog boxes during execution
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 may display when the ImportFile function is called in a script
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 function in a script
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 function is called in a script
TypeID.SaveAs	A string specifying the Help topic for the Save As dialog box, which may display when the Save As function is called in a script
TypeID. SetCrosstab	A string specifying the Help topic for the Crosstab Definition dialog box, which may display when the CrosstabDialog function is called in a script
TypeID.SetFilter	A string specifying the Help topic for the Set Filter dialog box, which may display when the SetFilter and Filter functions are called in a script
TypeID.SetSort	A string specifying the Help topic for the Set Sort dialog box, which may display when the SetSort and Sort functions are called in a script

Property for Help	Value
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

Examples

```
setting = dw_1.Object.DataWindow.Help.Command

dw_1.Object.DataWindow.Help.File = "c:\pb40\myhelp.hlp"

dw_1.Object.DataWindow.Help.Command = 1

setting = dw_1.Describe("DataWindow.Help.Command")

dw_1.Modify("DataWindow.Help.File='c:\pb40\myhelp.hlp'")

dw_1.Modify("DataWindow.Help.Command=1")

dw_1.Modify( &
"DataWindow.Help.TypeID.SetFilter='filter_topic'")

dw_1.Modify("DataWindow.Help.TypeID.Retrieve.Criteria" &
+ " = 'criteria_topic'")
```

HideSnaked

Description

Whether the object appears only once per page when you print the DataWindow using the newspaper columns format.

Applies to

Bitmap, Button, Column, Computed Field, Graph, GroupBox, Line, OLE, Oval, Rectangle, Report, RoundRectangle, TableBlob, and Text objects

Syntax

Dot notation:

dw_control.Object.*objectname*.HideSnaked

Describe and Modify argument:

"*objectname*.HideSnaked { = ' *value* ' }"

Parameter	Description
objectname	The name of the object for which you want to get or set its HideSnaked setting

Parameter	Description
value	(*exp*) Whether the object appears once or multiple times in the printed output when the output has multiple columns (like a newspaper). Values are: Yes — The object will appear only once on a page No — The object will appear in each column on a page *Value* can be a quoted DataWindow painter expression

Usage

In the painter Set the value using the Object property sheet, General tab, Suppress Print After First Newspaper Column checkbox.

Examples

```
setting = dw_1.Object.graph_1.HideSnaked

dw_1.Object.text_title.HideSnaked = "yes"

setting = dw_1.Describe("graph_1.HideSnaked")

dw_1.Modify("text_title.HideSnaked=yes")
```

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

SyntaxFromSQL:

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

```
SQLCA.SyntaxFromSQL(sqlstring, &
'Style(...Horizontal_Spread = 25 ...)', errstring)
```

HorizontalScrollMaximum

Description

The maximum width of the scroll box of the DataWindow's horizontal scrollbar. This value is set by PowerBuilder 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

Dot notation:

dw_control.Object.DataWindow.HorizontalScrollMaximum

Describe argument:

"DataWindow.HorizontalScrollMaximum"

Examples

```
setting = &
dw_1.Object.DataWindow.HorizontalScrollMaximum
```

```
setting = &
dw_1.Describe("DataWindow.HorizontalScrollMaximum")
```

HorizontalScrollMaximum2

Description

The maximum width of the second scroll box when the horizontal scrollbar is split (HorizontalScrollSplit is greater than 0). This value is set by PowerBuilder based on the content of the DataWindow. Use HorizontalScrollMaximum2 with HorizontalScrollPosition2 to synchronize horizontal scrolling in multiple DataWindow objects.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.HorizontalScrollMaximum2

Describe argument:

"DataWindow.HorizontalScrollMaximum2"

Examples

```
setting = &
dw_1.Object.DataWindow.HorizontalScrollMaximum2
```

```
setting = &
dw_1.Describe("DataWindow.HorizontalScrollMaximum2")
```

HorizontalScrollPosition

Description

The position of the scroll box in the horizontal scrollbar. Use HorizontalScrollMaximum with HorizontalScrollPosition to synchronize horizontal scrolling in multiple DataWindow objects.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.HorizontalScrollPosition

Describe and Modify argument:

"DataWindow.HorizontalScrollPosition { = *scrollvalue* }"

Parameter	Description
scrollvalue	An integer specifying the position of the scroll box in the horizontal scrollbar of the DataWindow

Examples

```
spos1 = dw_1.Object.DataWindow.HorizontalScrollPosition

smax1 = dw_1.Describe( &
"DataWindow.HorizontalScrollMaximum")
spos1 = dw_1.Describe( &
"DataWindow.HorizontalScrollPosition")
smax2 = dw_2.Describe( &
"DataWindow.HorizontalScrollMaximum")
pos2 = Integer(spos1) * Integer(smax2) / Integer(smax1)
modstring = "DataWindow.HorizontalScrollPosition=" &
+ String(pos2)
dw_1.Modify(modstring)
```

HorizontalScrollPosition2

Description

The position of the scroll box in the second portion of the horizontal scrollbar when the scrollbar is split (HorizontalScrollSplit is greater than 0). Use HorizontalScrollMaximum2 with HorizontalScrollPosition2 to synchronize horizontal scrolling in multiple DataWindow objects.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.HorizontalScrollPosition2

Describe and Modify argument:

"DataWindow.HorizontalScrollPosition2 { = *scrollvalue* }"

Parameter	Description
scrollvalue	An integer specifying the position of the scroll box in the second portion of a split horizontal scrollbar of the DataWindow

Examples

```
spos = dw_1.Object.DataWindow.HorizontalScrollPosition2

dw_1.Object.DataWindow.HorizontalScrollPosition2 = 200

spos = dw_1.Describe( &
"DataWindow.HorizontalScrollPosition2")

dw_1.Modify("DataWindow.HorizontalScrollPosition2=200")
```

HorizontalScrollSplit

Description

The position of the split in the DataWindow's horizontal scrollbar. If HorizontalScrollSplit is zero, the scrollbar is not split.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.HorizontalScrollSplit

Describe and Modify argument:

"DataWindow.HorizontalScrollSplit { = *splitdistance* }"

Parameter	Description
splitdistance	An integer indicating where the split will occur in the horizontal scrollbar in a DataWindow object in the unit of measure specified for the DataWindow object

Examples

```
setting = dw_1.Object.DataWindow.HorizontalScrollSplit

dw_1.Object.DataWindow.HorizontalScrollSplit = 250

setting = &
dw_1.Describe("DataWindow.HorizontalScrollSplit")

dw_1.Modify("DataWindow.HorizontalScrollSplit=250")
```

HTextAlign

Description

The way text in a button is horizontally aligned.

Applies to

Button objects

Syntax

Dot notation:

dw_control.Object.*buttonname*.HTextAlign

Describe and Modify argument:

"*buttonname*.HTextAlign { = ' *value* ' }"

Parameter	Description
buttonname	The name of the button for which you want to align text
value	An integer indicating how the button text is horizontally aligned. Values are: 0 — Center 1 — Left 2 — Right

Usage

In the painter Set the value using the Button object property sheet, General tab.

Examples

```
dw_1.Object.b_name.HTextAlign = "1"

setting = dw_1.Describe("b_name.HTextAlign")

dw_1.Modify("b_name.HTextAlign ='1'")
```

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 a HTML page. They are particularly useful when used with Web.PB to create HTML pages dynamically.

Applies to

DataWindow objects

Syntax

Dot notation:

dw_control.Object.DataWindow.HTMLTable.*property*

Describe and Modify argument:

"DataWindow.HTMLTable.*property* { = ' *value* ' }"

Parameter	Description
property	A property for a DataWindow to be displayed in HTML table format. 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 painter expression

Property for HTMLTable	Value
Border	(*exp*) Border attribute for the HTMLTable element. The default is 1 (line around the table)
CellPadding	(*exp*) CellPadding attribute for the HTMLTable element. The default is 0
CellSpacing	(*exp*) CellSpacing attribute for the HTMLTable element. The default is 0
GenerateCSS	(*exp*) 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, which control style sheet use. The default is no
NoWrap	(*exp*) NoWrap attribute for the HTMLTable element. The default is to include this attribute
StyleSheet	(*exp*) HTML cascading style sheet generated for the DataWindow

Property for HTMLTable	Value
Width	Width attribute for the HTMLTable element. The default is 0

Examples

```
dw_1.Object.DataWindow.HTMLTable.Border = "2"

setting = dw_1.Describe
    ("DataWindow.HTMLTable.StyleSheet")

dw_1.Modify("DataWindow.HTMLTable.NoWrap = 'yes'")
```

ID

Description

The number of the column or TableBlob.

Applies to

Column and TableBlob objects

Syntax

Dot notation:

dw_control.Object.*objectname*.ID

Describe and Modify argument:

"*objectname*.ID"

Parameter	Description
objectname	The name of the column or TableBlob for which you want the ID number

Examples

```
setting = dw_1.Object.empname.ID

setting = dw_1.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 objects

Syntax

Dot notation:

dw_control.Object.*columnname*.Identity

Describe and Modify argument:

"*columnname*.Identity { = ' *value* ' }"

Parameter	Description
columnname	A string containing the name of the column for which you want to get or set its identity property
value	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

Examples

```
dw_1.Object.empid.Identity = "yes"

dw_1.Modify("empid.Identity='yes'")
```

Initial

Description

The initial value of the column in a newly inserted row.

Applies to

Column objects

Syntax

Dot notation:

dw_control.Object.*columnname*.Initial

Describe and Modify argument:

"*columnname*.Initial { = ' *initialvalue* ' }"

Parameter	Description
columnname	A string containing the name of the column for which you want to get or set its initial property

Parameter	Description
initialvalue	A string containing the initial value of the column. Special values include: Empty — A string of length 0 Null — No value Spaces — All blanks Today — Current date, time, or date and time

Examples

```
setting = dw_1.Object.empname.Initial

dw_1.Object.empname.Initial = "empty"

setting = dw_1.Describe("empname.Initial")

dw_1.Modify("empname.Initial='empty'")

dw_1.Modify("empstatus.Initial='A'")
```

Invert

Description

The way the colors in a Bitmap object are displayed, either inverted or normal.

Applies to

Bitmap objects

Syntax

Dot notation:

dw_control.Object.*bitmapname*.Invert

Describe and Modify argument:

"*bitmapname*.Invert { = ' *number* ' }"

Parameter	Description
bitmapname	The name of the bitmap object in the DataWindow for which you want to invert the colors
number	(*exp*) A boolean number indicating whether the colors of the bitmap will display inverted. Values are: 0 — (Default) No; do not invert the bitmap's colors 1 — Yes; display the bitmap with colors inverted *Number* can be a quoted DataWindow painter expression

Usage

In the painter Set the value using the Picture Object property sheet, General tab, Invert Image checkbox.

Examples

```
setting = dw_1.Object. bitmap_1.Invert

dw_1.Object.bitmap_1.Invert="0~tIf(empstatus='A',0,1)"

setting = dw_1.Describe("bitmap_1.Invert")

dw_1.Modify( &
"bitmap_1.Invert='0~tIf(empstatus=~~~'A~~~',0,1)'")
```

Key

Description

Whether the column is part of the database table's primary key.

Applies to

Column objects

Syntax

Dot notation:

dw_control.Object.*columnname*.Key

Describe and Modify argument:

"*columnname*.Key { = *value* }"

Parameter	Description
columnname	The column for which you want to get or set primary key status
value	Whether the column is part of the primary key. Values are: Yes — The column is part of the primary key No — The column is not part of the key

Usage

In the painter Set the value using the Rows menu, Update Properties.

Examples

```
setting = dw_1.Object.empid.Key

dw_1.Object.empid.Key = "Yes"

setting = dw_1.Describe("empid.Key")

dw_1.Modify("empid.Key=Yes")
```

KeyClause

Description

An expression to be used as the key clause when retrieving the blob.

Applies to

TableBlob objects

Syntax

Dot notation:

dw_control.Object.*tblobname*.KeyClause

Describe and Modify argument:

"*tblobname*.KeyClause { = ' *keyclause* ' }"

Parameter	Description
tblobname	The name of the TableBlob for which you want to specify a key clause
keyclause	(*exp*) A string that will be built into a key clause using the substitutions provided. The key clause can be any valid WHERE clause. *Keyclause* can be a quoted DataWindow painter expression

Usage

In the painter Set the value using the Object property sheet, Definition tab, Key Clause option.

Examples

With the following setting, the value of key_col will be put in col2 when PowerBuilder constructs the WHERE clause for the SELECTBLOB statement.

```
dw_1.Modify(blob_1.KeyClause='Key_col = :col2'")
```

Label.*property*

Description

Settings for a DataWindow whose presentation style is Label.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.Label.*property*

Describe and Modify argument:

"DataWindow.Label.*property* { = *value* }"

SyntaxFromSQL:

DataWindow(Label.*property* = *value*)

Parameter	Description
property	A property for the Label presentation style. Properties and their settings are listed in the table below
value	The value to be assigned to the property. For Label properties, *value* cannot be a DataWindow painter expression

Property for Label	Value
Columns	An integer indicating the number of columns of labels on a sheet Painter: Label group, Labels Across option
Columns.Spacing	An integer indicating the space between columns of labels in the units specified for the DataWindow object Painter: Margins group, Between Columns option
Ellipse_Height	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
Ellipse_Width	An integer radius of the horizontal part of the label in the unit of measure specified for the DataWindow object Painter: Not set in painter
Height	An integer specifying the height of a label in the units specified for the DataWindow object Painter: Label group, Height option
Name	A string containing the name of a label Painter: Predefined Label option
Rows	An integer indicating the number rows of labels on a sheet Painter: Label group, Down option
Rows.Spacing	An integer indicating the space between rows of labels on a sheet in the units specified for the DataWindow object Painter: Margins group, Between Rows option

Property for Label	Value
Shape	A string specifying the shape of a label. Values are: Rectangle RoundRectangle Oval Painter: Not set in painter
Sheet	(*Describe only*) Whether the paper is sheet fed or continuous. Values are: Yes — Sheet fed No — Continuous Painter: Paper group
TopDown	(*Describe only*) Whether the labels will be printed from the top to the bottom or across the page. Values are: No — Print labels across the page Yes — Print labels from top to bottom Painter: Arrange group
Width	An integer specifying the width of a label in the units specified for the DataWindow object Painter: Label group, Width option

Usage

In the painter Set the value using the DataWindow Object property sheet, Definition tab (when presentation style is Label).

Examples

```
setting = dw_1.Object.DataWindow.Label.Sheet

dw_1.Object.DataWindow.Label.Width = 250

setting = dw_1.Describe("DataWindow.Label.Sheet")

dw_1.Modify("DataWindow.Label.Width=250")

dw_1.Modify("DataWindow.Label.Height=150")

dw_1.Modify("DataWindow.Label.Columns=2")

dw_1.Modify("DataWindow.Label.Width=250")

dw_1.Modify("DataWindow.Label.Name='Address1'")
```

LabelDispAttr.*fontproperty*

See DispAttr.fontproperty.

LastRowOnPage

Description: The last row currently visible in the DataWindow.

Applies to: DataWindows

Syntax: Dot notation:

dw_control.Object.DataWindow.LastRowOnPage

Describe argument:

"DataWindow.LastRowOnPage"

Examples:

```
setting = dw_1.Object. DataWindow.LastRowOnPage
setting = dw_1.Describe("DataWindow.LastRowOnPage")
```

Left_Margin

Description: The size of the left margin of the DataWindow object.

Applies to: Style keywords

Syntax: SyntaxFromSQL:

Style (Left_Margin = *value*)

Parameter	Description
value	An integer specifying the size of the left margin in the units specified for the DataWindow

Examples:

```
SQLCA.SyntaxFromSQL(sqlstring, &
'Style( ... LeftMargin = 500 ... )', errstring)
```

Legend

Description

The location of the legend in a Graph object in a DataWindow.

Applies to

Graph objects

Syntax

Dot notation:

dw_control.Object.*graphname*.Legend

Describe and Modify argument:

"*graphname*.Legend { = ' *value* ' }"

Parameter	Description
graphname	The name of the graph object for which you want to specify the location of the legend
value	(*exp*) 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 painter expression

Usage

In the painter Set the value using the Object property sheet, Graph tab, Legend Location option (available when the graph has more than one series).

Examples

```
setting = dw_1.Object. graph_1.Legend

dw_1.Object.graph_1.Legend = 2

setting = dw_1.Describe("graph_1.Legend")

dw_1.Modify("graph_1.Legend=2")

dw_1.Modify("graph_1.Legend='2~tIf(dept_id=200,0,2)'")
```

Legend.DispAttr.*fontproperty*

See DispAttr.fontproperty.

Level

Description

The grouping level.

Level is only used in DataWindow syntax for the Create function.

Applies to

Group keywords

Syntax

Group (BY(*colnum1*, *colnum2*, ...) ... Level = *n* ...)

LinkUpdateOptions

Description

When the OLE object is linked, the method for updating the link information. If the user tries to activate the object and PowerBuilder cannot find the linked file, which breaks the link, LinkUpdateOptions controls whether PowerBuilder 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 a script.

Applies to

OLE objects

Syntax

Dot notation:

dw_control.Object.*oleobjectname*.LinkUpdateOptions

Describe and Modify argument:

"*oleobjectname*.LinkUpdateOptions { = ' *updatetype* ' }"

Parameter	Description
oleobjectname	The name of the OLE container object for which you want to get or set the link update method
updatetype	A number specifying how broken links will be reestablished. *Updatetype* can be a quoted DataWindow painter expression. Values are: ♦ LinkUpdateAutomatic! ♦ LinkUpdateManual!

Usage

In the painter Set the value using the Object property sheet, Options tab, Link Update option.

Examples

```
ls_data = dw_1.Object.ole_report.LinkUpdateOptions

dw_1.Object.ole_report.LinkUpdateOptions = 0
```

```
ls_data = dw_1.Describe("ole_report.LinkUpdateOptions")
dw_1.Modify("ole_report.LinkUpdateOptions='0'")
```

LineRemove

Description

(RichText presentation style only) Whether the line of text that contains the input field for the column or computed field is removed when the input field is empty. LineRemove is similar to the SlideUp property for objects in other presentation styles.

Applies to

Column and Computed Field objects in the RichText presentation style

Syntax

Dot notation:

dw_control.Object.*objectname*.LineRemove

Describe and Modify argument:

"*objectname*.LineRemove { = ' *value* ' }"

Parameter	Description
objectname	The name of the column or computed field whose line of text you want removed when the input field is empty
value	(*exp*) Whether the line of text is removed so that the rest of the text slides up when the input field for *objectname* is empty. Values are: ♦ Yes — The line of text will be removed when the input field is empty ♦ No — The line of text will not be removed *Value* can be a quoted DataWindow painter expression

Examples

```
setting = dw_1.Object.emp_street2.LineRemove
dw_1.Object.emp_street2.LineRemove = TRUE
setting = dw_1.Describe("emp_street2.LineRemove")
dw_1.Modify("emp_street2.LineRemove=yes")
```

Message.Title

Description

The title of the dialog box that displays when an error occurs.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.Message.Title

Describe and Modify argument:

"DataWindow.Message.Title { = ' *titlestring* ' }"

SyntaxFromSQL:

DataWindow(Message.Title = ' *titlestring* ')

Parameter	Description
titlestring	A string containing the title that displays in the title bar of the DataWindow dialog box that displays when an error occurs

Examples

```
setting = dw_1.Object.DataWindow.Message.Title

dw_1.Object.DataWindow.Message.Title = "Mistake!"

setting = dw_1.Describe("DataWindow.Message.Title")

dw_1.Modify("DataWindow.Message.Title='Bad, Bad, Bad'")

SQLCA.SyntaxFromSQL(sql_syntax, &
"Style(...) &
DataWindow(Message.Title='Sales Report' ...) ...", &
ls_Errors)
```

Moveable

Description

Whether the specified object in the DataWindow can be moved during execution. Movable objects should be in the DataWindow's foreground.

Applies to

Bitmap, Button, Column, Computed Field, Graph, GroupBox, Line, OLE, Oval, Rectangle, Report, RoundRectangle, TableBlob, and Text objects

Syntax

Dot notation:

dw_control.Object.*objectname*.Moveable

Describe and Modify argument:

"*objectname*.Moveable { = *number* }"

Parameter	Description
objectname	The object within the DataWindow for which you want to get or set its Moveable property (whether the user can move the object)
number	An boolean number specifying whether the object is moveable. Values are: 0 — False, the object is not movable 1 — True, the object is movable

Usage

In the painter Set the value using the Object property sheet, Position tab, Moveable checkbox.

Examples

```
setting = dw_1.Object.bitmap_1.Moveable

dw_1.Object.bitmap_1.Moveable = 1

setting = dw_1.Describe("bitmap_1.Moveable")

dw_1.Modify("bitmap_1.Moveable=1")
```

Multiline

Description

(RichText presentation style) Whether the column or computed field can contain multiple lines. Multiline is only effective when Width.Autosize is set to No.

Applies to

Column and Computed Field objects in the RichText presentation style

Syntax

Dot notation:

dw_control.Object.*objectname*.Multiline

Describe and Modify argument:

"*objectname*.Multiline { = ' *value* ' }"

Parameter	Description
objectname	The name of the column or computed field that will contain multiple lines

Parameter	Description
value	(*exp*) Whether the input field can contain multiline lines. Values are: ♦ Yes — The input field can contain multiple line ♦ No — The input field cannot contain multiple lines *Value* can be a quoted DataWindow painter expression

Usage

In the painter Set the value using the Object property sheet, Input Field or Compute tab, MultiLine option.

To display the property sheet, click the input field (column or computed field) to select it. Then right-click and select Properties from the popup menu.

Examples

```
setting = dw_1.Object.emp_street2.Multiline

dw_1.Object.emp_street2.Multiline = TRUE

setting = dw_1.Describe("emp_street2.Multiline")

dw_1.Modify("emp_street2.Multiline=yes")
```

Name

Description

The name of the object. PowerBuilder gives names to column and text objects that label columns.

For other types of objects, you must give them names in order to address them in scripts. Depending on settings you have made, you may find that PowerBuilder has already given an object a rather cryptic name because of other settings you have made. You can specify or change an object's name on its property sheet.

Applies to

Bitmap, Button, Column, Computed Field, Graph, GroupBox, Line, OLE, Oval, Rectangle, Report, RoundRectangle, TableBlob, and Text objects

Syntax

Dot notation:

dw_control.Object.*objectname*.Name

Describe argument:

"*objectname*.Name"

Parameter	Description
objectname	The object for which you want the name. For columns, you can specify the column number preceded by #

Usage

In the painter Set the value using the Object property sheet, General tab, Name option.

Examples

```
setting = dw_1.Object.#4.Name
setting = dw_1.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 objects

Syntax

Dot notation:

dw_control.Object.*reportname*.Nest_Arguments

Describe and Modify argument:

"*reportname*.Nest_Arguments { = *list* } "

Parameter	Description
reportname	The name of the nested report for which you want to supply retrieval argument values

Parameter	Description
list	A list of values for the retrieval arguments of the nested report. The format for the list is: (("*arg1*") {,("*arg2*") {,("*arg3*") {,... } } }) 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 during execution 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 during execution

Usage

In the painter Set the value using the Object property sheet, Arguments tab.

Examples

```
setting = dw_1.Object.rpt_1.Nest_Arguments

dw_1.Object.rpt_1.Nest_Arguments = &
"((~"cust_id~"),(~"'Eastern'~"))"

setting = dw_1.Describe("rpt_1.Nest_Arguments")

dw_1.Modify("rpt_1.Nest_Arguments" &
"=((~"cust_id~"),(~"'Eastern'~"))")

dw_1.Modify("rpt_1.Nest_Arguments=()")
```

Nested

Description

Whether the DataWindow contains nested DataWindows. Values returned are Yes or No.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.Nested

Describe argument:

"DataWindow.Nested"

Examples

```
setting = dw_1.Object.DataWindow.Nested
setting = dw_1.Describe("DataWindow.Nested")
```

NewPage (Group keywords)

Description

Whether a change in the value of a group column causes a page break.

Applies to

Group keywords

Syntax

SyntaxFromSQL:

Group (*colnum1*, *colnum2* NewPage)

Examples

```
SQLCA.SyntaxFromSQL(sql_syntax, &
"Style(Type=Group) " + &
"Group(#3 NewPage ResetPageCount)", &
ls_Errors)
```

NewPage (Report objects)

Description

Whether a nested report starts on a new page. NewPage only applies 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 objects

Syntax

Dot notation:

dw_control.Object.*reportname*.NewPage

Describe and Modify argument:

"*reportname*.NewPage { = *value* } "

Parameter	Description
reportname	The name of the report object for which you want to get or set its NewPage property
value	Whether the report begins a new page. Values are: Yes — Start the report on a new page No — Do not start the report on a new page

Usage

In the painter Set the value using the Report Object property sheet (in the Composite presentation style), General tab, Start on New Page checkbox.

Examples

```
newpage_setting = dw_1.Object.rpt_1.NewPage

dw_1.Object.rpt_1.NewPage = "Yes"

newpage_setting = dw_1.Describe("rpt_1.NewPage")

dw_1.Modify("rpt_1.NewPage-Yes")
```

Objects

Description

A list of the objects in the DataWindow object. If an object doesn't have a name, PowerBuilder assigns it an arbitrary name. The names are returned as a tab-separated list.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.Objects

Describe argument:

"DataWindow.Objects"

Examples

```
setting = dw_1.Describe("DataWindow.Objects")
```

OLE.Client.*property*

Description

Settings that some OLE server applications use to identify the client's information. The property values may be used to construct the title of the server window.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.OLE.Client.*property*

Describe and Modify argument:

"DataWindow.OLE.Client.*property* { = ' *value* ' }"

Parameter	Description
property	An OLE client property, as shown in the table below
value	Values for the properties are shown in the table below. *Value* cannot be a DataWindow painter expression

Property for OLE.Client	Value
Class	The client class for the DataWindow. The default is DataWindow
Name	The client name for the DataWindow. The default is Untitled

Usage

In the painter Set the value using the Database Blob Object property sheet, Definition tab.

Examples

```
ls_data = dw_1.Object.DataWindow.OLE.Client.Class

dw_1.Object.DataWindow.OLE.Client.Class = "PB"

ls_data = dw_1.Describe("DataWindow.OLE.Client.Class")

dw_1.Modify("DataWindow.OLE.Client.Class = 'PB'")
```

OLEClass

Description

The name of the OLE class for the TableBlob object.

Applies to

TableBlob objects

Syntax

Dot notation:

dw_control.Object.*tblobname*.OLEClass

Describe and Modify argument:

"*tblobname*.OLEClass { = ' *oleclassname* ' }"

Parameter	Description
tblobname	The TableBlob column for which you want to get or set the class of server application
oleclassname	(*exp*) A string specifying a class of an OLE server application installed on your system. *Oleclassname* is quoted and can be a DataWindow painter expression

Usage

In the painter Set the value using the Object property sheet, Definition tab, OLE Class: Description option.

Examples

```
setting = dw_1.Object.blob_1.OLEClass

dw_1.Object.blob_1.OLEClass = 'Word.Document'

setting = dw_1.Describe("blob_1.OLEClass")

dw_1.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 objects

Syntax

Dot notation:

dw_control.Object.*graphname*.OverlapPercent

Describe and Modify argument:

"*graphname*.OverlapPercent { = ' *integer* ' }"

Parameter	Description
graphname	The name of the graph object in the DataWindow object for which you want to get or set the percentage of overlap

Parameter	Description
integer	(*exp*) An integer specifying the percent of the width of the data markers that will overlap. *Integer* can be a quoted DataWindow painter expression

Usage

In the painter Set the value using the Object property sheet, Graph tab, Overlap (% of width) option (available when a series has been specified).

Examples

```
setting = dw_1.Object.graph_1.OverlapPercent

dw_1.Object.graph_1.OverlapPercent = 25

setting = dw_1.Describe("graph_1.OverlapPercent")

dw_1.Modify("graph_1.OverlapPercent=25")
```

Pen.*property*

Description

Settings for a line or the outline of an object.

Applies to

Line, Oval, Rectangle, and RoundRectangle objects

Syntax

Dot notation:

dw_control.Object.*objectname*.Pen.*property*

Describe and Modify argument:

"*objectname*.Pen.*property* { = *value* }"

Parameter	Description
objectname	The name of the object whose Pen property you want to get or set
property	A property that applies to the Pen characteristics of *objectname*, as listed in the table below
value	The value of the property, as shown in the table below. *Value* can be a quoted DataWindow painter expression

Property for Pen	Value
Color	(*exp*) A long specifying the color (the red, green, and blue values) to be used as the object's line color Painter: Color group, Line 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: Line option
Width	(*exp*) A number specifying the width of the line in the unit of measure specified for the DataWindow Painter: Line option. For wider lines, the style is always solid

Usage

In the painter Set the value using the Object property sheet, General tab.

Examples

```
setting = dw_1.Object.line_1.Pen.Width

dw_1.Object.line_1.Pen.Width = 10
```

```
setting = dw_1.Describe("line_1.Pen.Width")
dw_1.Modify("line_1.Pen.Width=10")
```

Perspective

Description

The distance the graph appears from the front of the window.

Applies to

Graph objects

Syntax

Dot notation:

dw_control.Object.*graphname*.Perspective

Describe and Modify argument:

"*graphname*.Perspective { = ' *integer* ' }"

Parameter	Description
graphname	The name of the graph object in the DataWindow object for which you want to get or set the perspective
integer	(*exp*) 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 painter expression

Usage

In the painter Set the value using the Object property sheet, Graph tab, Perspective scrollbar (available when a 3D graph type is selected).

Examples

```
setting = dw_1.Object.graph_1.Perspective
dw_1.Object.graph_1.Perspective = 20
setting = dw_1.Describe("graph_1.Perspective")
dw_1.Modify("graph_1.Perspective=20")
```

PictureName

Description

The name of the file containing the picture to be used on the button (if not specified, just the text is used).

Applies to

Button objects

Syntax

Dot notation:

dw_control.Object.*buttonname*.PictureName

Describe and Modify argument:

"*buttonname*.PictureName { = ' *string* ' }"

Parameter	Description
buttonname	The name of the button for which you want to use a picture
string	A string containing the name of the file containing the picture

Usage

In the painter Set the value using the Button object property sheet, General tab.

Examples

```
dw_1.Object.b_name.PictureName = "picturefile"
ls_data = dw_1.Describe("b_name.PictureName")
dw_1.Modify("b_name.PictureName = 'picturefile'")
```

Pie.DispAttr.*fontproperty*

See DispAttr.fontproperty.

Pointer

Description

The image to be used for the mouse pointer when the pointer is over the specified object. If you specify a pointer for the whole DataWindow, PowerBuilder uses that pointer except when the pointer is over an object that also has a Pointer setting.

Applies to

DataWindow, Bitmap, Button, Column, Computed Field, Graph, GroupBox, Line, OLE, Oval, Rectangle, Report, RoundRectangle, TableBlob, and Text objects

Syntax

Dot notation:

dw_control.Object.*objectname*.Pointer

Describe and Modify argument:

"*objectname*.Pointer { = ' *pointername* ' }"

Parameter	Description
objectname	The name of the object in the DataWindow for which you want to get or set the pointer. Specify DataWindow to specify the pointer for the whole DataWindow
pointername	(*exp*) A string specifying a value of the Pointer enumerated data type or the name of a cursor file (.CUR) to be used for the pointer. (See the SetPointer function for a list of Pointer values.) *Pointername* can be a quoted DataWindow painter expression

Usage

In the painter Set the value using:

- Object property sheet, Pointer tab
- Object property sheet, Expressions tab (use a conditional expression)

Examples

```
setting = dw_1.Object.graph_1.Pointer

dw_1.Object.graph_1.Pointer = 'Cross!'

setting = dw_1.Describe("graph_1.Pointer")

dw_1.Modify("graph_1.Pointer = 'Cross!'")

dw_1.Modify("graph_1.Pointer = 'c:\pb040\mycurs.cur'")
```

Print.Buttons

Description

Whether buttons display on the printed output.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.Print.Buttons

Describe and Modify argument:

"DataWindow.Print.Buttons { = *value* }"

Parameter	Description
value	Whether buttons display on the printed output. Values are: Yes — Buttons are displayed No — Buttons are not displayed

Usage

In the painter Set the value using the DataWindow object property sheet, Print Specifications tab.

Examples

```
dw_1.Object.DataWindow.Print.Buttons = 'Yes'

setting = dw_1.Describe("DataWindow.Print.Buttons")

dw_1.Modify("DataWindow.Print.Buttons = 'Yes'")
```

Print.Preview.Buttons

Description

Whether buttons display in print preview.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.Print.Preview.Buttons

Describe and Modify argument:

"DataWindow.Print.Preview.Buttons { = *value* }"

Parameter	Description
value	Whether buttons display in print preview. Values are: Yes — Buttons are displayed No — Buttons are not displayed

Usage

In the painter Set the value using the DataWindow object property sheet, Print Specification tab.

Examples

```
dw_1.Object.DataWindow.Print.Preview.Buttons = 'Yes'

setting = dw_1.Describe
    ("DataWindow.Print.Preview.Buttons")

dw_1.Modify("DataWindow.Print.Preview.Buttons = 'Yes'")
```

Print.*property*

Description

Properties that control the printing of a DataWindow.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.Print.*property*

Describe and Modify argument:

"DataWindow.Print.*property* { = *value* }"

SyntaxFromSQL:

DataWindow (Print.*property* = *value*)

Parameter	Description
property	A property for printing. Properties and their settings are listed in the table below
value	The value to be assigned to the property. *Value* cannot be a DataWindow painter expression

Property for Print	Value
Collate	Whether printing is collated. Note that collating is usually slower since the print is repeated to produce collated sets. Values are: Yes — Collate the pages of the print job No — (Default) Do not collate The user can specify the value in the system's Print dialog box if the printer driver supports it
Color	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
Columns	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 Painter: Newspaper Columns group, Across option

Property for Print	Value
Columns.Width	An integer specifying the width of the newspaper-style columns in the units specified for the DataWindow Painter: Newspaper Columns group, Width option
Copies	An integer indicating the number of copies to be printed The user can specify the value in the system's Print dialog box if the printer driver supports it
DocumentName	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 Painter: Document Name option
Duplex	An integer indicating the orientation of the printed output. Values are: 1 — Simplex (none) 2 — Horizontal 3 — Vertical The user can specify the value in the system's Print dialog box if the printer driver supports it
Filename	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: Not settable in painter
Margin.Bottom	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 SyntaxFromSQL to generate DataWindow syntax. Painter: Margins in PowerBuilder Units group, Bottom option
Margin.Left	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 SyntaxFromSQL to generate DataWindow syntax Painter: Margins in PowerBuilder Units group, Left option
Margin.Right	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 SyntaxFromSQL to generate DataWindow syntax Painter: Margins in PowerBuilder Units group, Right option

Property for Print	Value
Margin.Top	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 SyntaxFromSQL to generate DataWindow syntax Painter: Margins in PowerBuilder Units group, Top option
Orientation	An integer indicating the print orientation. Values are: 0 — The default orientation for your printer 1 — Landscape 2 — Portrait Painter: Paper group, Orientation option
Page.Range	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
Page.RangeInclude	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

Property for Print	Value
Paper.Size	An integer indicating the size of the paper that will be 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 11in 3 — Tabloid 17 x 11 inches 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 275mm 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 220mm 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 Painter: Paper group, Size option

Property for Print	Value
Paper.Source	An integer indicating the bin that will be used as the paper source. 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 Painter: Paper group, Source option
Preview	Whether the DataWindow object is displayed in preview mode. Values are: Yes — Display in preview mode No — (Default) Do not display in preview mode
Preview.Rulers	Whether the rulers display when the DataWindow object displays in preview mode: Yes — Display the rulers No — (Default) Do not display the rulers You can view rulers in Preview mode in the DataWindow painter. Choose File>Print Preview, then File>Print Preview Rulers. However, the setting is not used at runtime. To see rulers during execution, set Print.Preview.Rulers in code
Preview.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. Choose File>Print Preview, then File>Print Preview Zoom. However, the setting is not used at runtime. To change the zoom factor during execution, set Print.Preview.Zoom in code
Prompt	Whether a prompt will display before the job prints so the use can cancel the print job. Values are: Yes — (Default) Display a prompt before the job prints No — Do not display a prompt before the job prints Painter: Prompt Before Printing checkbox

Property for Print	Value
Quality	An integer indicating the quality of the output. Values are: 0 — Default 1 — High 2 — Medium 3 — Low 4 — Draft The user can specify the value in the system's Print dialog box if the printer driver supports it
Scale	An integer specifying the scale of the printed output as a percent The scaling percentage is passed to the print driver. If you have problems with scaling, you may be using a driver that does not support scaling The user can specify the value in the system's Print dialog box if the printer driver supports it FOR INFO For more information, see your print driver documentation

Usage

In the painter Set the value using the DataWindow Object property sheet, Print Specifications tab.

Examples

```
ls_data = dw_1.Object.DataWindow.Print.Scale

dw_1.Object.DataWindow.Print.Paper.Size = 3

ls_data - dw_1.Describe("DataWindow.Print.Scale")

dw_1.Modify("DataWindow.Print.Paper.Size = 3")

dw_1.Modify("DataWindow.Print.Margin.Top=500")
```

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

Dot notation:

dw_control.Object.DataWindow.Printer

Describe argument:

"DataWindow.Printer"

Examples

```
setting = dw_1.Object.DataWindow.Printer
setting = dw_1.Describe("DataWindow.Printer")
```

Processing

Description

The type of processing required to display the data in the selected presentation style.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.Processing

Describe argument:

"DataWindow.Processing"

Return values are:

0 — (Default) Form, group, query, or tabular
1 — Grid
2 — Label
3 — Graph
4 — Crosstab
5 — Composite

Examples

```
setting = dw_1.Object.DataWindow.Processing
setting = dw_1.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

Dot notation:

dw_control.Object.*columnname*.Protect

Describe and Modify argument:

"*columnname*.Protect { = ' *integer* ' }"

Parameter	Description
columnname	The name of the column for which you want to get or set the protection
integer	(exp) A boolean integer specifying whether the column is protected. Values are: 0 — False, the column is not protected 1 — True, the column is protected *Integer* can be a quoted DataWindow painter expression

Usage

A user cannot change a column value if any one of these conditions are 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 makes some values in the column protected but not others.

In the painter Set the value using the Object property sheet, Expressions tab (use a conditional expression).

Examples

```
setting = dw_1.Object.emp_stat.Protect

dw_1.Object.emp_stat.Protect=1

setting = dw_1.Describe("emp_stat.Protect")

dw_1.Modify("emp_stat.Protect=1")

dw_1.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

Dot notation:

dw_control.Object.DataWindow.QueryClear

Modify argument:

"DataWindow.QueryClear { = *value* }"

Parameter	Description
value	Remove the WHERE clause from a query Yes is the only valid value

Examples

```
dw_1.Object.DataWindow.QueryClear = "yes"
dw_1.Modify("DataWindow.QueryClear=yes")
```

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. After the user specifies retrieval criteria in query mode, subsequent calls to Retrieve will use the new criteria.

DataWindow presentation styles
You cannot use QueryMode with DataWindow objects that use any of the following presentation styles: N-Up, Label, Crosstab, RichText, and Graph.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.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 and the user's specification stored for the next call to Retrieve

Usage

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.

Examples

```
setting = dw_1.Object.DataWindow.QueryMode

dw_1.Object.DataWindow.QueryMode = "yes"

setting = dw_1.Describe("DataWindow.QueryMode")

dw_1.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, RichText, and Graph.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.QuerySort

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

```
setting = dw_1.Object.DataWindow.QuerySort

dw_1.Object.DataWindow.QuerySort = "yes"

setting = dw_1.Describe("DataWindow.QuerySort")

dw_1.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 objects

Syntax

Dot notation:

dw_control.Object.*columnname*.RadioButtons.*property*

Describe and Modify argument:

"*columnname*.RadioButtons.*property* { = *value* }"

Parameter	Description
columnname	The name of column that has the RadioButton edit style
property	A property for the RadioButton column. Properties and their settings are listed in the table below
value	The value to be assigned to the property. For RadioButton properties, *value* cannot be a DataWindow painter expression

Property for RadioButtons	Value
3D	Whether the radio buttons are 3D. Values are: Yes — Make the buttons 3D No — Do not make the buttons 3D Painter: Options group, 3D Look

Property for RadioButtons	Value
Columns	An integer constant specifying the number of columns of radio buttons Painter: Options group, Columns Across option
LeftText	Whether the text labels for the radio buttons are on the left side. Values are: Yes — The text is on the left of the radio buttons No — The text is on the right of the radio buttons Painter: Options group, Left Text option
Scale	Whether the circle is scaled to the size of the font. Scale has an effect only when 3D is No. Values are: Yes — Scale the circles No — Do not scale the circles Painter: Options group, Scale Circles option

Usage

In the painter Set the value using the Object property sheet, Edit tab when Style is RadioButtons.

Examples

```
setting = &
dw_1.Object.emp_gender.RadioButtons.LeftText

dw_1.Object.emp_gender.RadioButtons.LeftText = "no"

setting = &
dw_1.Describe("emp_gender.RadioButtons.LeftText")

dw_1.Modify("emp_gender.RadioButtons.LeftText=no")

dw_1.Modify("emp_gender.RadioButtons.3D=Yes")

dw_1.Modify("emp_gender.RadioButtons.Columns=2")
```

Range

Description

The rows in the DataWindow used in the graph or OLE object. Range can be all rows, the rows on the current page, a group that you've defined for the DataWindow, or the current row (OLE objects only).

Applies to

Graph and OLE objects

Syntax

Dot notation:

dw_control.Object.*objectname*.Range

Describe argument:

"*objectname*.Range"

Parameter	Description
objectname	The name of the graph object within the DataWindow that will display the graphed rows or the name of the OLE container object that holds an OLE object to which the specified range of rows will be transferred

Usage

Possible values are:

-2 — The current row (OLE objects 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 Set the value using the Object property sheet, Data tab, Rows option.

Examples

```
setting = dw_1.Object.graph_salary.Range

setting = dw_1.Object.ole_report.Range

setting = dw_1.Describe("graph_salary.Range")

setting = dw_1.Describe("ole_report.Range")
```

ReadOnly

Description

Whether the DataWindow is read-only. (DataWindows defined in the Report painter are automatically read-only.)

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.ReadOnly

Describe and Modify argument:

"DataWindow.ReadOnly { = *value* }"

Parameter	Description
value	Whether the DataWindow is read-only. Values are: Yes — Make the DataWindow read-only No — (Default) Do not make the DataWindow read-only

Examples

```
setting = dw_1.Object.DataWindow.ReadOnly

dw_1.Object.DataWindow.ReadOnly="Yes"

setting = dw_1.Describe("DataWindow.ReadOnly")

dw_1.Modify("DataWindow.ReadOnly=Yes")
```

Report

Description

Whether the DataWindow is a read-only report.

Applies to

Style keywords

Syntax

SyntaxFromSQL:

Style (Report = *value*)

Parameter	Description
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

```
SQLCA.SyntaxFromSQL(sqlstring, &
'Style(...Report = yes ...)', errstring)
```

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

SyntaxFromSQL:

Group (*col1* {*col2* ...} ... ResetPageCount)

Examples

```
SQLCA.SyntaxFromSQL(sql_syntax, &
"Style(Type=Group) " &
+ "Group(#3 NewPage ResetPageCount)", &
errorvar)
```

Resizeable

Description

Whether the user can resize the specified object.

Applies to

Bitmap, Button, Column, Computed Field, Graph, GroupBox, Line, OLE, Oval, Rectangle, Report, RoundRectangle, TableBlob, and Text objects

Syntax

Dot notation:

dw_control.Object.*objectname*.Resizeable

Describe and Modify argument:

"*objectname*.Resizeable { = *value* }"

Parameter	Description
objectname	The object within the DataWindow whose Resizeable setting you want to get or set
value	A boolean number indicating whether *objectname* can be resized. Values are: 0 — (False) The object cannot be resized 1 — (True) The object can be resized

Usage

In the painter Set the value using the Object property sheet, Position tab, Resizeable option.

Setting Border to Resize (General tab) also makes the object resizable.

Examples

```
setting = dw_1.Object.graph_1.Resizeable

dw_1.Object.graph_1.Resizeable = 1

setting = dw_1.Describe("graph_1.Resizeable")

dw_1.Modify("graph_1.Resizeable=1")

dw_1.Modify("bitmap_1.Resizeable=0")
```

Retrieve

Description

The SQL statement for the DataWindow.

Retrieve is only set in DataWindow syntax for the Create function.

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 function 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 haven't been viewed yet.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.Retrieve.AsNeeded

Describe and Modify argument:

"DataWindow.Retrieve.AsNeeded { = ' *value* ' }"

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 function is called

Usage

In the painter Set the value using Rows>Retrieve>Rows As Needed.

Examples

```
setting = dw_1.Object.DataWindow.Retrieve.AsNeeded
dw_1.Object.DataWindow.Retrieve.AsNeeded= "Yes"
setting = dw_1.Describe("DataWindow.Retrieve.AsNeeded")
dw_1.Modify("DataWindow.Retrieve.AsNeeded=Yes")
```

RichText.*property*

Description

Properties for the DataWindow RichText presentation style.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.RichText.*property*

Describe and Modify argument:

"DataWindow.RichText.*property* { = *value* }"

Parameter	Description
property	A property for the DataWindow RichText presentation style. Properties and appropriate values are listed in the table below
value	A value to be assigned to the property

Property for RichText	Value
BackColor	A long specifying the numeric value of the background color of the text editing area. Values are –2 to 16,777,215 FOR INFO For more information about color, see RGB. Painter: Background Color group, General option
DisplayOnly	Specifies whether users can make changes to the contents. Values are: ♦ Yes — The content, including text and input files, is protected (the user cannot edit it) ♦ No — The user can edit the content Painter: Display Only option
HeaderFooter	(Read-only) Specifies whether the RichTextEdit DataWindow has a header/footer section. This property must be set in the painter and cannot be changed during execution. Values are: ♦ Yes — The control has a header/footer section ♦ No — The control does not have a header/footer section If a document has a header or footer and the HeaderFooter property is set to no, then header/footer information in the document is ignored. If the document is then saved in the same file, the header/footer information is lost Painter: Header/Footer option

Property for RichText	Value
InputField BackColor	A long specifying the default background color for all input fields: –2 to 16,777,215 Painter: Background Color group, Input Field option
InputField NamesVisible	Specifies whether input field names are displayed in input fields, rather than the input field values. Values are: ♦ Yes — Input fields display their names ♦ No — Input fields display their data Painter: RichText Presentation group, Input Field Names Visible option
InputFields Visible	Not for external use
PictureFrame	Specifies whether pictures (bitmaps) are displayed as empty frames. Values are: ♦ Yes — Pictures are displayed as empty frames ♦ No — The pictures are displayed Painter: Pictures As Frame option
PopMenu	Specifies whether the user has access to a popup menu by clicking the right mouse button on the DataWindow. The menu allows the user to cut and paste, insert a file, and select formatting options. Values are: ♦ Yes — Popup menu is enabled ♦ No — Popup menu is disabled Painter: PopUp Menu option
ReadOnly	Specifies whether the user can change the data and the text in the DataWindow. Values are: ♦ Yes — The DataWindow is read-only (text and data cannot be modified) ♦ No — The text and the data can be modified
ReturnsVisible	Specifies whether carriage returns are visible. Values are: ♦ Yes — Carriage returns are visible ♦ No — Carriage returns are hidden Painter: RichText Presentation group, Returns Visible option

Property for RichText	Value
RulerBar	Specifies whether a ruler bar is visible above the editing area. If visible, the user can use it to see measurements while setting tabs and margins on the tab bar (see the TabBar property in this table). Values are: ♦ Yes — Ruler bar is visible ♦ No — Ruler bar is hidden If the RichTextEdit popup menu is enabled, the user can use it to turn ruler bar display on and off (see the PopMenu property in this table) Painter: RichText Bars group, Ruler option
SpacesVisible	Specifies whether spaces are visible. Values are: ♦ Yes — Spaces are visible ♦ No — Spaces are hidden Painter: RichText Presentation group, Spaces Visible option
TabBar	Specifies whether a bar for setting tabs is visible above the editing area. Values are: ♦ Yes — Tab bar is visible ♦ No — Tab bar is hidden If the popup menu is enabled, the user can use it to turn tab bar display on and off (see the PopMenu property in this table) Painter: RichText Bars group, Tab option
TabsVisible	Specifies whether tabs are visible. Values are: ♦ Yes — Spaces are visible ♦ No — Spaces are hidden Painter: RichText Presentation group, Tabs Visible option
ToolBar	Specifies whether a tool bar for formatting text is visible above the editing area. Values are: ♦ Yes — Tool bar is visible ♦ No — Tool bar is not visible If the popup menu is enabled, the user can use it to turn tool bar display on and off (see the PopMenu property in this table). Painter: RichText Bars group, Tool option

Property for RichText	Value
WordWrap	Specifies whether text wraps automatically to the next line when the line reaches the margin. Values are: ♦ Yes — Automatic wordwrap is enabled ♦ No — Automatic wordwrap is disabled Painter: Word Wrap option

Usage

In the painter Set the value using the DataWindow Object property sheet, General tab, when the presentation style is RichText.

Examples

```
setting = &
    dw_1.Object.DataWindow.RichText.DisplayOnly

dw_1.Object.DataWindow.RichText.PopMenu = "yes"

setting = &
    dw_1.Describe("DataWindow.RichText.DisplayOnly")

dw_1.Modify("DataWindow.RichText.PopMenu = 'yes'")
```

Rotation

Description

The degree of left-to-right rotation for the graph object within the DataWindow when the graph has a 3D type.

Applies to

Graph objects

Syntax

Dot notation:

dw_control.Object.*graphname*.Rotation

Describe and Modify argument:

"*graphname*.Rotation = { ' *integer* ' }"

Parameter	Description
graphname	The name of the Graph object for which you want to get or set the rotation
integer	(*exp*) The degree of rotation for the graph. Effective values range from -90 to 90. Integer can be a quoted DataWindow painter expression

Usage

In the painter Set the value using the Object property sheet, Graph tab, Rotation scrollbar (available when a 3D graph type is selected).

Examples

```
setting = dw_1.Object.graph_1.Rotation

dw_1.Object.graph_1.Rotation=25

setting = dw_1.Describe("graph_1.Rotation")

dw_1.Modify("graph_1.Rotation=25")

dw_1.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

Dot notation:

dw_control.Object.DataWindow.Row.Resize

Describe and Modify argument:

"DataWindow.Row.Resize { = *value* } "

Parameter	Description
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 Set the value using the DataWindow Object property sheet, General tab, Row Resize option (available when the presentation style is Grid or Crosstab).

Examples

```
setting = dw_1.Object.DataWindow.Row.Resize

dw_1.Object.DataWindow.Row.Resize = 0

setting = dw_1.Describe("DataWindow.Row.Resize")

dw_1.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

Syntax

Dot notation:

dw_control.Object.DataWindow.Rows_Per_Detail

Describe argument:

"DataWindow.Rows_Per_Detail"

SyntaxFromSQL:

DataWindow (... Rows_Per_Detail = *n* ...)

Parameter	Description
n	A long specifying the number of rows in each column

Examples

```
dw_1.SyntaxFromSQL(sqlselect, &
'DataWindow(...Rows_Per_Detail = 12 ...)', &
errstring)
```

Selected

Description

A list of selected objects within the DataWindow.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.Selected

Describe and Modify argument:

"DataWindow.Selected = ' *list* ' "

Parameter	Description
list	A list of the objects you want to select. In the list you designate a group of objects by specifying range of row numbers and a range of objects in the format: *startrow/endrow/startobject/endobject* To specify more that one group, separate each group with a semicolon: *startrow1/endrow1/startobj1/endobj1;startrow2/endrow2/startobj2/endobj2;...* Do not include spaces in the string. You must use column names, not column numbers

Examples

```
setting = dw_1.Object.DataWindow.Selected

dw_1.Object.DataWindow.Selected = &
"1/10/emp_id/emp_name;12/23/salary/status"

setting = dw_1.Describe("DataWindow.Selected")

dw_1.Modify("DataWindow.Selected=" &
"'1/10/emp_id/emp_name;12/23/salary/status'")
```

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

Grid DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.Selected.Data

Describe argument:

"DataWindow.Selected.Data"

Examples

```
setting = dw_1.Object.DataWindow.Selected.Data

setting = dw_1.Describe("DataWindow.Selected.Data")
```

Selected.Mouse

Description

Whether the user can use the mouse to select columns.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.Selected.Mouse

Describe and Modify argument:

"DataWindow.Selected.Mouse { = *value* }"

Parameter	Description
value	Whether the user can use the mouse to select columns. Values are: ♦ Yes — The mouse can be used ♦ No — The mouse cannot be used

Usage

In the painter Set the value using the DataWindow Object property sheet, General tab, Mouse Selection option (available when the presentation style is Grid or Crosstab).

Examples

```
setting = dw_1.Object.DataWindow.Selected.Mouse

dw_1.Object.DataWindow.Selected.Mouse = "Yes"

setting = dw_1.Describe("DataWindow.Selected.Mouse")

dw_1.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 objects

Syntax

Dot notation:

dw_control.Object.*graphname*.ShadeColor

Describe and Modify argument:

"*graphname*.ShadeColor { = ' *long* ' }"

Parameter	Description
graphname	The Graph object 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 or in PowerScript to calculate the desired color value. However, be sure to convert the return value of the PowerScript function to a string *Long* can be a quoted DataWindow painter expression

Usage

To set the shade color for individual series markers, such as bars or pie slices, use the function SetDataStyle.

In the painter Set the value using the Object property sheet, General tab, Shade Color option.

Examples

```
setting = dw_1.Object.graph_1.ShadeColor

dw_1.Object.graph_1.ShadeColor = 16600000

setting = dw_1.Describe("graph_1.ShadeColor")

dw_1.Modify("graph_1.ShadeColor=16600000")

dw_1.Modify("graph_1.ShadeColor=" + &
String(RGB(90,90,90)))

dw_1.Modify("graph_1.ShadeColor='0~t" &
+ If(salary>50000," &
+ String(RGB(100,90,90)) &
```

```
+ "," &
+ String(RGB(90,90,100)) &
+ ")'")
```

ShowDefinition

Description

Whether the DataWindow definition will display. The DataWindow will display the column names instead of data.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.ShowDefinition

Describe and Modify argument:

"DataWindow.ShowDefinition { = ' *value* ' }"

Parameter	Description
value	(*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 painter expression

Examples

```
setting = dw_1.Object.DataWindow.ShowDefinition
dw_1.Object.DataWindow.ShowDefinition = "Yes"
setting = dw_1.Describe("DataWindow.ShowDefinition")
dw_1.Modify("DataWindow.ShowDefinition=Yes")
```

SizeToDisplay

Description

Whether the graph should be sized automatically to the display area.

Applies to

Graph objects

Syntax

Dot notation:

dw_control.Object.*graphname*.SizeToDisplay

Describe and Modify argument:

"*graphname*.SizeToDisplay { = ' *value* ' }"

Parameter	Description
graphname	The graph object in the DataWindow for which you want to get or set its adjustability
value	(*exp*) A boolean number specifying whether to adjust the size of the graph to the display. Values are: ♦ 0 — False, do not adjust the size of the graph ♦ 1 — True, adjust the size of the graph *Value* can be a quoted DataWindow painter expression

Usage

In the painter Set the value using the Object property sheet, General tab, Size To Display option.

Examples

```
setting = dw_1.Object.graph_1.SizeToDisplay

dw_1.Object.graph_1.SizeToDisplay = 0

setting = dw_1.Describe("graph_1.SizeToDisplay")

dw_1.Modify("graph_1.SizeToDisplay=0")
```

SlideLeft

Description

Whether the object moves to the left when other objects to the left leave empty space available.

Applies to

Bitmap, Button, Column, Computed Field, Graph, GroupBox, Line, Oval, Rectangle, Report, RoundRectangle, TableBlob, and Text objects

Syntax

Dot notation:

dw_control.Object.*objectname*.SlideLeft

Describe and Modify argument:

"*objectname*.SlideLeft { = ' *value* ' }"

Parameter	Description
objectname	The name of the object for which you want to get or set its Slide setting

Parameter	Description
value	(*exp*) Whether the object slides left when there is empty space to its left. Values are: ♦ Yes — The object will slide left into available space ♦ No — The object will remain in position *Value* can be a quoted DataWindow painter expression

Usage

In the painter Set the value using the Object property sheet, Position tab, Slide group, Left checkbox.

Examples

```
setting = dw_1.Object.graph_1.SlideLeft

dw_1.Object.emp_lname.SlideLeft = "yes"

setting = dw_1.Describe("graph_1.SlideLeft")

dw_1.Modify("emp_lname.SlideLeft=yes")
```

SlideUp

Description

Whether the object moves up when other objects above it leave empty space available.

Applies to

Bitmap, Button, Column, Computed Field, Graph, GroupBox, Line, Oval, Rectangle, Report, RoundRectangle, TableBlob, and Text objects

Syntax

Dot notation:

dw_control.Object.*objectname*.SlideUp

Describe and Modify argument:

"*objectname*.SlideUp { = ' *value* ' }"

Parameter	Description
objectname	The name of the object for which you want to get or set its Slide setting

Parameter	Description
value	(*exp*) How the object slides up when there is empty space to its left. Values are: ♦ AllAbove — Slide the object up if all the objects in the row above it are empty ♦ DirectlyAbove — Slide the column or object up if the objects directly above it are empty ♦ No — The object will not slide up *Value* can be a quoted DataWindow painter expression

Usage

In the painter Set the value using Object property sheet, Position tab, Slide group, Up checkbox.

Examples

```
setting = dw_1.Object.graph_1.SlideUp

dw_1.Object.emp_lname.SlideUp = "no"

setting = dw_1.Describe("graph_1.SlideUp")

dw_1.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 functions.

Applies to

Table keywords in DataWindow syntax

Syntax

DataWindow syntax for Create function:

Table (... Sort = *stringexpression* ...)

Parameter	Description
stringexpression	A string whose value is valid sort criteria. See the SetSort function for the format for sort criteria. If the criteria string is NULL, PowerBuilder prompts for a sort specification when it displays the DataWindow

Spacing

Description

The gap between categories in a graph.

Applies to

Graph objects

Syntax

Dot notation:

dw_control.Object.*graphname*.Spacing

Describe and Modify argument:

"*graphname*.Spacing { = ' *integer* ' }"

Parameter	Description
graphname	The name of the graph object in the DataWindow for which you want to get or set the spacing
integer	(*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 painter expression

Usage

In the painter Set the value using Object property sheet, Graph tab, Spacing (% of width) option.

Examples

```
setting = dw_1.Object.graph_1.Spacing

dw_1.Object.graph_1.Spacing = 120

setting = dw_1.Describe("graph_1.Spacing")

dw_1.Modify("graph_1.Spacing=120")
```

Sparse

Description

The names of repeating columns that will be suppressed in the DataWindow.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.Sparse

Describe and Modify argument:

"DataWindow.Sparse { = ' *list* ' }"

Parameter	Description
list	(*exp*) A tab-separated list of column names to be suppressed. *List* can be a quoted DataWindow painter expression

Create function (include at the end of the DataWindow syntax):

Sparse (names = "*col1*~t*col2*~t*col3* ...")

Usage

In the painter Set the value using RowsSuppress Repeating Values.

Examples

```
setting = dw_1.Object.DataWindow.Sparse

dw_1.Object.DataWindow.Sparse = 'col1~tcol2'

setting = dw_1.Describe("DataWindow.Sparse")

dw_1.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

Dot notation:

dw_control.Object.DataWindow.Storage

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

```
setting = dw_1.Object.DataWindow.Storage

setting = dw_1.Describe("DataWindow.Storage")
IF Long(setting) > 50000 THEN RETURN 1
```

Summary.*property*

See Bandname.property.

SuppressEventProcessing

Description

Whether the ButtonClicked or ButtonClicking event is fired for this particular button.

Applies to

Button objects

Syntax

Dot notation:

dw_control.Object.*buttonname*.SuppressEventProcessing

Describe and Modify argument:

"*buttonname*.SuppressEventProcessing { = ' *value* ' }"

Parameter	Description
buttonname	The name of the button object for which you want to suppress event processing
value	Whether event processing is to occur. Values are: Yes — The event should be fired No — The event should not be fired

Usage

In the painter Set the value using the Button object property sheet, General tab.

Examples

```
dw_1.Object.b_name.SuppressEventProcessing = "Yes"

setting = dw_1.Describe("b_name.SuppressEventProcessing")

dw_1.Modify("b_name.SuppressEventProcessing ='No'")
```

Syntax

Description

The complete syntax for the DataWindow.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.Syntax

Describe argument:

"DataWindow.Syntax"

Examples

```
setting = dw_1.Object.DataWindow.Syntax
setting = dw_1.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

Dot notation:

dw_control.Object.DataWindow.Syntax.Data

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 function or changing the size of the DataWindow grid automatically sets Syntax.Modified to Yes.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.Syntax.Modified

Describe and Modify argument:

"DataWindow.Syntax.Modified { = *value* }"

Parameter	Description
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

```
setting = dw_1.Object.DataWindow.Syntax.Modified

dw_1.Object.DataWindow.Syntax.Modified = "No"

setting = dw_1.Describe("DataWindow.Syntax.Modified")

dw_1.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 function.

Syntax

Does not apply.

Usage

Use this property to redefine a DataWindow result set. You can add a column, change the data type of a column, or make other changes to in the table section of your DataWindow involving properties that are not accessible through Modify calls or dot notation.

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 functions 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 TableBlobs)

Description

The name of the database table that contains the blob.

Applies to

TableBlob objects

Syntax

Dot notation:

dw_control.Object.*tblobname*.Table

Describe and Modify argument:

"*tblobname*.Table { = ' *tablename* ' }"

Parameter	Description
tblobname	The name of the TableBlob object in the DataWindow
tablename	(*exp*) A string specifying the name of the table that contains the blob data. *Tablename* can be a quoted DataWindow painter expression

Usage

In the painter Set the value using the Object property sheet, Definition tab, Table option.

Examples

```
setting = dw_1.Object.blob_1.Table
dw_1.Object.blob_1.Table = "emp_pictures"
setting = dw_1.Describe("blob_1.Table")
dw_1.Modify("blob_1.Table='emp_pictures'")
```

Table.*property*

Description

Properties for the DataWindow's DBMS connection.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.Table.*property*

Describe and Modify argument:

"DataWindow.Table.*property* { = *value* }"

Parameter	Description
property	A property for the DataWindow's DBMS connection. Properties and appropriate values are listed in the table below
value	The value to be assigned to the property

Property for Table	Value
CrosstabData	A string containing a tab-separated list of the expressions used to calculate the values of columns in a crosstab DataWindow
Data.Storage	A string indicating whether table data is to be kept in memory or offloaded to disk. Values are: ♦ Memory (Default)— Table data is to be kept in memory ♦ Disk — Table data is to be offloaded to disk. Painter: Rows>Retrieve>Rows to Disk

Property for Table	Value
Filter	(*exp*) 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 functions The filter string can be a quoted DataWindow painter expression Painter: Rows>Filter
GridColumns	(Read-only) The grid columns of a DataWindow
Procedure	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

Property for Table	Value
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 function: ♦ 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 SetTransObject function has been executed for the DataWindow object 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, the SetSort and Sort functions The value for Sort is quoted and can be a DataWindow painter expression. Painter: Rows>Sort
SQLSelect	The most recently executed SELECT statement. Setting this has no effect. See Select in this table

Property for Table	Value
UpdateKey InPlace	Whether the key column can be updated in place or whether the row has to be deleted and reinserted. This value determines the syntax PowerBuilder will generate 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** When there are multiple rows in a DataWindow object and the user switches keys or rows, updating in place may fail due to DBMS duplicate restrictions. Painter: RowsäUpdate Properties, Key Modification
UpdateTable	A string specifying the name of the database table used to build the Update syntax Painter: RowsäUpdate 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 and 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 INFO For more about the effects of this setting, see the discussion of the Specify Update Characteristics dialog box in the *PowerBuilder User's Guide* Painter: RowsäUpdate Properties, Where Clause for Update/Delete

Examples

```
setting = dw_1.Object.DataWindow.Table.Sort

dw_1.Object.DataWindow.Table.Data.Storage &
= "disk"
```

```
dw_1.Object.DataWindow.Table.Filter = "salary>50000"

setting = dw_1.Describe("DataWindow.Table.Sort")

dw_1.Modify("DataWindow.Table.Filter='salary>50000'")

dw_1.Modify &
(" DataWindow.Table.Procedure= &
'1 Execute MyOwner MyProcName;1 &
@NameOfProcArg=:NameOfDWArg,
@NameOfProcArg=:NameOfDWArg...' ")

sqlvar = 'SELECT ... WHERE ...'
dw_1.Modify("DataWindow.Table.Select='" + sqlvar + "'")
```

TabSequence

Description

The number assigned to the specified object in the DataWindow's tab order. You can also call the SetTabOrder function to change TabSequence.

Applies to

Column objects

Syntax

Dot notation:

dw_control.Object.*columnname*.TabSequence

Describe and Modify argument:

"*columnname*.TabSequence { = *number* }"

Parameter	Description
columnname	The name of the column whose tab order you want to get or set
number	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

Usage

In the painter Set the value using Design>Tab Order.

Examples

```
setting = dw_1.Object.emp_name.TabSequence

dw_1.Object.emp_name.TabSequence = 10

setting = dw_1.Describe("emp_name.TabSequence")

dw_1.Modify("emp_name.TabSequence = 10")
```

Tag

Description

The tag value of the specified object. The tag value can be any text you see fit to use in your application.

Applies to

Bitmap, Button, Column, Computed Field, Graph, GroupBox, Oval, Rectangle, Report, RoundRectangle, TableBlob, and Text objects

Syntax

Dot notation:

dw_control.Object.*objectname*.Tag

Describe and Modify argument:

"*objectname*.Tag { = ' *string* ' }"

Parameter	Description
objectname	The name of an object in the DataWindow
string	(*exp*) A string specifying the tag for *objectname*. *String* is quoted and can be a DataWindow painter expression

Usage

In the painter Set the value using Object property sheet, General tab, Tag option.

Examples

```
setting = dw_1.Object.blob_1.Tag

dw_1.Object.graph_1.Tag = 'Graph of results'

setting = dw_1.Describe("blob_1.Tag")

dw_1.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 objects

Syntax

Dot notation:

dw_control.Object.*oleobjectname*.Target

Describe and Modify argument:

"*oleobjectname*.Target { = ' *columnlist* ' }"

Parameter	Description
oleobjectname	The name of the OLE container object for which you want to get or set the data to be transferred
columnlist	(*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 painter expression

Usage

GroupBy and Range also affect the data that is transferred to the OLE object.

In the painter Set the value using Object property sheet, Data tab, Target Data option.

Examples

```
setting = dw_1.Object.ole_1.Target

dw_1.Object.ole_1.Target = 'lname, Len(companyname)'

setting = dw_1.Describe("ole_1.Target")

dw_1.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 objects

Syntax

Dot notation:

dw_control.Object.*tblobname*.Template

Describe and Modify argument:

"*tblobname*.Template { = ' *string* ' }"

Parameter	Description
tblobname	The name of a TableBlob object in the DataWindow
string	(*exp*) A string whose value is the filename of an application to be the OLE template. *String* is quoted and can be a DataWindow painter expression

Usage

In the painter Set the value using the Object property sheet, Definition tab, File Template option.

Examples

```
setting = dw_1.Object.blob_1.Template
dw_1.Object.blob_1.Template='Excel.xls'
setting = dw_1.Describe("blob_1.Template")
dw_1.Modify("blob_1.Template='Excel.xls'")
```

Text

Description

The text of the specified Text object.

Applies to

Button, GroupBox, and Text objects

Syntax

Dot notation:

dw_control.Object.*textname*.Text

Describe and Modify argument:

"*textname*.Text { = ' *string* ' }"

Parameter	Description
textname	The name of a Text object in the DataWindow
string	(*exp*) A string specifying the text for *textname*. To specify an accelerator key in the text, include an ampersand before the desired letter. The letter will display underlined. *String* is quoted and can be a DataWindow painter expression

Usage

In the painter Set the value using the Object property sheet, General tab, Text option.

Examples

```
setting = dw_1.Object.text_1.Text
dw_1.Object.text_1.Text = "Employee &Name"
setting = dw_1.Describe("text_1.Text")
dw_1.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.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.Timer_Interval

Describe and Modify argument:

"DataWindow.Timer_Interval { = *number* }"

SyntaxFromSQL:

DataWindow (Timer_Interval = *number*)

Parameter	Description
number	An integer specifying the interval between timer events in milliseconds. The default is 0 (which specifies 60,000 or one minute)

Usage

In the painter Set the value using DataWindow Object property sheet, General tab, Timer Interval option.

Examples

```
setting = dw_1.Object.DataWindow.Timer_Interval
dw_1.Object.DataWindow.Timer_Interval = 10000
setting = dw_1.Describe("DataWindow.Timer_Interval")
dw_1.Modify("DataWindow.Timer_Interval=10000")
```

Title

Description

The title of the graph.

Applies to

Graph objects

Syntax

Dot notation:

dw_control.Object.*graphname*.Title

Describe and Modify argument:

"*graphname*.Title { = ' *titlestring* ' }"

Parameter	Description
graphname	The name of the Graph object in the DataWindow object for which you want to get or set the title
titlestring	A string specifying the graph's title

Usage

In the painter Set the value using the Object property sheet, Graph tab, Title option.

The Title property is used in the graph title only if the display expression for the Title text object includes the Title property in its expression. You can build an expression for the graph title using the Title property plus other text, functions, and operators.

Examples

```
setting = dw_1.Object.gr_1.Title

dw_1.Object.gr_1.Title = 'Sales Graph'

setting = dw_1.Describe("gr_1.Title")

dw_1.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 only applies to reports in a composite DataWindow. Setting Trail_Footer to No forces objects following the report onto a new page.

Applies to

Report objects

Syntax

Dot notation:

dw_control.Object.*reportname*.Trail_Footer

Describe and Modify argument:

"*reportname*.Trail_Footer { = *value* }"

Parameter	Description
reportname	The name of the report object for which you want to get or set Trail_Footer
value	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

Examples

```
setting = dw_1.Object.rpt_1.Trail_Footer

dw_1.Object.rpt_1.Trail_Footer = "Yes"

setting = dw_1.Describe("rpt_1.Trail_Footer")

dw_1.Modify("rpt_1.Trail_Footer = Yes")
```

Trailer.#.*property*

See Bandname.property.

Type

Description

The type of the object (for Describe) or the type of presentation style (for SyntaxFromSQL).

Syntax

Dot notation:

dw_control.Object.*objectname*.Type

Describe argument:

"*objectname*.Type"

Parameter	Description
objectname	The name of the object for which you want the type. Valid values are: ♦ datawindow ♦ bitmap ♦ button ♦ column ♦ compute (for Computed Field) ♦ graph ♦ groupbox ♦ line ♦ ole ♦ ellipse (for Oval) ♦ rectangle ♦ report ♦ roundrectangle ♦ tableblob ♦ text

SyntaxFromSQL:

Style (Type = *value*)

Parameter	Description
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 ♦ Ole ♦ RichText

Examples

```
setting = dw_1.Object.emp_name.Type

setting = dw_1.Describe("emp_name.Type")

SQLCA.SyntaxFromSQL(sqlstring, &
'Style(... Type=grid ...)', errstring)
```

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

Dot notation:

dw_control.Object.DataWindow.Units

Describe argument:

"DataWindow.Units"

SyntaxFromSQL:

DataWindow (Units = *value*)

Parameter	Description
value	The type of units for measurements in the DataWindow. Values are: 0 — PowerBuilder units 1 — Display pixels 2 — 1/1000 of a logical inch 3 — 1/1000 of a logical centimeter

Usage

PowerBuilder units and display pixels are adjusted for printing.

In the painter Set the value using the DataWindow Object property sheet, General tab, Units option.

Examples

```
setting = dw_1.Object.DataWindow.Units

setting = dw_1.Describe("DataWindow.Units")
```

Update

Description

Whether the specified column is updatable. Each updatable column is included in the SQL statement that the Update function sends to the database. All updatable columns should be in the same database table.

Applies to

Column objects

Syntax

Dot notation:

dw_control.Object.*columnname*.Update

Describe and Modify argument:

"*columnname*.Update { = *value* }"

Parameter	Description
columnname	The column for which you want to get or set its updatable status
value	Whether the column is updatable. Values are: Yes — Include the column in the SQL statement for updating the database No — Do not include the column in the SQL statement

Usage

In the painter Set the value using:

- Rows>Update Properties, Updateable Columns option
- StyleBar (useful for multiple objects)
- Object property sheet, Expressions tab (use a conditional expression)

Examples

```
setting = dw_1.Object.emp_name.Update

dw_1.Object.emp_name.Update = "No"

setting = dw_1.Describe("emp_name.Update")

dw_1.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 function. To find out the current validation expression, use the GetValidate function.

Applies to

Column objects

Syntax

Dot notation:

dw_control.Object.*columnname*.Validation

Describe and Modify argument:

"*columnname*.Validation { = ' *validationstring* ' }"

Parameter	Description
columnname	The column for which you want to get or set the validation rule
validationstring	(*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 painter expression

Usage

In the painter Set the value using the Object property sheet, Validation tab, Validation Expression option.

Use operators, functions and columns to build an expression. Use Verify to test it.

Examples

```
setting = dw_1.Object.emp_status.Validation
setting = dw_1.Describe("emp_status.Validation")
```

ValidationMsg

Description

The message that PowerBuilder displays instead of the default message when an ItemError event occurs in the column.

Applies to

Column objects

Syntax

Dot notation:

dw_control.Object.*columnname*.ValidationMsg

Describe and Modify argument:

"*columnname*.ValidationMsg { = ' *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 painter expression

Usage

In the painter Set the value using the Object property sheet, Validation tab, Error Message Expression option.

Examples

```
setting = dw_1.Object.emp_salary.ValidationMsg

dw_1.Object.emp_salary.ValidationMsg = &
"Salary must be between 10,000 and 100,000"

setting = dw_1.Describe("emp_salary.ValidationMsg")

dw_1.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 objects

Syntax

Dot notation:

dw_control.Object.*columnname*.Values

Describe and Modify argument:

"*columnname*.Values { = ' *string* ' }"

Parameter	Description
columnname	The column for which you want to specify the contents of the code table

Parameter	Description
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~tactualval/displayval~tactualval/* ..." For example: `"red~t1/white~t2"` *String* is quoted and can be a DataWindow painter expression

Usage

In the painter Set the value using the Object property sheet, Edit tab, Display Value and Data Value options (available for edit style DropDownListBox and for Edit and EditMask when a code table is enabled).

When Style is Edit, check Use Code Table.

When Style is EditMask, check Spin Control and Code Table.

Examples

```
setting = dw_1.Object.emp_status.Values

dw_1.Object.emp_status.Values = &
"Active~tA/Part Time~tP/Terminated~tT"

setting = dw_1.Describe("emp_status.Values")

dw_1.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, PowerBuilder 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

SyntaxFromSQL:

Style (Vertical_Size = *value*)

Parameter	Description
value	An integer specifying the height of the columns in the detail area of the DataWindow object area in the units specified for the DataWindow

Examples

```
SQLCA.SyntaxFromSQL(sqlstring, &
'Style(... Vertical_Size=1225...)', errstring)
```

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

SyntaxFromSQL:

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

```
SQLCA.SyntaxFromSQL(sqlstring, &
'Style(... Vertical_Spread=25...)', errstring)
```

VerticalScrollMaximum

Description

The maximum height of the scroll box of the DataWindow's horizontal scrollbar. This value is set by PowerBuilder based on the content of the DataWindow. Use VerticalScrollMaximum with VerticalScrollPosition to synchronize vertical scrolling in multiple DataWindow objects.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.VerticalScrollMaximum

Describe argument:

"DataWindow.VerticalScrollMaximum"

Examples

```
setting = dw_1.Object.DataWindow.VerticalScrollMaximum

setting = &
dw_1.Describe("DataWindow.VerticalScrollMaximum")
```

VerticalScrollPosition

Description

The position of the scroll box in the vertical scrollbar. Use VerticalScrollMaximum with VerticalScrollPosition to synchronize vertical scrolling in multiple DataWindow objects.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.VerticalScrollPosition

Describe and Modify argument:

"DataWindow.VerticalScrollPosition { = *scrollvalue* }"

Parameter	Description
scrollvalue	An integer specifying the position of the scroll box in the vertical scrollbar of the DataWindow

Examples

```
spos1 = dw_1.Object.DataWindow.VerticalScrollPosition

spos1 = &
dw_1.Describe("DataWindow.VerticalScrollPosition")
```

```
smax = &
dw_1.Describe("DataWindow.VerticalScrollMaximum")
sscroll = String(Integer(smax)/2)
modstring = &
"DataWindow.VerticalScrollPosition=" + sscroll
dw_1.Modify(modstring)
```

Visible

Description

Whether the specified object in the DataWindow is visible.

Applies to

Bitmap, Button, Column, Computed Field, Graph, GroupBox, Line, OLE, Oval, Rectangle, Report, RoundRectangle, TableBlob, and Text objects

Syntax

Dot notation:

dw_control.Object.*objectname*.Visible

Describe and Modify argument:

"*objectname*.Visible { = ' *value* ' }"

Parameter	Description
objectname	The name of the object for which you want to get or set the Visible property
value	(*exp*) Whether the specified object is visible. Values are: 0 — False; the object is not visible 1 — True; the object is visible *Value* can be a quoted DataWindow painter expression

Usage

In the painter Set the value using Object property sheet, Expressions tab (use a conditional expression).

Examples

```
setting = dw_1.Object.emp_status.Visible

dw_1.Object.emp_status.Visible = 0

dw_1.Object.emp_stat.Visible="0~tIf(emp_class=1,0,1)"

setting = dw_1.Describe("emp_status.Visible")

dw_1.Modify("emp_status.Visible=0")

dw_1.Modify("emp_stat.Visible='0~tIf(emp_class=1,0,1)'")
```

VTextAlign

Description

The way text in a button is vertically aligned.

Applies to

Button objects

Syntax

Dot notation:

dw_control.Object.*buttonname*.VTextAlign

Describe and Modify argument:

"*buttonname*.VTextAlign { = ' *value* ' }"

Parameter	Description
buttonname	The name of the button for which you want to align text
value	An integer indicating how the button text is horizontally aligned. Values are: 0 — Center 1 — Top 2 — Bottom 3 — Multiline

Usage

In the painter Set the value using the Button object property sheet, General tab.

Examples

```
dw_1.Object.b_name.VTextAlign = "0"

setting = dw_1.Describe("b_name.VTextAlign")

dw_1.Modify("b_name.VTextAlign ='0'")
```

Width

Description

The width of the specified object.

Applies to

Bitmap, Button, Column, Computed Field, Graph, GroupBox, OLE, Oval, Rectangle, Report, RoundRectangle, TableBlob, and Text objects

Syntax

Dot notation:

dw_control.Object.*objectname*.Width

Describe and Modify argument:

"*objectname*.Width { = ' *value* ' }"

Parameter	Description
objectname	The name of the object for which you want to get or set the width
value	(*exp*) The width of the *objectname* in the units specified for the DataWindow. *Value* can be a quoted DataWindow painter expression

Usage

In the painter Set the value using:

- Object property sheet, Position tab
- Object property sheet, Expressions tab (use a conditional expression)

Examples

```
setting = dw_1.Object.emp_name.Width

dw_1.Object.emp_name.Width = 250

setting = dw_1.Describe("emp_name.Width")

dw_1.Modify("emp_name.Width=250")
```

Width.Autosize

Description

(RichText presentation style only) Whether the column or computed field input field adjusts its width according to the data it contains.

The Width.Autosize and Multiline properties can be set together so that the input field can display multiple lines.

Applies to

Column and Computed Field objects in the RichText presentation style

Syntax

Dot notation:

dw_control.Object.*objectname*.Width.Autosize

Describe and Modify argument:

"*objectname*.Width.Autosize { = ' *value* ' }"

Parameter	Description
objectname	The name of the column or computed field for which you want to get or set the Autosize setting

Parameter	Description
value	(*exp*) Whether the width of the input field adjusts according to the data it contains. Values are: ♦ Yes — The width adjusts according to the data ♦ No — The width is fixed and is set to the value of the Width property

Usage

In the painter Set the value using the Input Field Object property sheet, Input Field tab, Fixed Size option.

To display the property sheet, click to select the input field. Then right-click and select Properties from the popup menu.

Examples

```
setting = dw_1.Object.emp_name.Width.Autosize

dw_1.Object.emp_name.Width.Autosize = "yes"

setting = dw_1.Describe("emp_name.Width.Autosize")

dw_1.Modify("emp_name.Width.Autosize=yes")
```

X

Description

The distance of the specified object from the left edge of the DataWindow object.

Applies to

Bitmap, Button, Column, Computed Field, Graph, GroupBox, OLE, Oval, Rectangle, Report, RoundRectangle, TableBlob, and Text objects

Syntax

Dot notation:

dw_control.Object.*objectname*.X

Describe and Modify argument:

"*objectname*.X { = ' *value* ' }"

Parameter	Description
objectname	The name of the object for which you want to get or set the x coordinate
value	(*exp*) An integer specifying the x coordinate of the object in the unit of measure specified for the DataWindow object. *Value* can be a quoted DataWindow painter expression

Usage

In the painter Set the value using:

- Object property sheet, Position tab
- Object property sheet, Expressions tab (use a conditional expression)

Examples

```
setting = dw_1.Object.emp_name.X

dw_1.Object.emp_name.X = 10

setting = dw_1.Describe("emp_name.X")

dw_1.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 objects

Syntax

Dot notation:

dw_control.Object.*objectname*.X1

dw_control.Object.*objectname*.X2

Describe and Modify argument:

"*objectname*.X1 { = ' *value* ' }"

"*objectname*.X2 { = ' *value* ' }"

Parameter	Description
objectname	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 painter expression

Usage

In the painter Set the value using:

- Object property sheet, Position tab
- Object property sheet, Expressions tab (use a conditional expression)

Examples

```
setting = dw_1.Object.line_1.X1
```

```
dw_1.Object.line_1.X1 = 10
dw_1.Object.line_1.X2 = 1000

setting = dw_1.Describe("line_1.X1")

dw_1.Modify("line_1.X1=10")
dw_1.Modify("line_1.X2=1000")
```

Y

Description

The distance of the specified object from the top of the object's band.

Applies to

Bitmap, Button, Column, Computed Field, Graph, GroupBox, OLE, Oval, Rectangle, Report, RoundRectangle, TableBlob, and Text objects

Syntax

Dot notation:

dw_control.Object.*objectname*.Y

Describe and Modify argument:

"*objectname*.Y { = ' *value* ' }"

Parameter	Description
objectname	The name of the object for which you want to get or set the y coordinate
value	(*exp*) An integer specifying the y coordinate of the object in the unit of measure specified for the DataWindow object. *Value* can be a quoted DataWindow painter expression

Usage

In the painter Set the value using:

- Object property sheet, Position tab
- Object property sheet, Expressions tab (use a conditional expression)

Examples

```
setting = dw_1.Object.emp_name.Y

dw_1.Object.emp_name.Y = 100

setting = dw_1.Describe("emp_name.Y")

dw_1.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 objects

Syntax

Dot notation:

dw_control.Object.*objectname*.Y1

dw_control.Object.*objectname*.Y2

Describe and Modify argument:

"*objectname*.Y1 { = ' *value* ' }"

"*objectname*.Y2 { = ' *value* ' }"

Parameter	Description
objectname	The name of the line for which you want to get or set one of the y coordinates
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 painter expression

Usage

In the painter Set the value using:

- Object property sheet, Position tab
- Object property sheet, Expressions tab (use a conditional expression)

Examples

```
setting = dw_1.Object.line_1.Y1

dw_1.Object.line_1.Y1 = 50
dw_1.Object.line_1.Y2 = 50

setting = dw_1.Describe("line_1.Y1")

dw_1.Modify("line_1.Y1=50")
dw_1.Modify("line_1.Y2=50")
```

Zoom

Description

The scaling percentage of the DataWindow object.

Applies to

DataWindows

Syntax

Dot notation:

dw_control.Object.DataWindow.Zoom

Describe and Modify argument:

"DataWindow.Zoom { = *value* }"

Parameter	Description
value	An integer specifying the scaling percentage of the DataWindow object. The default is 100%

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 during execution.

Examples

```
setting = dw_1.Object.DataWindow.Zoom

dw_1.Object.DataWindow.Zoom = 50

setting = dw_1.Describe("DataWindow.Zoom")

dw_1.Modify("DataWindow.Zoom=50")
```

Index

A

B

C

D

E

F

G

H

I

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z