# PHPExcel Function Reference Developer Documentation

| | |
|---|---|
| **Author:** | Mark Baker |
| **Version:** | 1.7.3 |
| **Date:** | 20 January 2011 |

# 1. Contents

# 2. Frequently asked questions

The up-to-date F.A.Q. page for PHPExcel can be found on http://www.codeplex.com/PHPExcel/Wiki/View.aspx?title=FAQ&referringTitle=Requirements.

### Formulas don’t seem to be calculated in Excel2003 using compatibility pack?

This is normal behaviour of the compatibility pack, Excel2007 displays this correctly. Use PHPExcel_Writer_Excel5 if you really need calculated values, or force recalculation in Excel2003.

# 3. Function Reference

## 3.1. Function that are not Supported in Excel5

Not all functions are supported by the Excel 5 Writer. Use of these functions within your workbooks will result in an error when trying to write to Excel5.
The following is the list of those functions that are implemented within PHPExcel, but that cannot currently be written to Excel 5.

#### Date and Time

| | |
|---|---|
| EDATE | Not a standard function within Excel 5, but an add-in from the Analysis ToolPak. |
| EOMONTH | Not a standard function within Excel 5, but an add-in from the Analysis ToolPak. |

## 3.2. Date and Time Values

### 3.2.1. Excel functions that return a Date and Time value

Any of the Date and Time functions that return a date value in Excel can return either an Excel timestamp or a PHP timestamp or date object.

It is possible for scripts to change the data type used for returning date values by calling the PHPExcel_Calculation_Functions::setReturnDateType() method:

```
PHPExcel_Calculation_Functions::setReturnDateType($returnDateType);
```

where the following constants can be used for $returnDateType

```
PHPExcel_Calculation_Functions::RETURNDATE_PHP_NUMERIC
PHPExcel_Calculation_Functions::RETURNDATE_PHP_OBJECT
PHPExcel_Calculation_Functions::RETURNDATE_EXCEL
```

The method will return a Boolean True on success, False on failure (e.g. if an invalid value is passed in for the return date type).

The PHPExcel_Calculation_Functions::getReturnDateType() method can be used to determine the current value of this setting:

```
$returnDateType = PHPExcel_Calculation_Functions::getReturnDateType();
```

The default is RETURNDATE_PHP_NUMERIC.

#### PHP Timestamps

If RETURNDATE_PHP_NUMERIC is set for the Return Date Type, then any date value returned to the calling script by any access to the Date and Time functions in Excel will be an integer value that represents the number of seconds from the PHP base date. The PHP base date (0) is 00:00 GMT on 1st January 1970. This value can be positive or negative: so a value of -3600 would be 23:00 hrs on 31st December 1969; while a value of +3600 would be 01:00 hrs on 1st January 1970. This gives PHP a date range of between 14th December 1901 and 19th January 2038.

#### PHP date/Time Objects

If the Return Date Type is set for RETURNDATE_PHP_NUMERIC, then any date value returned to the calling script by any access to the Date and Time functions in Excel will be a PHP date/time object[1].

[1] See http://www.php.net/manual/en/ref.datetime.php for details of PHP date/time objects.

### Excel Timestamps

If RETURNDATE_EXCEL is set for the Return Date Type, then the returned date value by any access to the Date and Time functions in Excel will be a floating point value that represents a number of days from the Excel base date. The Excel base date is determined by which calendar Excel uses: the Windows 1900 or the Mac 1904 calendar. 1st January 1900 is the base date for the Windows 1900 calendar while 1st January 1904 is the base date for the Mac 1904 calendar.
It is possible for scripts to change the calendar used for calculating Excel date values by calling the PHPExcel_Shared_Date::setExcelCalendar() method:

```
PHPExcel_Shared_Date::setExcelCalendar($baseDate);
```

where the following constants can be used for $baseDate

```
PHPExcel_Shared_Date::CALENDAR_WINDOWS_1900
PHPExcel_Shared_Date::CALENDAR_MAC_1904
```

The method will return a Boolean True on success, False on failure (e.g. if an invalid value is passed in).

The PHPExcel_Shared_Date::getExcelCalendar() method can be used to determine the current value of this setting:

```
$baseDate = PHPExcel_Shared_Date::getExcelCalendar();
```

The default is CALENDAR_WINDOWS_1900[2].

### Functions that return a Date/Time Value

DATE
DATEVALUE
EDATE
EOMONTH

## 3.2.2. Excel functions that accept Date and Time values as parameters

Date values passed in as parameters to a function can be an Excel timestamp or a PHP timestamp; or date object; or a string containing a date value (e.g. '1-Jan-2009'). PHPExcel will attempt to identify their type based on the PHP datatype:

- An integer numeric value will be treated as a PHP date timestamp
- A real (floating point) numeric value will be treated as an Excel date timestamp.
- Any PHP date object will be treated as a date object.
- Any string value (even one containing straight numeric data) will be converted to a date/time object for validation as a date value based on the server locale settings, so passing through an ambiguous value of '07/08/2008' will be treated as 7th August 2008 if your server settings are UK, but as 8th July 2008 if your server settings are US. However, if you pass through a value such

[2] When reading from an Excel file generated using the Windows 1900 or the Mac 1904 calendar, PHPExcel will set this flag automatically to the correct value for that workbook. However, the setting is applied globally. Note that when you are reading multiple workbooks, some of which use Windows 1900 and others using Mac 1904, then the calendar setting that is applied will be that of the latest file to be read. This may lead to errors in calculations.

When writing an Excel file, the calendar in that file will be set to the current value of the calendar flag in PHPExcel_Shared_Date.

as ‘31/12/2008’ that would be considered an error by a US-based server, but which is not ambiguous, then PHPExcel will attempt to correct this to 31st December 2008.
If the content of the string doesn’t match any of the formats recognised by the php date/time object implementation of strtotime() (which can handle a wider range of formats than the normal strtotime() function), then the function will return a ‘#VALUE’ error. However, Excel recommends that you should always use date timestamps for your date functions, and the recommendation for PHPExcel is the same: avoid strings because the result is not predictable.

The same principle applies when data is being written to Excel. Cells containing date actual values (rather than Excel functions that return a date value) are always written as Excel dates, converting where necessary. If a cell formatted as a date contains an integer or date/time object value, then it is converted to an Excel value for writing: if a cell formatted as a date contains a real value, then no conversion is required. Note that string values are written as strings rather than converted to Excel date timestamp values.

#### Functions that expect a Date/Time Value

DATEDIF
DAY
DAYS360
EDATE
EOMONTH
HOUR
MINUTE
MONTH
NETWORKDAYS
SECOND
WEEKDAY
YEAR

### 3.2.3. Helper Methods

In addition to the setExcelCalendar() and getExcelCalendar() methods, a number of other methods are available in the PHPExcel_Shared_Date class that can help when working with dates:

#### PHPExcel_Shared_Date::ExcelToPHP($excelDate)

Converts a date/time from an Excel date timestamp to return a PHP date/time timestamp.

Note that this method does not trap for Excel dates that fall outside of the valid range for a PHP date timestamp.

#### PHPExcel_Shared_Date::ExcelToPHPObject($excelDate)

Converts a date from an Excel date/time timestamp to return a PHP date/time object.

#### PHPExcel_Shared_Date::PHPToExcel($PHPDate)

Converts a PHP date timestamp or a PHP date/times object to return an Excel date timestamp.

#### PHPExcel_Shared_Date::FormattedPHPToExcel($year, $month, $day, $hours=0, $minutes=0, $seconds=0)

Takes year, month and day values (and optional hour, minute and second values) and returns an Excel date timestamp value.

## *3.3. Cube Functions*

### 3.3.1. CUBEKPIMEMBER

Not yet implemented.

### 3.3.2. CUBEMEMBER

Not yet implemented.

### 3.3.3. CUBEMEMBERPROPERTY

Not yet implemented.

### 3.3.4. CUBERANKEDMEMBER

Not yet implemented.

### 3.3.5. CUBESET

Not yet implemented.

### 3.3.6. CUBESETCOUNT

Not yet implemented.

### 3.3.7. CUBEVALUE

Not yet implemented.

## 3.4. Database Functions

### 3.4.1. DAVERAGE
Not yet implemented.

### 3.4.2. DCOUNT
Not yet implemented.

### 3.4.3. DCOUNTA
Not yet implemented.

### 3.4.4. DGET
Not yet implemented.

### 3.4.5. DMAX
Not yet implemented.

### 3.4.6. DMIN
Not yet implemented.

### 3.4.7. DPRODUCT
Not yet implemented.

### 3.4.8. DSTDEV
Not yet implemented.

### 3.4.9. DSTDEVP
Not yet implemented.

### 3.4.10. DSUM
Not yet implemented.

### 3.4.11. DVAR
Not yet implemented.

### 3.4.12. DVARP
Not yet implemented.

## 3.5. Date and Time Functions

Excel provides a number of functions for the manipulation of dates and times, and calculations based on date/time values. it is worth spending some time reading the section titled “Function that are not Supported in Excel5” above on passing date parameters and returning date values to understand how PHPExcel reconciles the differences between dates and times in Excel and in PHP.

### 3.5.1. DATE

The DATE function returns an Excel timestamp or a PHP timestamp or date object representing the date that is referenced by the parameters.

#### Syntax

DATE(year, month, day)

#### Parameters

| | |
|---|---|
| year | **The year number.**<br>If this value is between 0 (zero) and 1899 inclusive (for the Windows 1900 calendar), or between 4 and 1903 inclusive (for the Mac 1904), then PHPExcel adds it to the Calendar base year, so a value of 108 will interpret the year as 2008 when using the Windows 1900 calendar, or 2012 when using the Mac 1904 calendar. |
| month | **The month number.**<br>If this value is greater than 12, the DATE function adds that number of months to the first month in the year specified. For example, DATE(2008,14,2) returns a value representing February 2, 2009.<br>If the value of **month** is less than 1, then that value will be adjusted by -1, and that will then be subtracted from the first month of the year specified. For example, DATE(2008,0,2) returns a value representing December 2, 2007; while DATE(2008,-1,2) returns a value representing November 2, 2007. |
| day | **The day number.**<br>If this value is greater than the number of days in the month (and year) specified, the DATE function adds that number of days to the first day in the month. For example, DATE(2008,1,35) returns a value representing February 4, 2008.<br>If the value of **day** is less than 1, then that value will be adjusted by -1, and that will then be subtracted from the first month of the year specified. For example, DATE(2008,3,0) returns a value representing February 29, 2008; while DATE(2008,3,-2) returns a value representing February 27, 2008. |

#### Return Value

| | |
|---|---|
| mixed | **A date/time stamp that corresponds to the given date.**<br>This could be a PHP timestamp value (integer), a PHP date/time object, or an Excel timestamp value (real), depending on the value of PHPExcel_Calculation_Functions::getReturnDateType(). |

#### Examples

```
$worksheet->setCellValue('A1', 'Year');
$worksheet->setCellValue('A2', 'Month');
$worksheet->setCellValue('A3', 'Day');

$worksheet->setCellValue('B1', 2008);
$worksheet->setCellValue('B2', 12);
$worksheet->setCellValue('B3', 31);

$worksheet->setCellValue('D1', '=DATE(B1,B2,B3)');

$retVal = $worksheet->getCell('D1')->getCalculatedValue();
//             $retVal = 1230681600
//      We're going to be calling the same cell calculation multiple times,
//              and expecting different return values, so disable calculation cacheing
PHPExcel_Calculation::getInstance()->setCalculationCacheEnabled(False);

$saveFormat = PHPExcel_Calculation_Functions::getReturnDateType();

PHPExcel_Calculation_Functions::setReturnDateType(PHPExcel_Calculation_Functions::RETURNDATE_E
XCEL);
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'DATE'),
                               array(2008, 12, 31)
                              );
//              $retVal = 39813.0

PHPExcel_Calculation_Functions::setReturnDateType(PHPExcel_Calculation_Functions::RETURNDATE_P
HP_NUMERIC);
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'DATE'),
                               array(2008, 12, 31)
                              );
//              $retVal = 1230681600

PHPExcel_Calculation_Functions::setReturnDateType($saveFormat);
```

#### Notes

There are no additional notes on this function

### 3.5.2. DATEDIF

The DATEDIF function computes the difference between two dates in a variety of different intervals, such number of years, months, or days.

#### Syntax

DATEDIF(date1, date2 [, unit])

#### Parameters

**date1** **First Date.**
An Excel date value, PHP date timestamp, PHP date object, or a date represented as a string.

**date2** **Second Date.**
An Excel date value, PHP date timestamp, PHP date object, or a date represented as a string.

**unit** **The interval type to use for the calculation**
This is a string, comprising one of the values listed below:

| Unit | Meaning | Description |
|---|---|---|
| m | Months | Complete calendar months between the dates. |
| d | Days | Number of days between the dates. |
| y | Years | Complete calendar years between the dates. |
| ym | Months Excluding Years | Complete calendar months between the dates as if they were of the same year. |
| yd | Days Excluding Years | Complete calendar days between the dates as if they were of the same year. |
| md | Days Excluding Years And Months | Complete calendar days between the dates as if they were of the same month and same year. |

The unit value is not case sensitive, and defaults to ‘d’.

### Return Value

| | |
|---|---|
| integer | **An integer value that reflects the difference between the two dates.** This could be the number of full days, months or years between the two dates, depending on the interval unit value passed into the function as the third parameter. |

### Examples

```
$worksheet->setCellValue('A1', 'Year');
$worksheet->setCellValue('A2', 'Month');
$worksheet->setCellValue('A3', 'Day');

$worksheet->setCellValue('B1', 2001);
$worksheet->setCellValue('C1', 2009);
$worksheet->setCellValue('B2', 7);
$worksheet->setCellValue('C2', 12);
$worksheet->setCellValue('B3', 1);
$worksheet->setCellValue('C3', 31);

$worksheet->setCellValue('D1', '=DATEDIF(DATE(B1,B2,B3),DATE(C1,C2,C3),"d")');
$worksheet->setCellValue('D2', '=DATEDIF(DATE(B1,B2,B3),DATE(C1,C2,C3),"m")');
$worksheet->setCellValue('D3', '=DATEDIF(DATE(B1,B2,B3),DATE(C1,C2,C3),"y")');
$worksheet->setCellValue('D4', '=DATEDIF(DATE(B1,B2,B3),DATE(C1,C2,C3),"ym")');
$worksheet->setCellValue('D5', '=DATEDIF(DATE(B1,B2,B3),DATE(C1,C2,C3),"yd")');
$worksheet->setCellValue('D6', '=DATEDIF(DATE(B1,B2,B3),DATE(C1,C2,C3),"md")');

$retVal = $worksheet->getCell('D1')->getCalculatedValue();
//             $retVal = 3105
$retVal = $worksheet->getCell('D2')->getCalculatedValue();
//             $retVal = 101
$retVal = $worksheet->getCell('D3')->getCalculatedValue();
//             $retVal = 8
$retVal = $worksheet->getCell('D4')->getCalculatedValue();
//             $retVal = 5
$retVal = $worksheet->getCell('D5')->getCalculatedValue();
//             $retVal = 183
$retVal = $worksheet->getCell('D6')->getCalculatedValue();
//             $retVal = 30
```

```
$date1 = 1193317015          //      PHP timestamp for 25-Oct-2007
$date2 = 1449579415          //      PHP timestamp for 8-Dec-2015

$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'DATEDIF'),
                               array($date1, $date2, 'd')
                              );
//             $retVal = 2966
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'DATEDIF'),
                               array($date1, $date2, 'm')
                              );
//             $retVal = 97
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'DATEDIF'),
                               array($date1, $date2, 'y')
                              );
//             $retVal = 8
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'DATEDIF'),
                               array($date1, $date2, 'ym')
                              );
//             $retVal = 1
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'DATEDIF'),
                               array($date1, $date2, 'yd')
                              );
//             $retVal = 44
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'DATEDIF'),
                               array($date1, $date2, 'md')
                              );
//             $retVal = 13
```

**Notes**

If Date1 is later than Date2, DATEDIF will return a #NUM! error.

### 3.5.3. DATEVALUE

The DATEVALUE function returns the date represented by a date formatted as a text string. Use DATEVALUE to convert a date represented by text to a serial number.

**Syntax**

DATEVALUE(dateString)

**Parameters**

| | |
|---|---|
| dateString | **Date String.** A string, representing a date value. |

**Return Value**

| | |
|---|---|
| mixed | **A date/time stamp that corresponds to the given date.** This could be a PHP timestamp value (integer), a PHP date/time object, or an Excel timestamp value (real), depending on the value of PHPExcel_Calculation_Functions::getReturnDateType(). |

**Examples**

```
$worksheet->setCellValue('A1', 'Date String');

$worksheet->setCellValue('A2', '31-Dec-2008');
$worksheet->setCellValue('A3', '31/12/2008');
$worksheet->setCellValue('A4', '12-31-2008');

$worksheet->setCellValue('B2', '=DATEVALUE(A2)');
$worksheet->setCellValue('B3', '=DATEVALUE(A3)');
$worksheet->setCellValue('B4', '=DATEVALUE(A4)');

PHPExcel_Calculation_Functions::setReturnDateType(PHPExcel_Calculation_Functions::RETURNDATE_E
XCEL);
$retVal = $worksheet->getCell('B2')->getCalculatedValue();
$retVal = $worksheet->getCell('B3')->getCalculatedValue();
$retVal = $worksheet->getCell('B4')->getCalculatedValue();
//             $retVal = 39813.0 for all cases
```

```
//      We're going to be calling the same cell calculation multiple times,
//             and expecting different return values, so disable calculation cacheing
PHPExcel_Calculation::getInstance()->setCalculationCacheEnabled(False);

$saveFormat = PHPExcel_Calculation_Functions::getReturnDateType();
PHPExcel_Calculation_Functions::setReturnDateType(PHPExcel_Calculation_Functions::RETURNDATE_E
XCEL);
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'DATEVALUE'),
                               array('31-Dec-2008')
                              );
//             $retVal = 39813.0

PHPExcel_Calculation_Functions::setReturnDateType(PHPExcel_Calculation_Functions::RETURNDATE_P
HP_NUMERIC);
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'DATEVALUE'),
                               array('31-Dec-2008')
                              );
//             $retVal = 1230681600

PHPExcel_Calculation_Functions::setReturnDateType($saveFormat);
```

### Notes

DATEVALUE uses the php date/time object implementation of strtotime() (which can handle a wider range of formats than the normal strtotime() function), and it is also called for any date parameter passed to other date functions (such as DATEDIF) when the parameter value is a string.

WARNING:- PHPExcel accepts a wider range of date formats than MS Excel, so it is entirely possible that Excel will return a #VALUE! error when passed a date string that it can't interpret, while PHPExcel is able to translate that same string into a correct date value.
Care should be taken in workbooks that use string formatted dates in calculations when writing to Excel5 or Excel2007.

## 3.5.4. DAY

The DAY function returns the day of a date. The day is given as an integer ranging from 1 to 31.

### Syntax

DAY(datetime)

### Parameters

| | |
|---|---|
| **datetime** | **Date.**<br>An Excel date value, PHP date timestamp, PHP date object, or a date represented as a string. |

### Return Value

| | |
|---|---|
| **integer** | **An integer value that reflects the day of the month.**<br>This is an integer ranging from 1 to 31. |

### Examples

```
$worksheet->setCellValue('A1', 'Date String');

$worksheet->setCellValue('A2', '31-Dec-2008');
$worksheet->setCellValue('A3', '14-Feb-2008');

$worksheet->setCellValue('B2', '=DAY(A2)');
$worksheet->setCellValue('B3', '=DAY(A3)');

$retVal = $worksheet->getCell('B2')->getCalculatedValue();
//              $retVal = 31
$retVal = $worksheet->getCell('B3')->getCalculatedValue();
//              $retVal = 14
```

```
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'DAYOFMONTH'),
                               array('25-Dec-2008')
                              );
//              $retVal = 25
```

### Notes

Note that the PHPExcel function is PHPExcel_Calculation_Functions::DAYOFMONTH() when the method is called statically.

## 3.5.5. DAYS360

The DAYS360 function computes the difference between two dates based on a 360 day year (12 equal periods of 30 days each) used by some accounting systems.

## Syntax

DAYS360(date1, date2 [, method])

## Parameters

| | |
|---|---|
| **date1** | **First Date.** An Excel date value, PHP date timestamp, PHP date object, or a date represented as a string. |
| **date2** | **Second Date.** An Excel date value, PHP date timestamp, PHP date object, or a date represented as a string. |
| **method** | **A boolean flag (TRUE or FALSE)** This is a flag that determines which method to use in the calculation, based on the values listed below: |

| method | Description |
|---|---|
| FALSE | U.S. (NASD) method. If the starting date is the last day of a month, it becomes equal to the 30th of the same month. If the ending date is the last day of a month and the starting date is earlier than the 30th of a month, the ending date becomes equal to the 1st of the next month; otherwise the ending date becomes equal to the 30th of the same month. |
| TRUE | European method. Starting dates and ending dates that occur on the 31st of a month become equal to the 30th of the same month. |

The method value defaults to FALSE.

## Return Value

**integer** **An integer value that reflects the difference between the two dates.**
This is the number of full days between the two dates, based on a 360 day year.

## Examples

```
$worksheet->setCellValue('B1', 'Start Date');
$worksheet->setCellValue('C1', 'End Date');
$worksheet->setCellValue('A2', 'Year');
$worksheet->setCellValue('A3', 'Month');
$worksheet->setCellValue('A4', 'Day');

$worksheet->setCellValue('B2', 2003);
$worksheet->setCellValue('B3', 2);
$worksheet->setCellValue('B4', 3);
$worksheet->setCellValue('C2', 2007);
$worksheet->setCellValue('C3', 5);
$worksheet->setCellValue('C4', 31);

$worksheet->setCellValue('E2', '=DAYS360(DATE(B2,B3,B4),DATE(C2,C3,C4))');
$worksheet->setCellValue('E4', '=DAYS360(DATE(B2,B3,B4),DATE(C2,C3,C4),FALSE)');

$retVal = $worksheet->getCell('E2')->getCalculatedValue();
//              $retVal = 1558
$retVal = $worksheet->getCell('E4')->getCalculatedValue();
//              $retVal = 1557
```

```
$date1 = 37655.0              //      Excel timestamp for 25-Oct-2007
$date2 = 39233.0              //      Excel timestamp for 8-Dec-2015

$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'DAYS360'),
                               array($date1, $date2)
                              );
//              $retVal = 1558
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'DAYS360'),
                               array($date1, $date2, True)
                              );
//              $retVal = 1557
```

### Notes

WARNING:- This function does not currently work with the Excel5 Writer when a PHP Boolean is used for the third (optional) parameter (as shown in the example above), and the writer will generate and error. It will work if a numeric 0 or 1 is used for the method parameter; or if the Excel TRUE() and FALSE() functions are used instead.

## 3.5.6. EDATE

The EDATE function returns an Excel timestamp or a PHP timestamp or date object representing the date that is the indicated number of months before or after a specified date (the start_date). Use EDATE to calculate maturity dates or due dates that fall on the same day of the month as the date of issue.

### Syntax

EDATE(baseDate, months)

### Parameters

| | |
|---|---|
| **baseDate** | **Start Date.**<br>An Excel date value, PHP date timestamp, PHP date object, or a date represented as a string. |
| **months** | **Number of months to add.**<br>An integer value indicating the number of months before or after baseDate. A positive value for months yields a future date; a negative value yields a past date. |

### Return Value

| | |
|---|---|
| **mixed** | **A date/time stamp that corresponds to the basedate + months.**<br>This could be a PHP timestamp value (integer), a PHP date/time object, or an Excel timestamp value (real), depending on the value of PHPExcel_Calculation_Functions::getReturnDateType(). |

### Examples

```
$worksheet->setCellValue('A1', 'Date String');

$worksheet->setCellValue('A2', '1-Jan-2008');
$worksheet->setCellValue('A3', '29-Feb-2008');

$worksheet->setCellValue('B2', '=EDATE(A2,5)');
$worksheet->setCellValue('B3', '=EDATE(A3,-12)');

PHPExcel_Calculation_Functions::setReturnDateType(PHPExcel_Calculation_Functions::RETURNDATE_E
XCEL);
$retVal = $worksheet->getCell('B2')->getCalculatedValue();
//              $retVal = 39600.0      (1-Jun-2008)
$retVal = $worksheet->getCell('B3')->getCalculatedValue();
//              $retVal = 39141.0      (28-Feb-2007)
```

```
PHPExcel_Calculation_Functions::setReturnDateType(PHPExcel_Calculation_Functions::RETURNDATE_E
XCEL);
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'EDATE'),
                                 array('31-Oct-2008',25)
                                );
//              $retVal = 40512.0             (30-Nov-2010)
```

### Notes

WARNING:- This function is currently not supported by the Excel5 Writer because it is not a standard function within Excel 5, but an add-in from the Analysis ToolPak.

## 3.5.7. EOMONTH

The EOMONTH function returns an Excel timestamp or a PHP timestamp or date object representing the date of the last day of the month that is the indicated number of months before or after a specified date (the start_date). Use EOMONTH to calculate maturity dates or due dates that fall on the last day of the month.

### Syntax

EOMONTH(baseDate, months)

### Parameters

| | |
|---|---|
| baseDate | **Start Date.** An Excel date value, PHP date timestamp, PHP date object, or a date represented as a string. |
| months | **Number of months to add.** An integer value indicating the number of months before or after baseDate. A positive value for months yields a future date; a negative value yields a past date. |

### Return Value

| | |
|---|---|
| mixed | **A date/time stamp that corresponds to the last day of basedate + months.** This could be a PHP timestamp value (integer), a PHP date/time object, or an Excel timestamp value (real), depending on the value of PHPExcel_Calculation_Functions::getReturnDateType(). |

### Examples

```
$worksheet->setCellValue('A1', 'Date String');

$worksheet->setCellValue('A2', '1-Jan-2000');
$worksheet->setCellValue('A3', '14-Feb-2009');

$worksheet->setCellValue('B2', '=EOMONTH(A2,5)');
$worksheet->setCellValue('B3', '=EOMONTH(A3,-12)');

PHPExcel_Calculation_Functions::setReturnDateType(PHPExcel_Calculation_Functions::RETURNDATE_E
XCEL);
$retVal = $worksheet->getCell('B2')->getCalculatedValue();
//              $retVal = 39629.0      (30-Jun-2008)
$retVal = $worksheet->getCell('B3')->getCalculatedValue();
//              $retVal = 39507.0      (29-Feb-2008)
```

```
PHPExcel_Calculation_Functions::setReturnDateType(PHPExcel_Calculation_Functions::RETURNDATE_E
XCEL);
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'EOMONTH'),
                               array('31-Oct-2008',13)
                              );
//              $retVal = 40147.0              (30-Nov-2010)
```

### Notes

WARNING:- This function is currently not supported by the Excel5 Writer because it is not a standard function within Excel 5, but an add-in from the Analysis ToolPak.

## 3.5.8. HOUR

The HOUR function returns the hour of a time value. The hour is given as an integer, ranging from 0 (12:00 A.M.) to 23 (11:00 P.M.).

### Syntax

HOUR(datetime)

### Parameters

| | |
|---|---|
| **datetime** | **Time.** An Excel date/time value, PHP date timestamp, PHP date object, or a date/time represented as a string. |

### Return Value

| | |
|---|---|
| **integer** | **An integer value that reflects the hour of the day.** This is an integer ranging from 0 to 23. |

### Examples

```
$worksheet->setCellValue('A1', 'Time String');

$worksheet->setCellValue('A2', '31-Dec-2008 17:30');
$worksheet->setCellValue('A3', '14-Feb-2008 4:20 AM');
$worksheet->setCellValue('A4', '14-Feb-2008 4:20 PM');

$worksheet->setCellValue('B2', '=HOUR(A2)');
$worksheet->setCellValue('B3', '=HOUR(A3)');
$worksheet->setCellValue('B4', '=HOUR(A4)');

$retVal = $worksheet->getCell('B2')->getCalculatedValue();
//              $retVal = 17
$retVal = $worksheet->getCell('B3')->getCalculatedValue();
//              $retVal = 4
$retVal = $worksheet->getCell('B4')->getCalculatedValue();
//              $retVal = 16
```

```
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'HOUROFDAY'),
                               array('09:30')
                              );
//              $retVal = 9
```

### Notes

Note that the PHPExcel function is PHPExcel_Calculation_Functions::HOUROFDAY() when the method is called statically.

## 3.5.9. MINUTE

The MINUTE function returns the minutes of a time value. The minute is given as an integer, ranging from 0 to 59.

### Syntax

MINUTE(datetime)

**Parameters**

| | |
|---|---|
| **datetime** | **Time.** An Excel date/time value, PHP date timestamp, PHP date object, or a date/time represented as a string. |

**Return Value**

| | |
|---|---|
| **integer** | **An integer value that reflects the minutes within the hour.** This is an integer ranging from 0 to 59. |

**Examples**

```
$worksheet->setCellValue('A1', 'Time String');

$worksheet->setCellValue('A2', '31-Dec-2008 17:30');
$worksheet->setCellValue('A3', '14-Feb-2008 4:20 AM');
$worksheet->setCellValue('A4', '14-Feb-2008 4:45 PM');

$worksheet->setCellValue('B2', '=MINUTE(A2)');
$worksheet->setCellValue('B3', '=MINUTE(A3)');
$worksheet->setCellValue('B4', '=MINUTE(A4)');

$retVal = $worksheet->getCell('B2')->getCalculatedValue();
//             $retVal = 30
$retVal = $worksheet->getCell('B3')->getCalculatedValue();
//             $retVal = 20
$retVal = $worksheet->getCell('B4')->getCalculatedValue();
//             $retVal = 45
```

```
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'MINUTEOFHOUR'),
                               array('09:30')
                              );
//             $retVal = 30
```

**Notes**

Note that the PHPExcel function is PHPExcel_Calculation_Functions::MINUTEOFHOUR() when the method is called statically.

## 3.5.10. MONTH

The MONTH function returns the month of a date. The month is given as an integer ranging from 1 to 12.

**Syntax**

MONTH(datetime)

**Parameters**

| | |
|---|---|
| **datetime** | **Date.** An Excel date value, PHP date timestamp, PHP date object, or a date represented as a string. |

**Return Value**

| | |
|---|---|
| **integer** | **An integer value that reflects the month of the year.** This is an integer ranging from 1 to 12. |

**Examples**

```
$worksheet->setCellValue('A1', 'Date String');

$worksheet->setCellValue('A2', '31-Dec-2008');
$worksheet->setCellValue('A3', '14-Feb-2008');

$worksheet->setCellValue('B2', '=MONTH(A2)');
$worksheet->setCellValue('B3', '=MONTH(A3)');

$retVal = $worksheet->getCell('B2')->getCalculatedValue();
//              $retVal = 12
$retVal = $worksheet->getCell('B3')->getCalculatedValue();
//              $retVal = 2
```

```
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'MONTHOFYEAR'),
                               array('14-July-2008')
                              );
//              $retVal = 7
```

### Notes

Note that the PHPExcel function is PHPExcel_Calculation_Functions::MONTHOFYEAR() when the method is called statically.

## 3.5.11. NETWORKDAYS

The NETWORKDAYS function returns the number of whole working days between a *start date* and an *end date*. Working days exclude weekends and any dates identified in *holidays*. Use NETWORKDAYS to calculate employee benefits that accrue based on the number of days worked during a specific term.

### Syntax

NETWORKDAYS(startDate, endDate [, holidays])

### Parameters

| | |
|---|---|
| **startDate** | **Start Date of the period.** An Excel date value, PHP date timestamp, PHP date object, or a date represented as a string. |
| **endDate** | **End Date of the period.** An Excel date value, PHP date timestamp, PHP date object, or a date represented as a string. |
| **holidays** | **Optional array of Holiday dates.** An optional range of one or more dates to exclude from the working calendar, such as state and federal holidays and floating holidays. The list can be either a range of cells that contains the dates or an array constant of Excel date values, PHP date timestamps, PHP date objects, or dates represented as strings. |

### Return Value

| | |
|---|---|
| **integer** | **Number of working days.** The number of working days between startDate and endDate. |

### Examples

```
$worksheet->setCellValue('A1', 'Date String');

$worksheet->setCellValue('A2', '1-Jan-2000');
$worksheet->setCellValue('A3', '14-Feb-2009');

$worksheet->setCellValue('B2', '=EOMONTH(A2,5)');
```

```
$worksheet->setCellValue('B3', '=EOMONTH(A3,-12)');

PHPExcel_Calculation_Functions::setReturnDateType(PHPExcel_Calculation_Functions::RETURNDATE_E
XCEL);
$retVal = $worksheet->getCell('B2')->getCalculatedValue();
//              $retVal = 39629.0      (30-Jun-2008)
$retVal = $worksheet->getCell('B3')->getCalculatedValue();
//              $retVal = 39507.0      (29-Feb-2008)
```

```
PHPExcel_Calculation_Functions::setReturnDateType(PHPExcel_Calculation_Functions::RETURNDATE_E
XCEL);
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'EDATE'),
                               array('31-Oct-2008',13)
                              );
//              $retVal = 40147.0             (30-Nov-2010)
```

### Notes

There are no additional notes on this function

## 3.5.12. NOW

The NOW function returns the current date and time.

### Syntax

NOW()

### Parameters

**There are now parameters for the NOW() function.**

### Return Value

| | |
|---|---|
| mixed | **A date/time stamp that corresponds to the current date and time.** This could be a PHP timestamp value (integer), a PHP date/time object, or an Excel timestamp value (real), depending on the value of PHPExcel_Calculation_Functions::getReturnDateType(). |

### Examples

```
$worksheet->setCellValue('A1', 'Date String');

$worksheet->setCellValue('A2', '31-Dec-2008');
$worksheet->setCellValue('A3', '14-Feb-2008');

$worksheet->setCellValue('B2', '=MONTH(A2)');
$worksheet->setCellValue('B3', '=MONTH(A3)');

$retVal = $worksheet->getCell('B2')->getCalculatedValue();
//              $retVal = 12
$retVal = $worksheet->getCell('B3')->getCalculatedValue();
//              $retVal = 2
```

```
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'DATETIMENOW'),
                               array()
                              );
//              $retVal = 7
```

### Notes

Note that the PHPExcel function is PHPExcel_Calculation_Functions::DATETIMENOW() when the method is called statically.

## 3.5.13. SECOND

The SECOND function returns the seconds of a time value. The second is given as an integer, ranging from 0 to 59.

### Syntax

SECOND(datetime)

### Parameters

|  |  |
|---|---|
| **datetime** | **Time.** An Excel date/time value, PHP date timestamp, PHP date object, or a date/time represented as a string. |

### Return Value

|  |  |
|---|---|
| **integer** | **An integer value that reflects the seconds within the minute.** This is an integer ranging from 0 to 59. |

### Examples

```
$worksheet->setCellValue('A1', 'Time String');

$worksheet->setCellValue('A2', '31-Dec-2008 17:30:20');
$worksheet->setCellValue('A3', '14-Feb-2008 4:20 AM');
$worksheet->setCellValue('A4', '14-Feb-2008 4:45:59 PM');

$worksheet->setCellValue('B2', '=SECOND(A2)');
$worksheet->setCellValue('B3', '=SECOND(A3)');
$worksheet->setCellValue('B4', '=SECOND(A4)');

$retVal = $worksheet->getCell('B2')->getCalculatedValue();
//             $retVal = 20
$retVal = $worksheet->getCell('B3')->getCalculatedValue();
//             $retVal = 0
$retVal = $worksheet->getCell('B4')->getCalculatedValue();
//             $retVal = 59
```

```
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'SECONDOFMINUTE'),
                               array('09:30:17')
                              );
//             $retVal = 17
```

### Notes

Note that the PHPExcel function is PHPExcel_Calculation_Functions::SECONDOFMINUTE() when the method is called statically.

## 3.5.14. TIME

## 3.5.15. TIMEVALUE

### 3.5.16. TODAY

### 3.5.17. WEEKDAY

The WEEKDAY function returns the day of the week for a given date. The day is given as an integer ranging from 1 to 7, although this can be modified to return a value between 0 and 6.

**Syntax**

WEEKDAY(datetime [, method])

**Parameters**

| | |
|---|---|
| **datetime** | **Date.** An Excel date value, PHP date timestamp, PHP date object, or a date represented as a string. |
| **method** | **An integer flag (values 0, 1 or 2)** This is a flag that determines which method to use in the calculation, based on the values listed below: |

| method | Description |
|---|---|
| 0 | Returns 1 (Sunday) through 7 (Saturday). |
| 1 | Returns 1 (Monday) through 7 (Sunday). |
| 2 | Returns 0 (Monday) through 6 (Sunday). |

The method value defaults to 1.

**Return Value**

| | |
|---|---|
| **integer** | **An integer value that reflects the day of the week.** This is an integer ranging from 1 to 7, or 0 to 6, depending on the value of method. |

**Examples**

```
$worksheet->setCellValue('A1', 'Date String');

$worksheet->setCellValue('A2', '31-Dec-2008');
$worksheet->setCellValue('A3', '14-Feb-2008');

$worksheet->setCellValue('B2', '=WEEKDAY(A2)');
$worksheet->setCellValue('B3', '=WEEKDAY(A3,0)');
$worksheet->setCellValue('B4', '=WEEKDAY(A3,2)');

$retVal = $worksheet->getCell('B2')->getCalculatedValue();
//              $retVal = 12
$retVal = $worksheet->getCell('B3')->getCalculatedValue();
//              $retVal = 2
$retVal = $worksheet->getCell('B4')->getCalculatedValue();
//              $retVal = 2
```

```
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'DAYOFWEEK'),
                               array('14-July-2008')
                              );
//              $retVal = 7
```

**Notes**

Note that the PHPExcel function is PHPExcel_Calculation_Functions::DAYOFWEEK() when the method is called statically.

### 3.5.18. WEEKNUM

### 3.5.19. WORKDAY

### 3.5.20. YEAR

The YEAR function returns the year of a date.

**Syntax**

YEAR(datetime)

**Parameters**

| | |
|---|---|
| **datetime** | **Date.**<br>An Excel date value, PHP date timestamp, PHP date object, or a date represented as a string. |

**Return Value**

| | |
|---|---|
| **integer** | **An integer value that reflects the month of the year.**<br>This is an integer year value. |

**Examples**

```
$worksheet->setCellValue('A1', 'Date String');

$worksheet->setCellValue('A2', '17-Jul-1982');
$worksheet->setCellValue('A3', '16-Apr-2009');

$worksheet->setCellValue('B2', '=YEAR(A2)');
$worksheet->setCellValue('B3', '=YEAR(A3)');

$retVal = $worksheet->getCell('B2')->getCalculatedValue();
//             $retVal = 1982
$retVal = $worksheet->getCell('B3')->getCalculatedValue();
//             $retVal = 2009
```

```
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'YEAR'),
                               array('14-July-2001')
                              );
//             $retVal = 2001
```

**Notes**

There are no additional notes on this function

### 3.5.21. YEARFRAC

## *3.6. Engineering Functions*

### 3.6.1. BESSELI

### 3.6.2. BESSELJ

### 3.6.3. BESSELK

### 3.6.4. BESSELY

### 3.6.5. BIN2DEC

### 3.6.6. BIN2HEX

### 3.6.7. BIN2OCT

### 3.6.8. COMPLEX

### 3.6.9. CONVERT

### 3.6.10. DEC2BIN

### 3.6.11. DEC2HEX

### 3.6.12. DEC2OCT

### 3.6.13. DELTA

### 3.6.14. ERF

### 3.6.15. ERFC

### 3.6.16. GESTEP

### 3.6.17. HEX2BIN

### 3.6.18. HEX2DEC

### 3.6.19. HEX2OCT

### 3.6.20. IMABS

### 3.6.21. IMAGINARY

### 3.6.22. IMARGUMENT

### 3.6.23. IMCONJUGATE

### 3.6.24. IMCOS

### 3.6.25. IMDIV

### 3.6.26. IMEXP

### 3.6.27. IMLN

### 3.6.28. IMLOG10

### 3.6.29. IMLOG2

### 3.6.30. IMPOWER

### 3.6.31. IMPRODUCT

### 3.6.32. IMREAL

### 3.6.33. IMSIN

### 3.6.34. IMSQRT

### 3.6.35. IMSUB

### 3.6.36. IMSUM

### 3.6.37. OCT2BIN

### 3.6.38. OCT2DEC

### 3.6.39. OCT2HEX

## 3.7. *Financial Functions*

### 3.7.1. ACCRINT

Not yet implemented.

### 3.7.2. ACCRINTM

Not yet implemented.

### 3.7.3. AMORDEGRC

Not yet implemented.

### 3.7.4. AMORLINC

Not yet implemented.

### 3.7.5. COUPDAYBS

Not yet implemented.

### 3.7.6. COUPDAYSNC

Not yet implemented.

### 3.7.7. COUPNCD

Not yet implemented.

### 3.7.8. COUPNUM

Not yet implemented.

### 3.7.9. COUPPCD

Not yet implemented.

### 3.7.10. CUMIPMT

Not yet implemented.

### 3.7.11. CUMPRINC

Not yet implemented.

### 3.7.12. DB

DB returns the depreciation of an asset for a specified period using the double-declining balance method.
This form of depreciation is used if you want to get a higher depreciation value at the beginning of the depreciation (as opposed to linear depreciation). The depreciation value is reduced with every depreciation period by the depreciation already deducted from the initial cost.

#### Syntax

```
DB(cost, salvage, life, period [, month])
```

#### Parameters

| | |
|---|---|
| **cost** | **Float**<br>The initial, purchase value of an asset. |
| **salvage** | **Float** |

| | |
|---|---|
| | The value of an asset at the end of the depreciation (sometimes called the salvage value of the asset). |
| **life** | **Integer**<br>The number of periods over which the asset is being depreciated (sometimes called the useful life of the asset). |
| **period** | **Integer**<br>The period for which you want to calculate the depreciation. Period must use the same units as life. |
| **month** | **Integer**<br>Number of months in the first year. If month is omitted, it is assumed to be 12. |

### Return Value

| | |
|---|---|
| Float | Depreciation.<br>The depreciation of the asset for the specified period |

### Examples

```
$worksheet->setCellValue('A1', 'Purchase Cost');
$worksheet->setCellValue('B1', 17500);
$worksheet->setCellValue('A2', 'Scrap Value');
$worksheet->setCellValue('B2', 6500);
$worksheet->setCellValue('A3', 'Depreciation Lifespan');
$worksheet->setCellValue('B3', 4);
$worksheet->setCellValue('A4', 'Period');
$worksheet->setCellValue('B4', 2);

$worksheet->setCellValue('D'.$period, '=DB(B1,B2,B3,B4)');
$retVal = $worksheet->getCell('C1')->getCalculatedValue();
//             $retVal = 2993.1825
```

```
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'DB'),
                                  array(17500,6500,4,2);
//             $retVal = 2993.1825
```

### Notes

Excel rounds the fixed depreciation rate used within this function to three decimal places, as does Open Office Calc and Gnumeric; and this behaviour is replicated within PHPExcel. It is planned that at some point PHPExcel will provide an option that allows this function to be calculated without rounding to provide an accurate result.

## 3.7.13. DDB

Not yet implemented.

## 3.7.14. DISC

Not yet implemented.

## 3.7.15. DOLLARDE

Not yet implemented.

## 3.7.16. DOLLARFR

Not yet implemented.

### 3.7.17. DURATION
Not yet implemented.

### 3.7.18. EFFECT

### 3.7.19. FV

### 3.7.20. FVSCHEDULE
Not yet implemented.

### 3.7.21. INTRATE
Not yet implemented.

### 3.7.22. IPMT
Not yet implemented.

### 3.7.23. IRR
Not yet implemented.

### 3.7.24. MDURATION
Not yet implemented.

### 3.7.25. MIRR
Not yet implemented.

### 3.7.26. NOMINAL

### 3.7.27. NPER

### 3.7.28. NPV

### 3.7.29. ODDFPRICE
Not yet implemented.

### 3.7.30. ODDFYIELD
Not yet implemented.

### 3.7.31. ODDLPRICE
Not yet implemented.

### 3.7.32. ODDLYIELD
Not yet implemented.

### 3.7.33. ORICEDISC
Not yet implemented.

### 3.7.34. PMT

### 3.7.35. PPMT
Not yet implemented.

### 3.7.36. PRICE
Not yet implemented.

### 3.7.37. PRICEMAT
Not yet implemented.

### 3.7.38. PV

### 3.7.39. RATE
Not yet implemented.

### 3.7.40. RECEIVED
Not yet implemented.

### 3.7.41. SLN

### 3.7.42. SYD

### 3.7.43. TBILLEQ
Not yet implemented.

### 3.7.44. TBILLPRICE
Not yet implemented.

### 3.7.45. TBILLYIELD
Not yet implemented.

### 3.7.46. USDOLLAR
Not yet implemented.

### 3.7.47. VDB
Not yet implemented.

### 3.7.48. XIRR
Not yet implemented.

### 3.7.49. XNPV
Not yet implemented.

### 3.7.50. YIELD
Not yet implemented.

### 3.7.51. YIELDDISC

Not yet implemented.

### 3.7.52. YIELDMAT

Not yet implemented.

## 3.8. Information Functions

### 3.8.1. CELL
Not yet implemented.

### 3.8.2. ERROR.TYPE

### 3.8.3. INFO
Not yet implemented.

### 3.8.4. ISBLANK

### 3.8.5. ISERR

### 3.8.6. ISERROR

### 3.8.7. ISEVEN

### 3.8.8. ISLOGICAL

### 3.8.9. ISNA

### 3.8.10. ISNONTEXT

### 3.8.11. ISNUMBER

### 3.8.12. ISODD

### 3.8.13. ISPMT
Not yet implemented.

### 3.8.14. ISREF
Not yet implemented.

### 3.8.15. ISTEXT

### 3.8.16. N

Not yet implemented.

### 3.8.17. NA

### 3.8.18. TYPE

Not yet implemented.

### 3.8.19. VERSION

## 3.9. *Logical Functions*

### 3.9.1. AND

### 3.9.2. FALSE

### 3.9.3. IF

### 3.9.4. IFERROR

### 3.9.5. NOT

### 3.9.6. OR

### 3.9.7. TRUE

## 3.10. *Lookup and Reference Functions*

### 3.10.1. ADDRESS

### 3.10.2. AREAS
Not yet implemented.

### 3.10.3. CHOOSE

### 3.10.4. COLUMN

### 3.10.5. COLUMNS
Not yet implemented.

### 3.10.6. GETPIVOTDATA
Not yet implemented.

### 3.10.7. HLOOKUP
Not yet implemented.

### 3.10.8. HYPERLINK
Not yet implemented.

### 3.10.9. INDEX

### 3.10.10. INDIRECT
Not yet implemented.

### 3.10.11. LOOKUP

### 3.10.12. MATCH

### 3.10.13. OFFSET

### 3.10.14. ROW

### 3.10.15. ROWS
Not yet implemented.

### 3.10.16. RTD

Not yet implemented.

### 3.10.17. TRANSPOSE

### 3.10.18. VLOOKUP

## 3.11. Mathematical and Trigonometric Functions

### 3.11.1. ABS

ABS implements the Absolute Value function: the result is to drop the negative sign (if present). This can be done for integers and floating point numbers.

**Syntax**

ABS(number)

**Parameters**

| | |
|---|---|
| number | The number whose absolute value is to be calculated. |

**Return Value**

| | |
|---|---|
| Float | The absolute value of the parameter value. |

**Examples**

```
$worksheet->setCellValue('A1', -5);
$worksheet->setCellValue('C1', '=ABS(A1)');
$retVal = $worksheet->getCell('C1')->getCalculatedValue();
//             $retVal = 5
```

```
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'ABS'),
                                  array(-5.1);
//             $retVal = 5.1
```

**Notes**

PHPExcel calls the standard PHP abs() function directly, with no additional validation of parameter data.

The Excel ABS() function will accept any data type, treating booleans as numeric 0 or 1, accepting strings containing numeric values, and returning a zero (0) if passed a null value. Excel will return a '#VALUE!' error if passed an empty string, or a non-numeric string.
The PHP abs() function will also work with booleans and strings that contain a numeric value; but returns a zero (0) value from an empty string, and gives a 'Wrong parameter count' warning when passed a null value.

### 3.11.2. ACOS

ACOS returns the arccosine, or inverse cosine, of a number. The arccosine is the angle whose cosine is the input number. The returned angle is given in radians in the range 0 (zero) to pi.

**Syntax**

ACOS(cosine)

**Parameters**

| | |
|---|---|
| cosine | The cosine whose angle is to be calculated. |

**Return Value**

| | |
|---|---|
| Float | The angle in radians whose cosine is the value of the parameter. |

### Examples

```
$worksheet->setCellValue('A1', 0.1);
$worksheet->setCellValue('C1', '= ACOS(A1)');
$retVal = $worksheet->getCell('C1')->getCalculatedValue();
//              $retVal = 1.47062890563
```

```
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'ACOS'),
                                  array(0.35);
//              $retVal = 1.21322522315
```

### Notes

PHPExcel calls the standard PHP acos() function directly, with no additional validation of parameter data.

Excel will return a '#NUM!' error if the cosine value is < -1.0 or > 1.0, while the PHP acos() function will return NAN.
The Excel ACOS() function will accept any data type, treating booleans as numeric 0 or 1, accepting strings containing numeric values, and treating a null value as a numeric 0. Excel will return a '#VALUE!' error if passed an empty string, or a non-numeric string.
The PHP acos() function will also work with booleans and strings that contain a numeric value; but treats an empty or non-numeric string as a numeric 0, and gives a 'Wrong parameter count' warning when passed a null value.

## 3.11.3. ACOSH

ACOSH returns the inverse hyperbolic cosine of a number.

### Syntax

ACOSH(number)

### Parameters

| | |
|---|---|
| number | The hyperbolic cosine whose inverse is to be calculated, number must be greater than or equal to 1. |

### Return Value

| | |
|---|---|
| Float | The angle in radians whose hyperbolic cosine is the value of the parameter. |

### Examples

```
$worksheet->setCellValue('A1', 5.3);
$worksheet->setCellValue('C1', '=ACOSH(A1)');
$retVal = $worksheet->getCell('C1')->getCalculatedValue();
//              $retVal = 2.35183281645
```

```
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'ACOSH'),
                                  array(10);
//              $retVal = 2.99322284613
```

### Notes

On most operating platforms/PHP versions, PHPExcel calls the standard PHP acosh() function directly, with no additional validation of parameter data. However, on Windows with a PHP version prior to 5.3.0, the acosh() function is not implemented. In this case, PHPExcel implements its own acosh() function, without any additional validation of parameter data.

Excel will return a '#NUM!' error if the hyperbolic cosine value is < 1.0, while the PHP (or PHPExcel) acosh() function will return NAN.
The Excel ACOSH() function will accept any data type, treating booleans as numeric 0 (which will return a '#NUM!' error) or 1, accept strings containing numeric values, and treat a null value as a numeric 0 (which will return a '#NUM!' error). Excel will return a '#VALUE!' error if passed an empty string, or a non-numeric string.
The PHP abs() function will also work with booleans and strings that contain a numeric value; but treats an empty or non-numeric string as a numeric 0, and gives a 'Wrong parameter count' warning when passed a null value.

## 3.11.4. ASIN

ASIN returns the arcsine, or inverse sine, of a number. The arcsine is the angle whose sine is the input number. The returned angle is given in radians in the range 0 (zero) to pi.

### Syntax

ASIN(sine)

### Parameters

| | |
|---|---|
| sine | The sine whose angle is to be calculated. |

### Return Value

| | |
|---|---|
| Float | The angle in radians whose sine is the value of the parameter. |

### Examples

```
$worksheet->setCellValue('A1', 0.1);
$worksheet->setCellValue('C1', '= ASIN(A1)');
$retVal = $worksheet->getCell('C1')->getCalculatedValue();
//              $retVal = 0.100167421162
```

```
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'ASIN'),
                                  array(0.35);
//              $retVal = 0.357571103646
```

### Notes

PHPExcel calls the standard PHP asin() function directly, with no additional validation of parameter data.

Excel will return a '#NUM!' error if the sine value is < -1.0 or > 1.0, while the PHP asin() function will return NAN.
The Excel ASIN() function will accept any data type, treating booleans as numeric 0 or 1, accepting strings containing numeric values, and treating a null value as a numeric 0. Excel will return a '#VALUE!' error if passed an empty string, or a non-numeric string.
The PHP asin() function will also work with booleans and strings that contain a numeric value; but treats an empty or non-numeric string as a numeric 0, and gives a 'Wrong parameter count' warning when passed a null value.

### 3.11.5. ASINH

ASINH returns the inverse hyperbolic sine of a number.

**Syntax**

ASINH(number)

**Parameters**

| | |
|---|---|
| number | The hyperbolic sine whose inverse is to be calculated. |

**Return Value**

| | |
|---|---|
| Float | The angle in radians whose hyperbolic sine is the value of the parameter. |

**Examples**

```
$worksheet->setCellValue('A1', -2.5);
$worksheet->setCellValue('C1', '=ASINH(A1)');
$retVal = $worksheet->getCell('C1')->getCalculatedValue();
//              $retVal = -1.647231146371
```

```
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'ASINH'),
                                  array(0.5));
//              $retVal = 0.481211825060
```

**Notes**

On most operating platforms/PHP versions, PHPExcel calls the standard PHP asinh() function directly, with no additional validation of parameter data. However, on Windows with a PHP version prior to 5.3.0, the asinh() function is not implemented. In this case, PHPExcel implements its own asinh() function, without any additional validation of parameter data.

The Excel ACOSH() function will accept any data type, treating booleans as numeric 0 or 1, accept strings containing numeric values, and treat a null value as a numeric 0. Excel will return a '#VALUE!' error if passed an empty string, or a non-numeric string.
The PHP abs() function will also work with booleans and strings that contain a numeric value; but treats an empty or non-numeric string as a numeric 0, and gives a 'Wrong parameter count' warning when passed a null value.

### 3.11.6. ATAN

### 3.11.7. ATAN2

### 3.11.8. ATANH

### 3.11.9. CEILING

### 3.11.10. COMBIN

### 3.11.11. COS

### 3.11.12. COSH

### 3.11.13. DEGREES

### 3.11.14. EVEN

### 3.11.15. EXP

### 3.11.16. FACT

### 3.11.17. FACTDOUBLE

### 3.11.18. FLOOR

### 3.11.19. GCD

### 3.11.20. INT

### 3.11.21. LCM

### 3.11.22. LN

### 3.11.23. LOG

### 3.11.24. LOG10

### 3.11.25. MDETERM
Not yet implemented.

### 3.11.26. MINVERSE
Not yet implemented.

### 3.11.27. MMULT

### 3.11.28. MOD

### 3.11.29. MROUND

### 3.11.30. MULTINOMIAL

### 3.11.31. ODD

### 3.11.32. PI

### 3.11.33. POWER

### 3.11.34. PRODUCT

### 3.11.35. QUOTIENT

### 3.11.36. RADIANS

### 3.11.37. RAND

### 3.11.38. RANDBETWEEN

### 3.11.39. ROMAN
Not yet implemented.

### 3.11.40. ROUND

### 3.11.41. ROUNDDOWN

### 3.11.42. ROUNDUP

### 3.11.43. SERIESSUM

### 3.11.44. SIGN

### 3.11.45. SIN

### 3.11.46. SINH

### 3.11.47. SQRT

### 3.11.48. SQRTPI

### 3.11.49. SUBTOTAL

### 3.11.50. SUM

### 3.11.51. SUMIF

Not yet implemented.

### 3.11.52. SUMIFS

Not yet implemented.

### 3.11.53. SUMPRODUCT

Not yet implemented.

### 3.11.54. SUMSQ

### 3.11.55. SUMX2MY2

Not yet implemented.

### 3.11.56. SUMX2PY2

Not yet implemented.

### 3.11.57. SUMXMY2

Not yet implemented.

### 3.11.58. TAN

### 3.11.59. TANH

### 3.11.60. TRUNC

## 3.12. Statistical Functions

### 3.12.1. AVEDEV

AVEDEV returns the average of the absolute deviations of a data set from their mean. This is a measure of the variability in the data set.

**Syntax**

AVEDEV(n1, n2, ...)

**Parameters**

n… A series of data values or cell references.

Only numeric data values are included in the AVEDEV calculation. Nulls, strings (including numeric values in a string datatype) and booleans are ignored.

**Return Value**

Float

**Examples**

```
$i = 1;
$dataValues = array(11.4, 17.3, 21.3, 25.9, 40.1);
foreach ($dataValues as $dataValue) {
        $worksheet->setCellValue('A'.$i++, $dataValue);
}
$worksheet->setCellValue('C1', '=AVEDEV(A1:A'.--$i.')');
$retVal = $worksheet->getCell('C1')->getCalculatedValue();
//              $retVal = 7.84
```

```
$dataValues = array(11.4, 17.3, 21.3, 25.9, 40.1);
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'AVEDEV'),
                                   $dataValues);
//              $retVal = 7.84
```

**Notes**

If you set compatibility mode to OPENOFFICE, Boolean values in the data series are not ignored: a boolean FALSE is evaluated as value zero (0) and TRUE as one (1).

### 3.12.2. AVERAGE

AVERAGE computes the average (arithmetic mean) of all the values and cells referenced in the argument list. This is equivalent to the sum of the arguments divided by the count of the arguments.

**Syntax**

AVERAGE(n1, n2, ...)

**Parameters**

n… A series of data values or cell references.

Only numeric data values are included in the AVERAGE calculation. Nulls, strings (including numeric values in a string datatype) and booleans are ignored.

### Return Value

Float

### Examples

```
$i = 1;
$dataValues = array(11.4, 17.3, 21.3, 25.9, 40.1);
foreach ($dataValues as $dataValue) {
        $worksheet->setCellValue('A'.$i++, $dataValue);
}
$worksheet->setCellValue('C1', '=AVERAGE(A1:A'.--$i.')');
$retVal = $worksheet->getCell('C1')->getCalculatedValue();
//              $retVal = 23.2
```

```
$dataValues = array(11.4, 17.3, 21.3, 25.9, 40.1);
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'AVERAGE'),
                                   $dataValues);
//              $retVal = 23.2
```

## 3.12.3. AVERAGEA

AVERAGEA returns the average (arithmetic mean) of all the values and cells referenced in the argument list. Numbers, text and logical values are included in the calculation too.

### Syntax

AVERAGEA(n1, n2, ...)

### Parameters

| | |
|---|---|
| n… | A series of data values or cell references. |

If a data value or cell contains text or the argument evaluates to FALSE, it is counted as value zero (0). If the argument evaluates to TRUE, it is counted as one (1). Note that empty cells are not counted.

### Return Value

Float

### Examples

```
$i = 1;
$dataValues = array(11.4, 17.3, 'missing', 25.9, 40.1);
foreach ($dataValues as $dataValue) {
        $worksheet->setCellValue('A'.$i++, $dataValue);
}
$worksheet->setCellValue('C1', '=AVERAGEA(A1:A'.--$i.')');
$retVal = $worksheet->getCell('C1')->getCalculatedValue();
//              $retVal = 18.94
```

```
$dataValues = array(11.4, 17.3, 'missing', 25.9, 40.1);
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'AVERAGEA'),
                                   $dataValues);
```

```
//              $retVal = 18.94
```

### 3.12.4. AVERAGEIF

This function has not yet been implemented.

### 3.12.5. AVERAGEIFS

This function has not yet been implemented.

### 3.12.6. BETADIST

The BetaDist function returns the cumulative beta probability density function.

#### Syntax

BETADIST(x, alpha, beta [, A, B])

#### Parameters

| | |
|---|---|
| x | is the value between A and B at which to evaluate the function. |
| alpha | is a parameter of the distribution. |
| beta | is a parameter of the distribution. |
| A | is an optional lower bound to the interval of x. |
| B | is an optional upper bound to the interval of x. |

#### Validation

If any argument is nonnumeric, BETADIST returns the #VALUE! error value.
If alpha ≤ 0 or beta ≤ 0, BETADIST returns the #NUM! error value.
If x < A, x > B, or A = B, BETADIST returns the #NUM! error value.
If you omit values for A and B, BETADIST uses the standard cumulative beta distribution, so that A = 0 and B = 1.

#### Return Value

Float

#### Examples

```
$dataValues = array(  array( 'label' => 'x',        'value' => 2 ),
                      array( 'label' => 'alpha',    'value' => 8 ),
                      array( 'label' => 'beta',     'value' => 10 ),
                      array( 'label' => 'A',        'value' => 1 ),
                      array( 'label' => 'B',        'value' => 3 )
                   );

$i = 1;
foreach ($dataValues as $dataValue) {
       $worksheet->setCellValue('A'.$i, $dataValue['label']);
       $worksheet->setCellValue('B'.$i++, $dataValue['value']);
}
$PHPExcelObject ->getActiveSheet()->setCellValue('D1', '=BETADIST(B1,B2,B3,B4,B5)');
$retVal = $PHPExcelObject ->getActiveSheet()->getCell('D1')->getCalculatedValue();
//              $retVal = 0.685470581
```

```
$dataValues = array(  array( 'label' => 'x',        'value' => 2 ),
                      array( 'label' => 'alpha',    'value' => 8 ),
                      array( 'label' => 'beta',     'value' => 10 ),
                      array( 'label' => 'A',        'value' => 1 ),
                      array( 'label' => 'B',        'value' => 3 )
                   );

foreach ($dataValues as $dataValue) {
        $$dataValue['label'] = $dataValue['value'];
}
$retVal = call_user_func_array( array('PHPExcel_Calculation_Functions','BETADIST'),
                                array($x, $alpha, $beta, $A, $B));
//              $retVal = 0.685470581
```

```
$dataValues = array(  array( 'label' => 'x',        'value'=> 2 ),
                      array( 'label' => 'alpha',    'value'=> 8 ),
                      array( 'label' => 'beta',     'value'=> 10 ),
                      array( 'label' => 'A',        'value'=> 1 ),
                      array( 'label' => 'B',        'value'=> 3 )
                   );

$parameterValues = array();
foreach ($dataValues as $dataValue) {
        $parameterValues[] = $dataValue['value'];
}
$retVal = call_user_func_array( array('PHPExcel_Calculation_Functions','BETADIST'),
                                $parameterValues);
//              $retVal = 0.685470581
```

### 3.12.7. BETAINV

BETAINV returns the inverse of the cumulative distribution function for a specified beta distribution. That is, if probability = BETADIST(x,...), then BETAINV(probability,...) = x.
The beta distribution can be used in project planning to model probable completion times given an expected completion time and variability.

**Syntax**

BETAINV(probability, alpha, beta [, A, B])

**Parameters**

| | |
|---|---|
| probability | is a probability associated with the beta distribution. |
| alpha | is a parameter of the distribution. |
| beta | is a parameter of the distribution. |
| A | is an optional lower bound to the interval of x. |
| B | is an optional upper bound to the interval of x. |

**Validation**

If any argument is nonnumeric, BETAINV returns the #VALUE! error value.
If alpha ≤ 0 or beta ≤ 0, BETAINV returns the #NUM! error value.
If probability ≤ 0 or probability > 1, BETAINV returns the #NUM! error value.
If you omit values for A and B, BETAINV uses the standard cumulative beta distribution, so that A = 0 and B = 1.

**Return Value**

Float

**Examples**

### 3.12.8. BINOMDIST

### 3.12.9. CHIDIST

### 3.12.10. CHIINV

### 3.12.11. CHITEST
This function has not yet been implemented.

### 3.12.12. CONFIDENCE

### 3.12.13. CORREL
This function has not yet been implemented.

### 3.12.14. COUNT
COUNT returns the total number of integer or floating point arguments passed.

#### Syntax
COUNT(n1, n2, ...)

#### Parameters

| | |
|---|---|
| n… | A series of data values or cell references. |

Only numeric data values are included in the COUNT calculation. Nulls, strings (including numeric values in a string datatype) and booleans are ignored.

#### Return Value
Integer

#### Examples

```
$i = 1;
$dataValues = array(11.4, 17.3, '21.3', 25.9, 'String', 40.1);
foreach ($dataValues as $dataValue) {
	$worksheet->setCellValue('A'.$i++, $dataValue);
}
$worksheet->setCellValue('C1', '=COUNT(A1:A'.--$i.')');
$retVal = $worksheet->getCell('C1')->getCalculatedValue();
//		$retVal = 4
```

```
$dataValues = array(11.4, 17.3, 21.3, False, 25.9, 40.1);
$retVal = call_user_func_array(array('PHPExcel_Calculation_Functions', 'COUNT'),
                               $dataValues);
//		$retVal = 5
```

### 3.12.15. COUNTA

### 3.12.16. COUNTBLANK

### 3.12.17. COUNTIF
This function has not yet been implemented.

#### 3.12.18. COUNTIFS
This function has not yet been implemented.

#### 3.12.19. COVAR
This function has not yet been implemented.

#### 3.12.20. CRITBINOM

#### 3.12.21. DEVSQ

#### 3.12.22. EXPONDIST

#### 3.12.23. FDIST
This function has not yet been implemented.

#### 3.12.24. FINV
This function has not yet been implemented.

#### 3.12.25. FISHER

#### 3.12.26. FISHERINV

#### 3.12.27. FORECAST
This function has not yet been implemented.

#### 3.12.28. FREQUENCY
This function has not yet been implemented.

#### 3.12.29. FTEST
This function has not yet been implemented.

#### 3.12.30. GAMMADIST

#### 3.12.31. GAMMAINV

#### 3.12.32. GAMMALN

#### 3.12.33. GEOMEAN

#### 3.12.34. GROWTH
This function has not yet been implemented.

### 3.12.35. HARMEAN

### 3.12.36. HYPGEOMDIST

### 3.12.37. INTERCEPT

This function has not yet been implemented.

### 3.12.38. KURT

### 3.12.39. LARGE

### 3.12.40. LINEST

This function has not yet been implemented.

### 3.12.41. LOGEST

This function has not yet been implemented.

### 3.12.42. LOGINV

### 3.12.43. LOGNORMDIST

### 3.12.44. MAX

### 3.12.45. MAXA

### 3.12.46. MEDIAN

### 3.12.47. MIN

### 3.12.48. MINA

### 3.12.49. MODE

### 3.12.50. NEGBINOMDIST

### 3.12.51. NORMDIST

#### 3.12.52. NORMINV

#### 3.12.53. NORMSDIST

#### 3.12.54. NORMSINV

#### 3.12.55. PEARSON

This function has not yet been implemented.

#### 3.12.56. PERCENTILE

#### 3.12.57. PERCENTRANK

This function has not yet been implemented.

#### 3.12.58. PERMUT

#### 3.12.59. POISSON

#### 3.12.60. PROB

This function has not yet been implemented.

#### 3.12.61. QUARTILE

#### 3.12.62. RANK

This function has not yet been implemented.

#### 3.12.63. RSQ

This function has not yet been implemented.

#### 3.12.64. SKEW

#### 3.12.65. SLOPE

This function has not yet been implemented.

#### 3.12.66. SMALL

#### 3.12.67. STANDARDIZE

#### 3.12.68. STDEV

### 3.12.69. STDEVA

### 3.12.70. STDEVP

### 3.12.71. STDEVPA

### 3.12.72. STEYX
This function has not yet been implemented.

### 3.12.73. TDIST

### 3.12.74. TINV

### 3.12.75. TREND
This function has not yet been implemented.

### 3.12.76. TRIMMEAN

### 3.12.77. TTEST
This function has not yet been implemented.

### 3.12.78. VAR

### 3.12.79. VARA

### 3.12.80. VARP

### 3.12.81. VARPA

### 3.12.82. WEIBULL

### 3.12.83. ZTEST
This function has not yet been implemented.

## 3.13. Text and Data Functions

### 3.13.1. ASC

This function has not yet been implemented.

### 3.13.2. BAHTTEXT

This function has not yet been implemented.

### 3.13.3. CHAR

### 3.13.4. CLEAN

### 3.13.5. CODE

### 3.13.6. CONCATENATE

### 3.13.7. DOLLAR

This function has not yet been implemented.

### 3.13.8. EXACT

This function has not yet been implemented.

### 3.13.9. FIND

### 3.13.10. FINDB

This function has not yet been implemented.

### 3.13.11. FIXED

This function has not yet been implemented.

### 3.13.12. JIS

This function has not yet been implemented.

### 3.13.13. LEFT

### 3.13.14. LEFTB

This function has not yet been implemented.

### 3.13.15. LEN

### 3.13.16. LENB
This function has not yet been implemented.

### 3.13.17. LOWER

### 3.13.18. MID

### 3.13.19. MIDB
This function has not yet been implemented.

### 3.13.20. PHONETIC
This function has not yet been implemented.

### 3.13.21. PROPER

### 3.13.22. REPLACE
This function has not yet been implemented.

### 3.13.23. REPLACEB
This function has not yet been implemented.

### 3.13.24. REPT

### 3.13.25. RIGHT

### 3.13.26. RIGHTB
This function has not yet been implemented.

### 3.13.27. SEARCH

### 3.13.28. SEARCHB
This function has not yet been implemented.

### 3.13.29. SUBSTITUTE
This function has not yet been implemented.

### 3.13.30. T

### 3.13.31. TEXT
This function has not yet been implemented.

### 3.13.32. TRIM

### 3.13.33. UPPER

### 3.13.34. VALUE

This function has not yet been implemented.

## 4. Credits

Please refer to the internet page http://www.codeplex.com/PHPExcel/Wiki/View.aspx?title=Credits&referringTitle=Home for up-to-date credits.