How to use NetworkDays in VBA (With Example)

Authored by

stats writer

November 18, 2025

In sophisticated financial modeling, project management, and scheduling applications built within Microsoft Excel, the ability to accurately calculate the number of working days between two specific dates is paramount. When automating these calculations using VBA (Visual Basic for Applications), developers often seek an efficient and reliable method that inherently excludes non-working periods, such as weekends and defined holidays. While one could write extensive custom code to iterate through dates and check for weekdays, the standard functionality of Excel provides a far more streamlined solution, which is accessible directly within your VBA procedures. This solution leverages the power of Excel's built-in functions via the WorksheetFunction object.

This article serves as an expert guide to utilizing the **NetworkDays** method within VBA. The **NetworkDays** function is essential for calculating the net number of whole working days between a designated start date and an end date. Crucially, this method automatically excludes Saturday and Sunday as non-working days, and it can be optionally configured to exclude specific holidays defined by the user. Understanding how to properly implement this function within a macro will significantly enhance the robustness and efficiency of your automated date calculations, allowing you to focus on the core logic of your application rather than complex date manipulation algorithms.

We will provide a detailed, step-by-step example demonstrating how to integrate the **NetworkDays** function into a VBA macro designed to process multiple rows of date ranges efficiently. This comprehensive approach ensures that you gain not only theoretical knowledge of the function but also practical, applicable coding skills necessary for immediate implementation in your projects.

Understanding the NetworkDays Function

The core purpose of the **NetworkDays** method is to calculate the duration of a project or task based solely on business days, which is a requirement across many professional sectors, including finance, logistics, and human resources. When accessed via VBA, this function is exposed through the **WorksheetFunction** object, making Excel's powerful calculation engine available directly within the procedural environment of your code. By utilizing `WorksheetFunction.NetworkDays`, you bypass the need for tedious manual calculations or custom holiday arrays, significantly simplifying the code required to achieve accurate business day counts.

It is critical to remember the precise definition of a "working day" when using this function: by default, **NetworkDays** considers Monday through Friday as working days, while Saturday and Sunday are treated as non-working weekend days. If your business operates on a different schedule (e.g., Sunday through Thursday), you would need to use the related function, **NetworkDays.Intl**, which provides parameters for customizing the weekend structure. However, for standard Monday-Friday operations, the basic **NetworkDays** method is the most straightforward choice.

The syntax is straightforward, requiring at minimum a start date and an end date. The result returned by the function is an Integer value representing the total count of business days within the specified range, inclusive of both the start and end dates if they fall on a weekday. This count provides immediate, actionable data for calculating elapsed time in business terms, which is invaluable for automating complex operational reports.

Initial Implementation Example in VBA

To demonstrate the method's fundamental usage within VBA, we often employ a loop structure to process multiple sets of data efficiently. The following code snippet illustrates how to access the **NetworkDays** method via the WorksheetFunction property to calculate the working day difference across a specified range of rows.

This particular code block defines a simple macro that iterates through rows 2 to 9, pulling the start and end dates from columns A and B, respectively, and placing the resulting NetworkDays calculation into column C.

Sub CalculateNetworkDays()

Dim i As Integer

For i = 2 To 9
Range("C" & i) = WorksheetFunction.NetworkDays(Range("A" & i), Range("B" & i))
Next i

End Sub

This implementation processes the start dates in the range **A2:A9** against the end dates in **B2:B9**. The calculated count of whole working days is then displayed automatically in the destination range **C2:C9**. This loop structure is robust and highly reusable for automating tasks involving date span calculations within tabular data.

Syntax and Parameters of the WorksheetFunction Method

In VBA, all standard Excel functions are accessed using the WorksheetFunction property. This property acts as a gateway, allowing your code to invoke the same mathematical engine used directly within Excel cell formulas. The general syntax for calling the **NetworkDays** function is as follows, where the square brackets denote optional arguments:

```
WorksheetFunction.NetworkDays(StartDate, EndDate, )
```

The parameters required for execution are clearly defined. **StartDate** and **EndDate** are mandatory

and must be valid date values. These dates can be supplied by referencing cells in the worksheet, using date variables, or passing literal date constants (though referencing cells is most common in loop structures). The order is important: if the StartDate is chronologically later than the EndDate, the function will return a negative Integer value, indicating the number of working days counting backward. The final, optional parameter, **Holidays**, is crucial for real-world business calculations.

The **Holidays** argument allows the developer to specify a range of cells containing dates that should also be excluded from the count, in addition to the standard Saturday/Sunday exclusion. This range must contain valid date serial numbers or date objects. If this parameter is omitted, the calculation relies solely on the default weekend exclusion. Proper definition of this holiday range ensures that the calculated number of working days is accurate according to your organization's specific calendar, making the results suitable for official time tracking and reporting purposes.

Example: Automating NetworkDays Calculations

Setting Up the Sample Data

To fully appreciate the utility of this macro, consider a common scenario where we are tracking the duration of several projects. Suppose we have the following list of start dates and end dates stored in Columns A and B of our Excel worksheet, beginning in Row 2:

	A	B	C	D	E
1	**Start Date**	**End Date**			
2	1/2/2023	1/3/2023			
3	1/5/2023	1/8/2023			
4	1/10/2023	1/20/2023			
5	2/1/2023	2/16/2023			
6	3/10/2023	4/19/2023			
7	4/15/2023	6/17/2023			
8	5/1/2023	6/18/2023			
9	6/28/2023	12/30/2023			
10					
11					
12					
13					
14					
15					
16					
17					

Our goal is to use the **NetworkDays** method, accessed through VBA, to programmatically calculate the number of whole working days between the start and end dates for every row shown above. This avoids repetitive manual formula entry and ensures the calculation is executed consistently across the entire data set.

We can achieve this by creating and executing the following macro in the VBA editor, which implements the iteration logic discussed previously:

Executing the Calculation Macro

The following code defines the procedure. Note that this specific implementation does not include the optional holiday argument, meaning it will only exclude Saturdays and Sundays from the count. If your project required holiday exclusion, you would need to add a third argument referencing a range containing your official holiday dates.

Sub CalculateNetworkDays()

Dim i As Integer

For i = 2 To 9

```
Range("C" & i) = WorksheetFunction.NetworkDays(Range("A" & i), Range("B" & i))
Next i

End Sub
```

When this macro runs, the loop iterates eight times (from i=2 to i=9). In each iteration, the WorksheetFunction object calls the NetworkDays method, retrieving the start and end dates dynamically based on the current row index `i`, and assigning the resulting Integer count to Column C.

Analyzing the Calculated Results

Once the macro has completed its execution, we receive the following output, with Column C now populated with the results of the calculation:

	A	B	C	D	E
1	**Start Date**	**End Date**			
2	1/2/2023	1/3/2023	2		
3	1/5/2023	1/8/2023	2		
4	1/10/2023	1/20/2023	9		
5	2/1/2023	2/16/2023	12		
6	3/10/2023	4/19/2023	29		
7	4/15/2023	6/17/2023	45		
8	5/1/2023	6/18/2023	35		
9	6/28/2023	12/30/2023	133		
10					
11					
12					
13					
14					
15					
16					
17					
18					

Column C shows the precise number of whole working days between the start and end dates in each row. Let us verify the logic of the output for specific entries to ensure full understanding of how the NetworkDays function handles different time spans:

The number of working days between 1/2/2023 and 1/3/2023 is **2**. This is correct as both dates fall

on weekdays (Monday and Tuesday) and are inclusive in the count.

The number of working days between 1/5/2023 and 1/8/2023 is **2**. This period spans Thursday, Friday, Saturday, and Sunday. Only Thursday and Friday are counted.

The number of working days between 1/10/2023 and 1/20/2023 is **9**. This period contains ten calendar days, excluding one weekend (four days total), resulting in the net count of nine business days.

This successful demonstration confirms that integrating `WorksheetFunction.NetworkDays` into a VBA loop provides an effective, high-performance solution for mass date calculation, adhering perfectly to standard business logic.

Conclusion: Leveraging NetworkDays for Robust Automation

The **NetworkDays** method, accessed through the WorksheetFunction object, stands as a fundamental tool in the VBA developer's arsenal for date manipulation. By seamlessly integrating the power of Excel's built-in date calculations directly into your macro code, you can significantly reduce development time and enhance the accuracy of time-sensitive calculations required for project management, finance, and operational reporting.

Remember that for non-standard weekend schedules or more complex localization needs, the related function **NetworkDays.Intl** offers additional flexibility by allowing you to specify exactly which days constitute the weekend. However, for most organizational needs operating on a Monday-to-Friday schedule, `WorksheetFunction.NetworkDays` remains the cleanest and most efficient choice.

Note: You can find the complete documentation for the **NetworkDays** method in VBA on the official Microsoft Developer Network (MSDN) website, offering comprehensive details on syntax and usage.