

## CreateDIBSection

The **CreateDIBSection** function creates a device-independent bitmap (DIB) that applications can write to directly. The function gives you a pointer to the location of the bitmap's bit values. You can supply a handle to a file mapping object that the function will use to create the bitmap, or you can let the operating system allocate the memory for the bitmap.

```
HBITMAP CreateDIBSection(  
    HDC hdc,                // handle of device context  
    CONST BITMAPINFO *p bmi,    // address of structure containing bitmap size, format, and color data  
    UINT iUsage,                // color data type indicator: RGB values or palette indices  
    VOID **ppvBits,            // pointer to variable to receive a pointer to the bitmap's bit values  
    HANDLE hSection,           // optional handle to a file mapping object  
    DWORD dwOffset           // offset to the bitmap bit values within the file mapping object  
);
```

### Parameters

*hdc*

Handle to a [device context](#). If the value of *iUsage* is DIB\_PAL\_COLORS, the function uses this device context's [logical palette](#) to initialize the device-independent bitmap's [colors](#).

*p bmi*

Points to a [BITMAPINFO](#) structure that specifies various attributes of the device-independent bitmap, including the bitmap's dimensions and colors.

*iUsage*

Specifies the type of data contained in the **bmiColors** array member of the **BITMAPINFO** structure pointed to by *p bmi*: logical palette indices or literal RGB values. The following values are defined:

| Value          | Meaning  |
|----------------|--|
| DIB_PAL_COLORS | The <b>bmiColors</b> member is an array of 16-bit indices into the logical palette of the device context specified by <i>hdc</i> . |
| DIB_RGB_COLORS | The <b>BITMAPINFO</b> structure contains an array of literal RGB values.   |

*ppvBits*

Points to a variable that receives a pointer to the location of the device-independent bitmap's bit values.

*hSection*

Handle to a file mapping object that the function will use to create the device-independent bitmap. This parameter can be NULL.

If *hSection* is not NULL, it must be a handle to a file mapping object created by calling the [CreateFileMapping](#) function. Handles created by other means will cause **CreateDIBSection** to fail.

If *hSection* is not NULL, the **CreateDIBSection** function locates the bitmap's bit values at offset *dwOffset* in the file mapping object referred to by *hSection*. An application can later retrieve the *hSection* handle by calling the [GetObject](#) function with the [HBITMAP](#) returned by **CreateDIBSection**.

If *hSection* is NULL, the operating system allocates memory for the device-independent bitmap. In this case, the **CreateDIBSection** function ignores the *dwOffset* parameter. An application cannot later obtain a handle to this memory: the **dshSection** member of the [DIBSECTION](#) structure filled in by calling the [GetObject](#) function will be NULL.

*dwOffset*

Specifies the offset from the beginning of the file mapping object referenced by *hSection* where storage for the bitmap's bit values is to begin. This value is ignored if *hSection* is NULL. The bitmap's bit values are aligned on doubleword boundaries, so *dwOffset* must be a multiple

The bitmap's bit values are aligned on doubleword boundaries, so *dwOffset* must be a multiple of the size of a **DWORD**.

### Return Value

If the function succeeds, the return value is a handle to the newly created device-independent bitmap, and *\*ppvBits* points to the bitmap's bit values.

If the function fails, the return value is NULL, and *\*ppvBits* is NULL. To get extended error information, call [GetLastError](#).

### Remarks

As noted above, if *hSection* is NULL, the operating system allocates memory for the device-independent bitmap. The operating system closes the handle to that memory when you later delete the device-independent bitmap by calling the **DeleteObject** function. If *hSection* is not NULL, you must close the *hSection* memory handle yourself after calling **DeleteObject** to delete the bitmap.

**Windows NT:** You need to guarantee that the **GDI** subsystem has completed any drawing to a bitmap created by **CreateDIBSection** before you draw to the bitmap yourself. Access to the bitmap must be synchronized. Do this by calling the **GdiFlush** function. This applies to any use of the pointer to the bitmap's bit values, including passing the pointer in calls to functions such as **SetDIBits**.