Upgrading Imaging Tools SDK from version 11 or older to 12.00

Before installing the latest version of Imaging Tools SDK, please uninstall the previous version from your system.

Important note: Every file and folder related to Imaging Tools SDK installation will be deleted during the uninstall process. If there were some custom files in the regular Imaging Tools SDK directories, they will be deleted as well.

Upgrading redistribution files

Notice: All 'Redistribution Files' usage is governed by the Black Ice Software license agreement.

Important notes when upgrading distributions from version 11.x or older to 12.x:

1. If your application is built using Visual Basic 6, don't upgrade to Imaging Tools SDK 12.00 or above. There are 64 bit integers in the OCX methods' parameter list and Visual Basic 6 does not support 64 bit integers.
   This does not apply to VB.NET applications. If your application is built using VB.NET, upgrading to version 6 or above will bring many benefits, especially Unicode file name support and 64 bit support.

2. If your application is built in Delphi, upgrading from Imaging Tools SDK 11.05 is only recommended if using Delphi 12 or higher.
   Imaging Tools SDK 12.00 includes UNICODE file name support for the BiDIB and BiTIFF functions, however full UNICODE support in Delphi became available only from version 12.

3. When evaluating the latest version of Imaging Tools SDK please install it on a separate machine in order to avoid file conflicts. The latest version cannot be installed 'next' to the previous version.

4. To upgrade redistribution files, follow the guidelines of your development environment and programming language, in many cases just simply overwrite the previous version of the redistribution files with the latest versions and rebuild your application.

5. In version 12.0 the parameter list of many of the Imaging Tools related functions and methods have been changed. Please update your application if needed.

6. If you are using a function that has string parameters, and you are loading the DLLs dynamically into your C/C++ application, then please indicate whether the UNICODE or the ANSI function should be used from the BiDIB and BITIFF dlls. Following the Microsoft conventions, functions supporting UNICODE parameters end with 'W', those supporting 'ANSI' end with 'A'.

When loading the dll statically into your C/C++ application using the header files, a macro in the header file will automatically select for you the UNICODE function if the 'UNICODE' macro is defined in your project.