Omnis Studio

Runtime Application Deployment

Table Of Contents

1. Aim

The aim of this document is to give a basic outline of how to deploy your Omnis Studio application. The process will begin by outlining the files required in the Omnis Studio runtime and will then go on to explain where to place your libraries and how to create an installer using BitRock InstallBuilder or Installer Vise from Mindvision.

Note: Members of the **Omnis Developer Partner Program** (ODPP) can obtain an installer template from us to allow you to build installers using BitRock InstallBuilder.

1.1 Warning

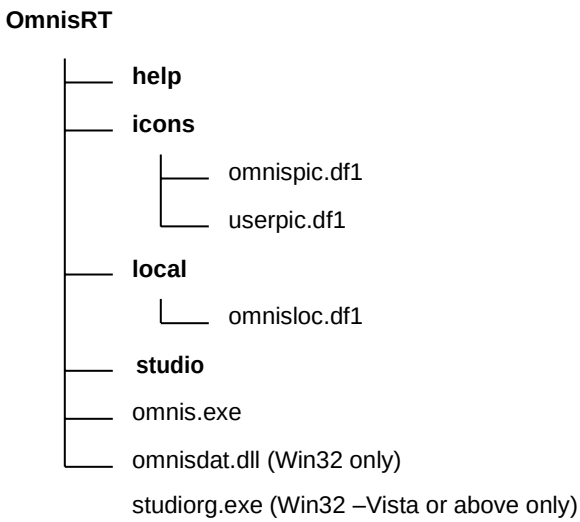
We do not recommend that any files should be removed from the runtime tree unless absolutely necessary. Furthermore, as disk space, memory, or bandwidth are no longer issues for a majority of end users, we recommend that you do not need to remove any files from the Omnis Runtime tree to reduce the size of the installer files. Therefore, the following information about files in the Runtime tree is for your information only, and only offers a guide to the files in Omnis Studio.

2. Omnis Studio Runtime Files

2.1 Basic Tree

The minimum required files for an Omnis Runtime environment to operate is shown in Figure 1, however only a very small simple application would be able to run in such circumstances. The more complex the application the more files will need to be in the tree.

Figure 1.



2.1.1 The 'AppData' Split

From Windows 2008/Vista onwards, Microsoft employs the UAC security system. Consequently, the *Program Files* directory is not writable without Administrator privileges. In order for Omnis Studio to run under this system, all of the files which the user may need to write to are moved out of Program Files, to the current user's AppData section.

If the Studio install directory contains a *'firstruninstall'* folder, its contents will be copied to the user's appropriate AppData section when Studio is launched, if there is not already a directory there of the same name.

Omnis will expect its writable section to be located in different locations, depending on the install path of the main Omnis tree.

Install Path	Expected AppData Path
C:\Program Files\Folder1\Folder2\FinalFolder	**C:\Users\<Current User>\AppData\local\Folder1\Folder2\FinalFolder**
C:\Not Program Files\Folder1\Folder2\FinalFolder	**C:\Users\<Current User>\AppData\local\TigerLogic\FinalFolder**

2.2 System Files

There are some files that do not appear in the tree but are installed into the user's system folder. These files are listed below in Figure 2. If these files are not installed on the user's system folder then the application may not work as expected.

Figure 2.

Windows 32bit
mfc40.dll
mfc42.dll
msvcrt.dll
msvcrt20.dll
msvcrt40.dll
olepro32.dll

2.1.2 Microsoft Visual C++ components

If you are building Windows installers you will need to include the "Microsoft Visual C++ Redistributable Package": the version of this package will depend on the version of Omnis Studio you are using and the target Windows operating system.

Figure 2a.

Omnis Studio version	Version of Visual C++ package/Windows
Omnis Studio 5.0.x to 6.0.x (Win Vista, Win 7)	Microsoft Visual C++ 2008 Redistributable Package
Omnis Studio 6.1.x (Win Vista, Win 7) to 8.0.x (Windows 8 or later)	Microsoft Visual C++ 2013 Redistributable Package
Omnis Studio 8.1.x to Omnis Studio 10.1.x (Windows 8 or later)	Microsoft Visual C++ 2015 Redistributable Package

Check the Microsoft website for further information about this package and to download the version you need.

2.2 Additional Files

This section explains why each file is in the Omnis Runtime tree and in what circumstances it can be removed from the final master tree.

2.2.1 Externals

These files exist in the external folder of the Omnis Runtime Tree and need to be present if any of the function calls, described below in Figure 3, are made in your application.

Figure 3.

Filename	Function Calls	Notes
msmail2.dll	MSMail commands, e.g. MSM Dispose message, MSM Delete message, MSM Save enclosure, etc.	Win32 only
wbinfile.dll	Binary functions, ReadBinFile, WriteBinFile	
wecommnd.dll	Web commands & functions, e.g. FTP, HTTP, etc	

Filename	Function Calls	Notes
weshared.dll	Web function calls	This can be found in the root of the runtime application tree
xcalldll.dll	Call DLL, Register DLL.	Not macOS
xcharfnc.dll	omniscode, oemchar, oemcode, ansichar, ansicode.	Not macOS (not present in later versions of Studio)
xdatefnc.dll	isoweek, setfye, getfye, setws, getws, fday, lday, pday, nday, dpart, dname, ddiff, dadd.	
xevmouse.dll	evMouse	Win32 only
xfileops.dll	FileOps functions: Does file exist, Get file name, Put file name, Create file, Move file, Copy file, and so on.	Must be included if using the Adhoc Reports library.
xfontfnc.dll	isfontinstalled, textsize, fontlist.	Must be included if using the Adhoc Reports library.
xlistfnc.dll	minc, maxc, stddevc, avgc.	
xrandom.dll	shufflelist, rollstring, rolldice, setseed, getseed, randrealrng, randintrng, rand.	
xstrfnc.dll	strtok, strpbrk, strspn, isnumber.	
xstrip.dll	Strip.	

If any of the function calls listed above are made without the relevant file in the externals folder then the application will not work correctly. In addition, if the Adhoc Reports library is used in your runtime application then the xfileops.dll and the xfontops.dll must be present as calls are made to both of these files from within the library. The macOS external files will not have the .dll extension.

2.2.2 DAMs

The older style (V2) DAMs files can be found in the externals folder, in **non-unicode** versions of Studio only. The object (V3) DAMs files can be found in the xcomp folder. These files are used when interacting with a database using SQL. If any of these databases, listed in Figure 4, are used in your application then the relevant files should be enclosed in the externals or xcomp folder.

Figure 4.

Older style(V2)	Object (V3)	Database	Notes
domnis.dll	damomsql.dll	Omnis Datafile	Only needed when interacting with an Omnis datafile using OmnisSQL
ddb2.dll	damdb2.dll	DB2 UDB	Older style - Win32 and Linux Only Object – Not macOS
doracle.dll	damora7.dll	Oracle 7	Older style - Win32 and Linux Only
doracle8.dll	damora8.dll	Oracle 8/9/10	Older style - Win32 and Linux Only
dinformx.dll	damifx.dll	Informix	Older Style - Win32 Only Object – Win32 and Linux Only
dodbc.dll	damodbc.dll	Any ODBC driver	Older Style - Not Solaris
dsybase.dll	damsybse.dll	Sybase11/12	Older Style - Win32 and Linux Only Object – Not Solaris

Older style(V2)	Object (V3)	Database	Notes
	damjdbc.dll	Any JDBC driver	
	dammysql.dll	MySql	
	damazon.dll	Amazon Simple DB	
	damobase.dll	OpenBase	
	dampgsql.dll	PostgreSQL	
	damsybse.dll	Sybase	
	frontbasedam.dll	FrontBase	Provided by FrontBase. http://www.frontbase.com
enablea.dll		Db2 UDB	Needed if using DB2 audio extenders. Win32 only
enablev.dll		Db2 UDB	Needed if using DB2 video extenders. Win32 only
enablei.dll		Db2 UDB	Needed if using DB2 image extenders. Win32 only
upload.bnd		Db2 UDB	Needed if using DB2 Extenders. Win32 only
upload.dll		Db2 UDB	Needed if using DB2 Extenders. Win32 only

Your application will not be able to access any of the databases or database functionality without the correct DAM. Please see our web site for information about clientware versions. The macOS DAMs will not have the .dll extension. The Unix DAMs will have the .so extension.

2.2.3 Localisation files

These files can be found in the local folder. Although the omnisloc.df1 is an essential file, the additional files in this folder may be removed if necessary. The additional files are listed below, in Figure 6, with a description of when they will be needed.

Figure 6.

Filename	Notes
client.stb	This is a string table, which contains language-specific translations used by the **web client**. By default it is empty, but you can add translations during development. **Studio 5 and later only.**
localib.pdf	This Adobe Acrobat file contains instructions on how to translate your libraries into another language. This should be included if you intend your user to be able to translate the application into another language (not available for later versions of Studio).
locomnis.pdf	This Adobe Acrobat file contains instructions on how to localise Omnis. This file should be included if you wish provide a mechanism to configure the following Omnis internal items which are stored in the omnisloc.df1: The names of the days of the week, The names of the months of the year, Separator characters, The text for Yes/No, OK/Cancel, True/False, Am/Pm and On/Off, The national sort ordering.
omnisloc.df1	This Omnis datafile stores language and separator settings etc for each language, as set using omnisloc.lbs.
omnisloc.lbs	This Omnis library file provides the user with an interface to change the internal settings stored in the omnisloc.df1 file. This library only needs the omnisloc.df1.
studio.stb	This is a string table, which is used in the same way as client.stb, but dictates strings used in the Omnis Studio environment rather than the web client. **Studio 5 and later only.**
trans.lbs	This Omnis library file provides the user with a tool for translating their library or Omnis itself into another language. **Not included in Studio 5 and later.**
xlate.lbs	This Omnis library contains string translation functions, which uses Google Translate. It will not be needed by the user by itself, but you may wish to call into it from your own libraries. **Included in Studio 5 and later, but not later versions.**

2.2.4 Adhoc library

The adhoc.lbs reports library can be found in the adhoc folder of a development version of Studio. This Omnis library allows users to create their own reports from their data.

If you wish to include this for your runtime users, you should copy adhoc.lbs from the development version, into the **startup** folder of the runtime. This library makes calls to functions contained in the xfontops.dll and xfileops.dll. Therefore both of these external files will need to be included in the external folder in order for the Adhoc reports library to function correctly.

2.2.5 Character Mapping

There is a file called charmap.exe that can be found within the studio folder of the runtime application tree. This file is used for creating character-mapping files that are used when interacting with databases that have a different character set to the one being used by the application. This file should not be removed if you expect the user to be able to create character-mapping files.

2.2.6 Java

The files to enable Java functionality are contained within a java folder that is in the xcomp folder in the root of the runtime application tree. If your application uses any Java functionality (including web services) this folder should remain within the xcomp folder with all files intact.

Earlier versions of Studio allowed use of javabean components: if you are using this feature within your Omnis Application then the user must also have the Java development environment installed on their machine.

2.2.7 External Components

The external components are objects used within Omnis Studio to enhance your applications. The files that represent the components can be found in the xcomp folder. These files need to be present in your application tree if the relevant component, listed below in Figure 7, is used.

Figure 7.

Filename	Component	Notes
accord.dll	Accordion component	
blowfish.dll	Blowfish component non-visual component	Enables encryption and decryption.
calendar.dll	Calendar component	
cooldrop.dll	Cool Drop-list component	Used internally only
docview.dll	Document Viewer component	Not in later versions of Studio
fadepict.dll	Fade Picture component	
filelist.dll	File List component	
fileops.dll	File operations non-visual component	
filterls.dll	Filter List component	
filters.dll	Image manipulation filter component	Available as part of the Icon Editor.
flic.dll	FLC image component	Not in later versions of Studio
fontops.dll	Font operations non-visual component	
formroll.dll	Roll Button component	
gif.dll	GIF image component	Not in later versions of Studio
graphs.dll	Graphs component	Needs to have the dgdsc32.dll in the root of the runtime application tree in order to function. Not Solaris.
graphs2.dll	Graphs2 Component	Updated graph component. Requires chartdir41.dll to be in the root of the Omnis tree for Win32, or libchartdir.so.4.1 for Linux.
helptls.dll	Help Utilities component	
hotpict.dll	Hot Picture component	
html.dll	HTML component	Needed if printing reports to HTML device.
hyplinks.dll	Hyperlinks component	
icnarray.dll	Icon Array component	
iconedit.dll	Icon Editor component	
javabean.dll	Java bean component	Needs the Java folder and files within the xcomp folder. Win32 only.
javacore.dll	Omnis Java Engine	
javaobjs.dll	Java Objects component	
jpeg.dll	JPEG image component	Not in later versions of Studio
marquee.dll	Marquee component	
mciplay.dll	Multimedia Player component	Win32 only; not in later versions of Studio
npapi.dll	NPAPI component	Not in later versions of Studio
oclock.dll	Clock component	

Filename	Component	Notes
ocxhndlr.dll	ActiveX handling component	Needs to be included if utilising the Omnis Help facility in your runtime application. Win32 only.
ole2auto.dll	Automation Library	Win32 only.
omdnobjs.dll	Dot Net Objects Library	
omnisicn.dll	Omnis icon component	
omnisqt3.dll	QuickTime3 component	Win32 and macOS only.
oole2.dll	Omnis OLE component	Needs to be used with omni2ui.dll in the root of the application tree. Win32 only
otimer.dll	Timer component	
oxml.dll	XML component	
pagecnt.dll	Report Object component	
pcx.dll	PCX image component	Not in later versions of Studio
piclist.dll	Picture List component	
portprof.dll	Port Profile component	
postpst.dll	Postscript Device	
progress.dll	Progress bar component	
regadmin.dll	RegAdmin component	Not supported but remains in tree for backwards compatibility
replists.dll	Report Lists component	
rtf.dll	RTF device	Needs to be included if printing reports to RTF device.
sidebar.dll	Side bar component	
slider.dll	Slider component	
ssgrid.dll	Spreadsheet component	
stix.dll	Stix component	Not in later versions of Studio
strtable.dll	String table component	
tabbar.dll	Tab bar component	
tile.dll	Tile component	
toolpal.dll	Icon Drawing Tools component	Available as part of the Icon Editor.
tooltip.dll	Tool tip component	
tranbutt.dll	Trans Button component	
trayobj.dll	Tray Object component	Win32 only.
treectrl.dll	Tree component	
wash.dll	Wash component	
wavplay.dll	WAV sound file player	Not in later versions of Studio
wbmp.dll	WBMP component	Not in later versions of Studio
webwin32.dll	HTTPS component	Win32 only.
zoom.dll	Zoom component	

If any of the components properties, methods or events listed above are called without the relevant file in the xcomp folder then the application will not work correctly. The ocxhndlr.dll needs to be in the xcomp folder if the Omnis Help utility is used and when printing reports to either HTML or RTF devices, then the relevant file needs to be included.

2.2.8 Miscellaneous Files.

These remaining files appear in the root of the Omnis runtime tree. These files have different functionality and should not be removed unless the specified functionality is not needed. These files are shown in Figure 8.

Figure 8.

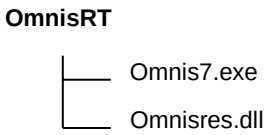
Filename	Functionality	Notes
charmap.ini	Character Map for conversion	Win32 only.
chartdir41/51.dll (Win32) ibchartdir.so.4.1 (Linux)	ChartDirector Graph API	Required for Graphs2 xcomp. It also needs the graphs2.dll file in the xcomp folder
dgdsc32.dll	Graphs API	Required for graphing functionality. It also needs the graphs.dll file in the xcomp folder. Win32 only
iconv.dll	PostgreSQL support library	Required for different languages and character sets
icudt34.dll	International Components for Unicode	Provides Unicode support in Omnis
icuin34.dll	International Components for Unicode	Provides Unicode support in Omnis
icuuc34.dll	International Components for Unicode	Provides Unicode support in Omnis
jikes.exe	Java compiler	Needed for Web Services and Java objects support
libcurl.dll	URL transfer utility	Required for PostgreSQL support
libpq.dll	PostgreSQL Client library	Required for PostgreSQL support
libxml2.dll	XML utility	Required for PostgreSQL support
license.html	License text	Remove for your deployment
msvcr71.dll	Microsoft C Runtime Library File	Win32 only
msvcr90.dll	Microsoft C Runtime Library File	Win32 only
msvcrt.dll	Microsoft Visual C++ Runtime Library	Win32 only
omdotnet.dll	Omnis .NET Objects component	Win32 only. This must be registered with .NET.
omni2ui.dll	Omnis OLE	These files should be used in conjunction with oole2.dll in the xcomp folder.
studiomsgs.dll	Message definitions for event viewer	Win32 only
studiorg.exe	Writes the registry entries required for Omnis Studio	Only required for platforms using UAC (i.e. Windows Vista or later)
wesecure.dll	Secure connection support for Web Enabler commands	Provides a layer between weshared and OpenSSL
weshared.dll	Web Enabler commands	Required to run wecommnd.dll
xerces-c_2_1_0.dll	Xerces Library used by OXML	libxerces-c.so.21 on Unix.
zlib1.dll	ZLIB compression utility	Required for PostgreSQL support

3. Omnis 7 Runtime Files

3.1 Basic Tree

The minimum required files for an Omnis Runtime environment to operate is shown in Figure 9, however only a very small simple application would be able to run in such circumstances. The more complex the application the more files will need to be in the tree.

Figure 9.



3.2 System Files

There are some files that do not appear in the tree but are installed into the user's system folder. These files are listed below in Figure 10. If these files are not installed on the user's system folder then the application may not work as expected.

Figure 10.

Windows 32bit
Omni2ui.dll
Msvcrt.dll
Msvcrt20.dll
Olepro32.dll

3.3 Additional Files

This section will explain why each file is in the Omnis Runtime tree and in what circumstances it can be removed from the final master tree.

3.3.1 Externals

These files exist in the External Folder of the Omnis Runtime Tree on Windows platforms or the Omnis Extensions folder on the Macintosh platforms and need to be present if any of the function calls, described below in Figure 11, are made in your application.

Figure 11.

Filename	Function Calls	Notes
Calldll.dll	Call DLL, Register DLL.	Windows only
Charfnc.dll	ansiflag, ansicode, ansichar, omniscode, oemchar, oemcode, omnischar.	Windows only
Datefnc.dll	isoweek, setfye, getfye, setws, getws, fday, lday, pday, nday, dpart, dname, ddiff, dadd.	
Fileops.dll	DoesFileExist, GetFile, PutFile, CreateFile, MoveFile, CopyFile, DeleteFile, OpenFile, CloseFile, etc.	
Fontfnc.dll	isfontinstalled, textsize, fontlist.	

Filename	Function Calls	Notes
Listfnc.dll	minc, maxc, stddevc, avgc.	
Msmail2.dll	MSMAIL_DISPOSE_MESSAGE, MSMAIL_DELETE_MESSAGE, etc.	
Random.dll	Shufflelist, rollstring, rolldice, setseed, getseed, randrealrng, randintrng, rand.	
Coolbtn.dll	Play Button Disable, Coolbtn, Play Button Enable	
Env00202.dll	ENV002D, ENV002C, ENV002B, ENV002A.	
Getfiles.dll	GetFiles.	
Graphs.dll	Get selected object, getgraph, Set graph attribute, Get graph attribute, Graph.	
Helpcom.dll	Help topic, Help by Key, Help Search.	
Notes.dll	NSF Mail Note, NSF Write composite, NSF Who am I, NSF Close all files, NSF Where's my mail?, etc.	
QT.dll	Quicktime	
Replace.dll	Replace	
PE	PE, Quicktime	Mac Only
Callxcmd	Call XCMD.	Mac Only

If any of the function calls listed above are made without the relevant file in the Externals folder then the application will not work correctly. The Mac External files will not have the .dll extension.

3.3.2 Dams

The Dams files can be found in the Externals folder. These files are used when interacting with a database using SQL. If any of these databases, listed in Figure 12, are used in your application then the relevant files should be enclosed in the Externals Folder.

Figure 12.

Filename	Database	Notes
Omnissql.dll	Omnis Datafile	Only needed when interacting with an Omnis datafile using OmnisSQL.
O7db2.dll	DB2 v5	Windows & PPC only
oracle.dll	Oracle7 or Oracle8	
O7inform.dll	Informix	Windows only
O7odbc.dll	Any via ODBC interface	
sybasedb.dll	Sybase10 or Sybase11	

Your application will not be able to access any of the databases or database functionality without the correct DAM. Please see our web site for information about client-ware versions. The Mac DAM"s will not have the .dll extension.

3.3.3 Web Extensions

The Web Extension files can be found in the External folder of your runtime application with the exception of the Weshared.dll file, which is found in the root of the runtime application tree. These files need to be present if any of the function calls, described below in Figure 13, are made in your application.

Figure 13.

Filename	Function Calls	Notes
Wbinfile.dll	ReadBinFile, WriteBinFile.	
Wcontmgr.dll	CMSP7Sattach, CMSP10Attach, CMSP7Attach, CMAttach, CMMGEnd, CMMGBegin, CMQuery, etc	
Wecommnd.dll	FTPDelete, FTPCwd, FTPConnect, FTPChmod, CGIEncode, CGIDecode, TCPGetMyAddr, etc	
Weshared.dll	This file should be included whenever any Web function calls are made.	This can be found in the root of the runtime application tree.
Wftpclnt.dll	FTPRename, FTPDisconnect, FTPPwd, FTPGetBinary, FTPPutBinary, FTPSetProgressProc, etc.	
Wgfx.dll	DisplayImage, getpicfromimage	
Omni2ole.dll	Omnis OLE	This file can be found in the External\Applets folder.

If any of the function calls listed above are made without the relevant file in the Externals folder or the Weshared.dll in the root of the runtime application folder then the application will not work correctly. The Mac Web Externals will not have the .dll extension.

3.3.4 SQL Utilities

There is a file called Charmap.exe that can be found within the SQLUTILS folder of the runtime application tree. This file is used for creating character-mapping files that are used when interacting with databases that have a different character set to the one being used by the application. This file should not be removed if you expect the user to be able to create character-mapping files. In addition to this application there is a Change SQLServe Size application on the Macintosh tree. This utility is used to change the amount of Omnis Ram that the SQLServer Dam uses.

3.3.5 Font Files

These are a set of cross platform fonts that are supplied with Omnis7 and can be found within the Fonts folder in the application tree. These fonts do not need to be in the Omnis tree but should be installed into the users'

system fonts folder when installing your application if necessary. These will be found on Windows platforms only.

3.3.6 Unused Files

There are certain files supplied with the Omnis7 Runtime tree that are not used but are maintained for backward compatibility. These files can be found in the Unused folder in the application tree. These files will be old DAMS, Externals and the old PPC ObjectSupportLib.

3.3.7 Miscellaneous Files.

These remaining files appear in the root of the Omnis runtime tree. These files have different functionality and should not be removed unless the specified functionality is not needed. These files are shown in Figure 14.

Figure 14.

Filename	Functionality	Notes
Dgdsc32.dll /Graph Shared Library.	Graphs API	This file needs to be included in the tree if any graphs functionality is used within the application. It also needs the Graphs.dll file in the External folder.
Psapi.dll	Omnis command “Test for Program Open”	NT only
Adhoc.hlp	The help file for Omnis7 adhoc reports.	
O7tk16.dll	Allows 16bit functions to run on 32bit platforms.	Win32only. Works in conjunction with O7tk32.dll.
O7tk32.dll	Allows 16bit functions to run on 32bit platforms.	Win32only. Works in conjunction with O7tk16.dll.
Omnispic.df1	The icons datafile for Omnis7.	
Template.rep	The template for Omnis7 adhoc reports.	

4. Adding the Application Files

In order for the application library to function in the Omnis runtime environment, it needs to be loaded. It can be loaded manually by the user who would select “Open” from the File menu and browse to the library file. This may be suitable for some applications that involve several unconnected libraries. The most common way is the automatic loading of the library by the Omnis runtime environment. In order to achieve this the initial library should be placed in the startup folder within the Omnis Studio runtime tree or the External/Extensions folder within the Omnis 7 runtime tree. All libraries that are placed in this folder are loaded automatically when the runtime environment is started. For applications with more than one library, it is not necessary to place them all in this folder, only those that you wish to be loaded on startup. Additional libraries and files relevant to your application can then be placed anywhere in the tree so long as your application knows where to find them.

5. Building the Installer

When installing a release version of a software product there are requirements that the end user would expect to see from a quality product installation. The first impressions that users will receive from a software product will be that of the product installer. The description of how to create the installer will give a basic outline.

We use an installer solution from BitRock (http://installbuilder.bitrock.com) to build our release products for Windows, Mac and Linux. We believe this is a suitable cross-platform option for developers to build their own product installers, in addition to the installers we have supported in previous versions of Omnis Studio.

In addition, the installer creation process is described using Installer Vise for Windows and macOS, which is available from Mindvision Software. An alternative for Windows is InstallShield. Information about his product can be found on www.flexerasoftware.com/products/installshield.htm.

5.1 Cross Platform: BitRock InstallBuilder

You can download a free evaluation version from their website which is fully-functional, but will print a small message on the first window of the installer saying that it was created with an evaluation version of InstallBuilder.

InstallBuilder allows a great deal of customisation, but this document will just concentrate on the minimum you need to know to get your own installer working. If you are interested in further customising your installation, the BitRock documentation should provide the information you need.

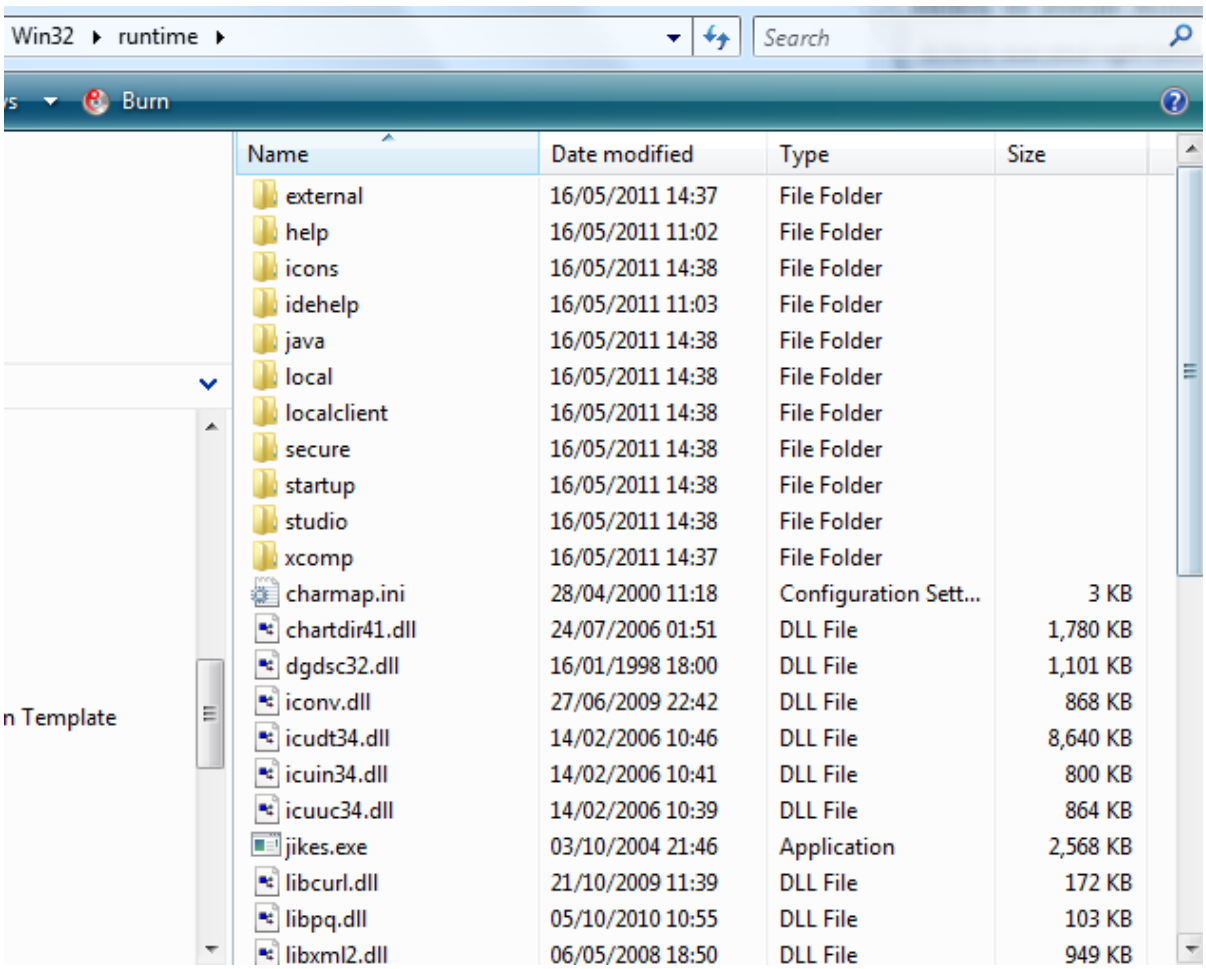
5.1.1 InstallBuilder template

To create your own Omnis-based product installer using BitRock InstallBuilder, you need to create a project with the necessary folder structure for the platforms you wish to support. Creating InstallBuilder projects and creating your own installers is described in detail in the BitRock documentation. The following section in this document refers to a project template that we have built to allow you to create Omnis-based product installers. Members of the **Omnis Developer Partner Program** (ODPP) can obtain the BitRock template via the usual Support channels. Developers not in the program will need to construct their own project based on those provided by BitRock.

5.1.2 Adding your files

1) Take the files (or .app for Mac) from a regular Omnis Studio Runtime installation and put them into the folder for the appropriate platform under (/<project directory>/files).

Note that if you are using Windows Vista or later, half of the Omnis tree will be installed to your local AppData folder, as UAC prevents write access to Program files. In this case you will need to copy both of these halves to the ‘…files/Win32/runtime’ folder to create a full tree.

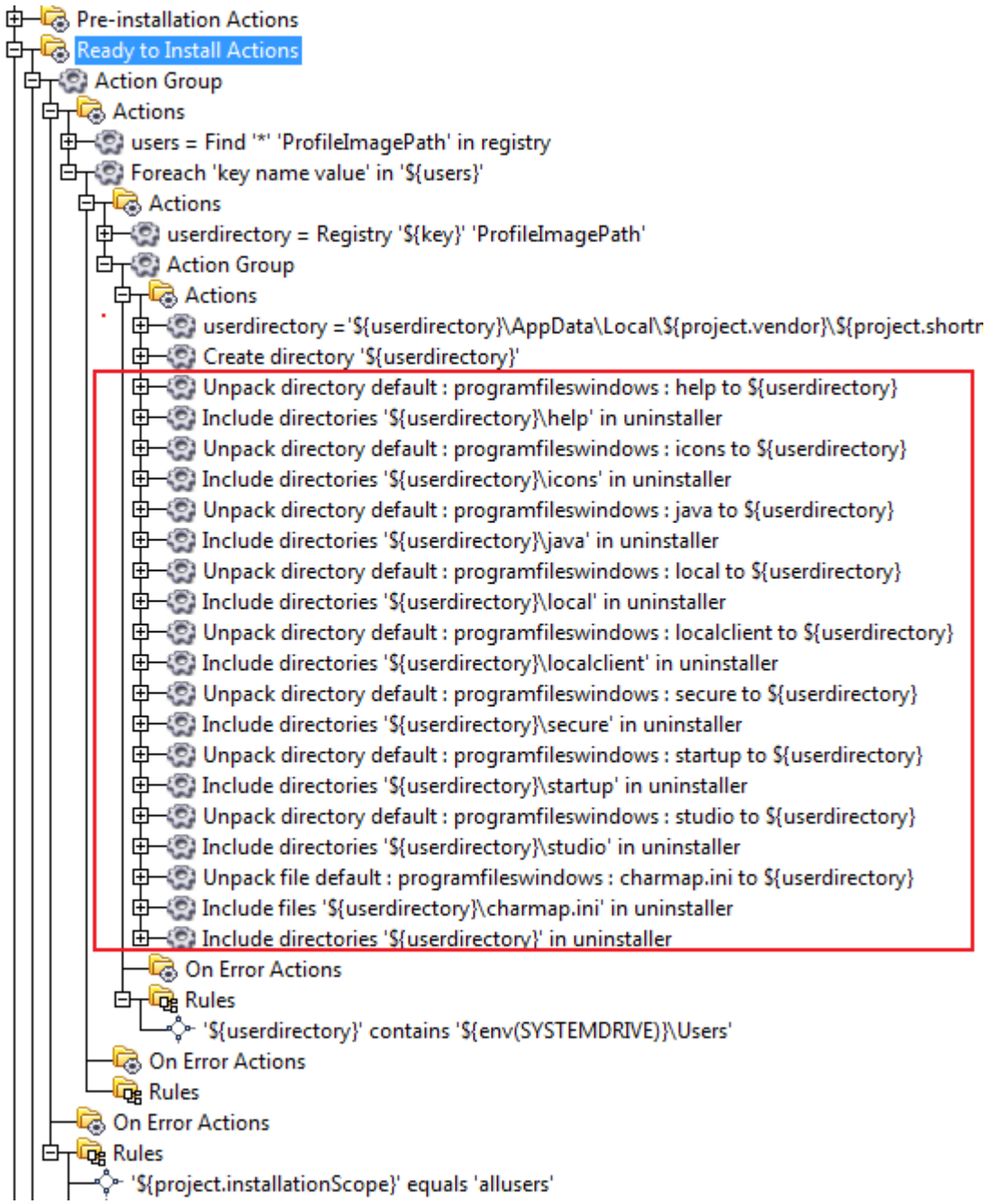


2) You should now add in your custom files to this tree.

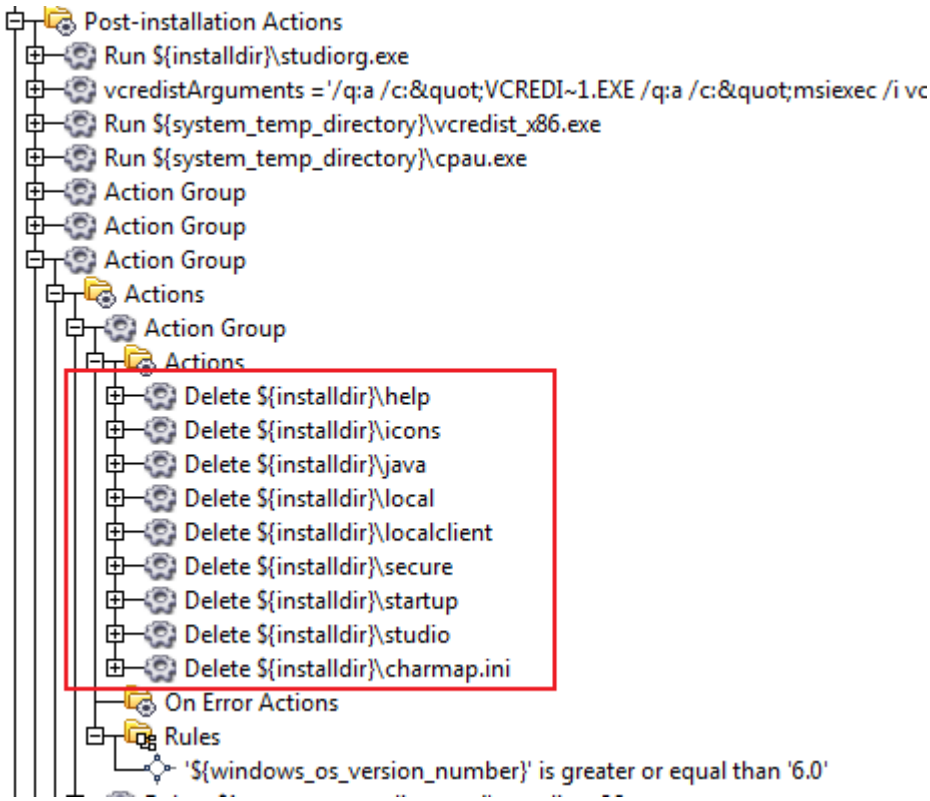
   For macOS installers, files need to be zipped up and placed into one file called OmnisRT.tbz. To do this, complete the application bundle Omnis.app then run the following command:

   tar cjvf OmnisRT.tbz Omnis.app

3) If you have extra files or directories to go into the root of the AppData folder (Windows Vista+ only), you"ll need to make a few changes to the project"s code:
   - Open the "*Advanced'* tab and find the "*Ready to Install Actions'* section.
   - This contains 2 "*Action Groups'* – the first is executed if an "*all users'* install is selected, the second is executed if a "*current user*" install is selected.
     Under both of these groups you will see that for each file/directory that goes into AppData there is an *'Unpack Directory/file…'* action to copy the directory/file to the AppData section, and an *'Include directories/files in uninstaller…'* to do just that.



- For each of your files/directories you wish to add to the AppData area, make your own versions of these actions, in both the "*All Users'* and *'Current User'* action groups.
- You should include a *'Post-install Actions*" section in your project. This can contain an action group which deletes each of the files/folders from the install directory, which were copied to AppData in the step above.
  You should add actions here for your extra files/directories which you have added to AppData.



5.1.3 Changing the Product name

Unless you want your installer and resulting application to be named *Omnis Studio Runtime,* you"ll need to change a few of the settings in your project.

1) Open *Product Details* from the sidebar and change the properties within appropriately. These properties are all self-explanatory, and also have a tooltip that gives an explanation of each.
2) Open *Customization* from the sidebar, and again just change the properties under the *User Interface* and *Installer* tabs to suit.
3) If you wish to edit the text shown on the "Install Complete" screen, you should edit *strings_en.lng* in the *files* folder using a text editor.

5.1.4 Changing the installer images

During installation it is possible to display a number of images or "billboards" that may contain information about your product. To change the images in your installer, open the *Advanced* section from the sidebar. At the end of the tree list, there is a *'Slideshow Images'* section – this lists the billboard images that will be cycled through while the files are installed. Add your own images here – they will be cycled through, each showing for 5 seconds.

5.1.5 Building the Installers

When you are ready to build, open "*Packaging'* from the sidebar, select the platform to build from the droplist, and push *'Build'.*

That"s it – you should now have a customised, fully working installer for Windows, Mac or Linux!

5.1.6 Explanation of Installation

The following is a brief explanation of the various stages in the installation process which may help you if you are intending to alter or extend the installer"s functionality yourself. Most actions and groups are commented – to see this, double-click on the action you"re interested in, go to the "*Display'* tab and you can see the "*Explanation'* there. Click on the edit button next to it to view the full text.

5.1.6.1 Pre-Installation Actions (Windows only)

The installer uses a “launcher” to raise privileges and preserve information from the old environment. The workflow is outlined below:

1. Launch the installer with requestedExecutionLevel = "asUser".
2. If the user does not have administrator permissions, unpack a launcher and call it with the environment variables to preserve and the path to the current installer (the one that is being executed).
3. Exit.
4. Now the launcher is executing and it requires UAC elevation. The user is requested for privileges elevation.
5. The launcher calls the original installer elevated, and passes it the environment information received.
6. Exit.
7. At this point, the main installer is called again with elevated privileges so it won't unpack or call the launcher, and now has received the necessary information to install the files into the relevant user’s AppData sections.

5.1.6.2 Ready to Install Actions (Windows only)

The installer installs the required files into the data directory
e.g. *C:\Users\<user>\AppData\Local\OS5.X*

5.1.6.3 Post-installation Actions

Windows

Studiorg.exe is run to set up the required registry entries.
The Microsoft Redistributable Package is installed.

macOS

The files are unzipped and the application bundle is renamed as appropriate.

Linux

If the installer is built on a Windows machine, all the files in the Linux tree require a chmod.
Omset is run.
Desktop shortcuts are setup using Xdg-Utils.

5.2 Windows: Installer Vise

Once you have built the library from the Omnis VCS and added it to the runtime tree along with any other files relevant to the application, you will need to create an installer to install the tree onto the user’s system. In this description of how to build the Windows Installer, we shall be using Installer Vise for Windows 3.5.1 from Mindvision Software www.mindvision.com.

At this stage, you should make a full check of the trees, comparing with a previous release in order to ensure no unwanted differences.

When building the installer project for the final release, it would be a wise policy to update the existing release project. The advantages of this are the consistency of the installer program from one release to another, the elimination of minor errors, which can cause constant rebuilding, and also the saving of a considerable amount of time. We will create an installer from scratch for the benefit of the document.

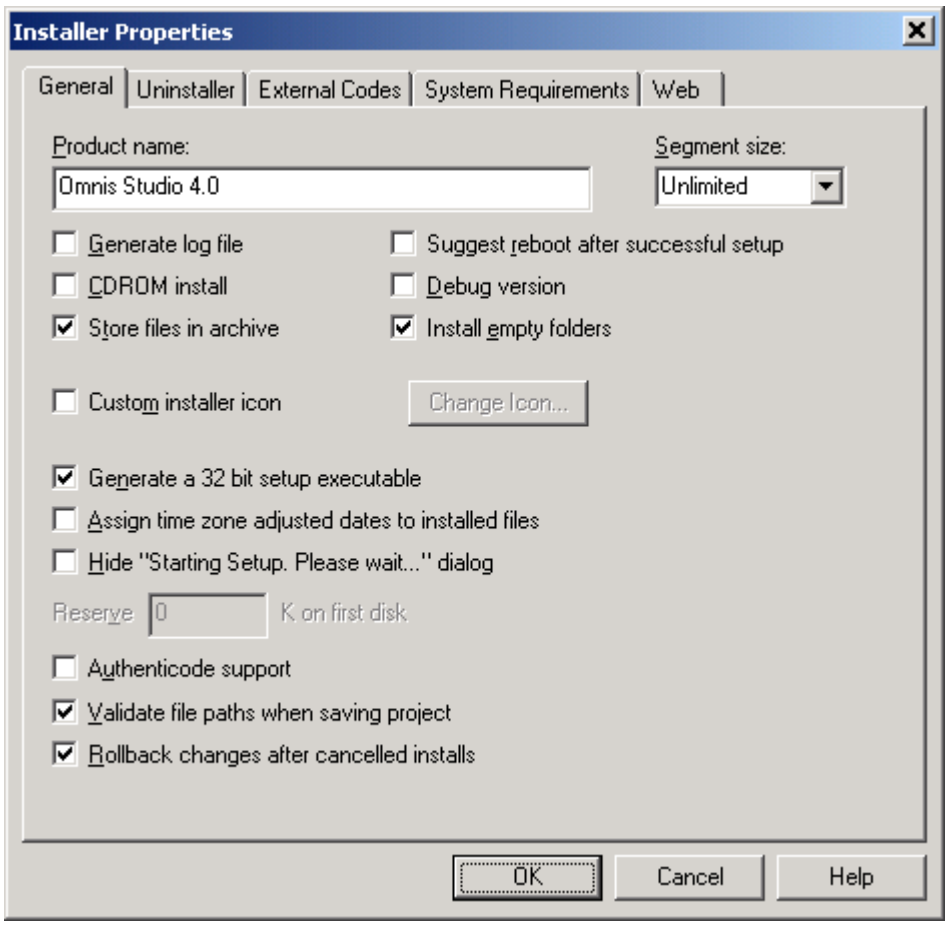
Start Installer Vise for Windows and a new project is opened automatically. Before you add the files to the project it is a good idea to add a comment stating the release name, the internal build number, date etc. This is useful for when you come to update projects at a later release. To add a comment select “Comment..” from the Add>>Items… menu.

Drag your entire tree onto the project window from your local drive. The project remembers where the files where copied from. This is helpful when you come to update the project as you can replace a single file in the tree or the entire tree and so long as it is in the same place as the previous release, the files are updated automatically. This means that it is a good idea to keep a tree structure available on your local drive for each project.

If you wish to display bitmaps during the installer then you will need to drag the files onto your project as support files. Create a folder called “support files” in the project and place all the bitmaps in this folder.

The next stage is to set up the Installer properties (Figure 15). From the File menu select Installer Properties... Type the name of the release in the Product Name entry field e.g. Omnis Studio 4.0. Ensure that the “Generate a 32 bit set-up executable” is checked for 32 bit platforms. Select the Uninstaller tab pane and check the “Include uninstaller”. The uninstaller properties have now been set and the product name is now stored in a variable called %ProductName%. This will be used later on.

Figure 15.



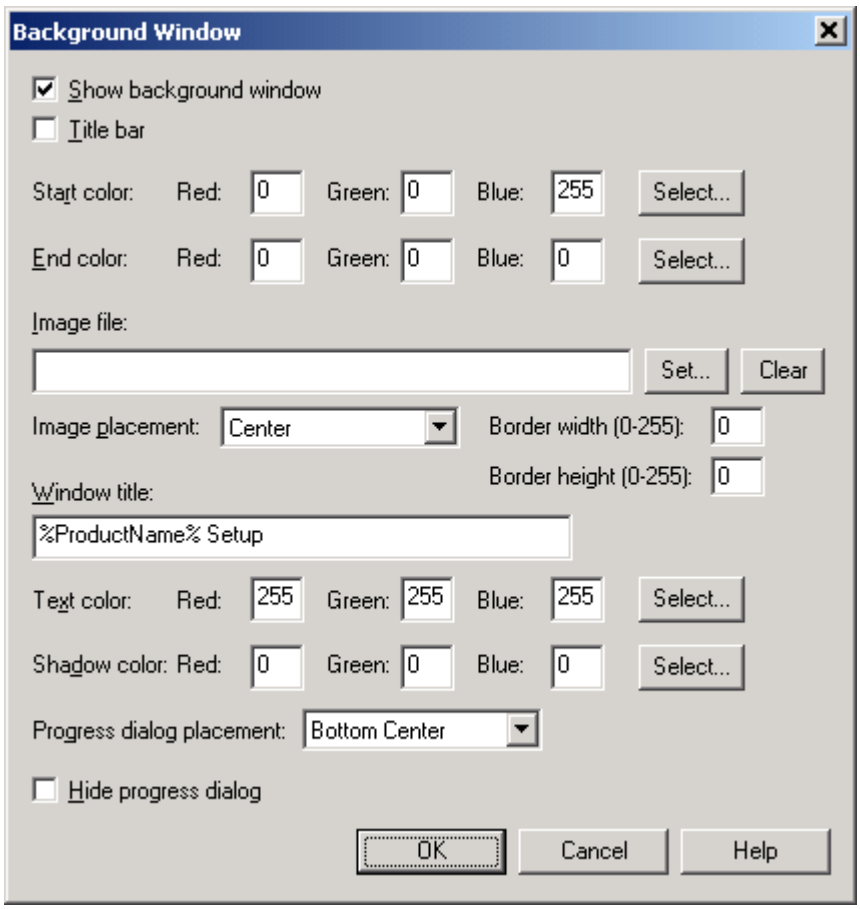
The next stage is to select the screens that will be displayed prior to the installation. The default screens that are usually displayed are the “Welcome Screen”, “Read Me Message”, “License Agreement”, “Select Install Directory”, “Select Program Folder” and “Finished Message 2”. The “Select Install Type” will only be used if the user has install options. We will deal with each screen in turn.

From the Screens menu, select “Background Window”. Modify the options on this window to give you the background window you require for your installer (Figure 16). Enter “%ProductName% Setup” in the window title entry field. As we are using the product name variable here, we will not need to keep updating it every time we update the installer. The Progress dialog placement should be set to “Bottom Center”.

Close the “Background Window” and from the Screens menu select “Welcome Screen”. Ensure that the “Show this dialog” option is checked. This option will need to be checked for each screen required in the installer. In each of these windows that you select you have the ability to add your own bitmap on the left-hand side. If you wish to do so then hit the Set button and browse to the required bitmap file. Hit Ok and from the Screens menu select “Read Me Message”. Open your Readme.txt or a document that contains your information in a text editor and copy the entire contents to the clipboard. Then return to the installer and paste into the empty Read Me field. Hit Ok and from the Screens menu select “License Agreement” if necessary. Paste the License agreement into this window in the same way as the readme.txt. Hit Ok and from the Screens menu choose “Select Install Directory”. The

Win95/WinNT destination directory should be %ProgramFilesDir%\<company-name>\ followed by the correct folder name. Hit Ok and from the Screens menu choose “Select Install Type” if necessary. This screen would only be needed if you were allowing your user to select options on installation. Usually nothing would need to be edited on this screen; however, there may be cases when this screen needs to be customized. When complete hit Ok. If your installation has a custom option with different packages you may want to edit the Select Components screen. From the Screens menu choose “Select Components” and edit the text if necessary. When complete hit Ok and from the Screens menu choose “Select Program Folder”. Here you should add the name of the program folder you wish to have on the start menu or program manager. Hit Ok and from the Screens menu select “Finished Message2”. This window does not need to be edited. Hit Ok and return to the main project window.

Figure 16.

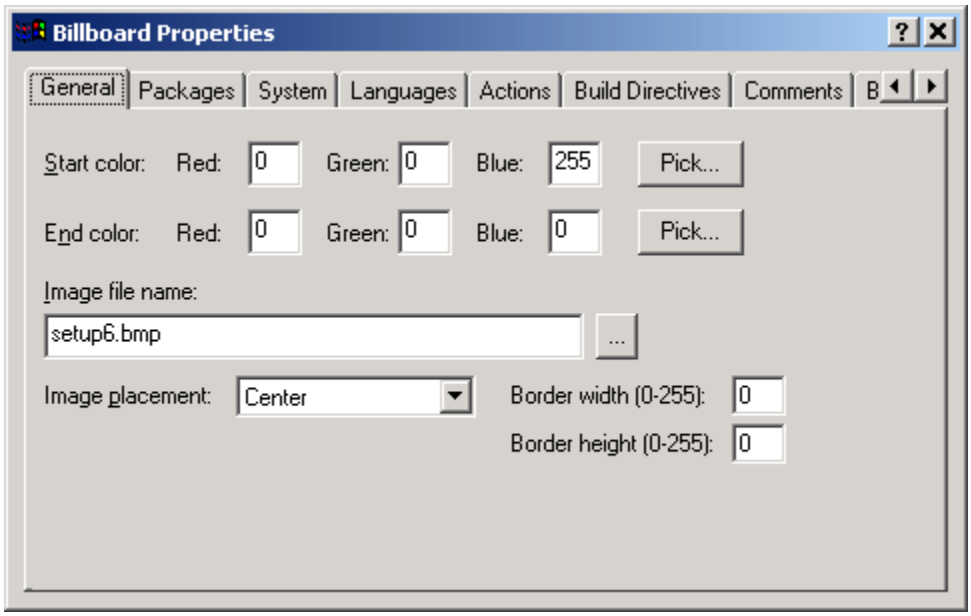


The next stage is to add the packages to the project. If you are not allowing the user to customize their installation, then you do not need to add packages. The packages are the groups of files that are selected when doing a custom install. For example, when doing a custom install on Omnis Studio 4.0 you will have options to install Omnis Studio, Web Enabler, Dams, VCS, Java, Web Client and the ODBC drivers. Packages would have to be created for each option and the relevant files would have been added to each one. You will notice in the installer project that there is five default packages already created. They are Support Files, Typical Set-up, Compact Set-up, Silent Set-up and Pre- screen items. The Support Files package should have the files that are not going to be installed but are just used by the installer e.g. the Support files folder. Files can be added to a package by highlighting the package and then checking the files in the project window that you wish to add. The Typical Set-up package should include all the files for a default installation. The Compact Set-up package contains the minimum files required for the product to run. To create a new package, select package from the add menu and enter the package details.

The final stage of the project creation is to add the action items. These are items that are triggered during the installation. Action items that will be created in this installer are items to display billboards and an action item to create the program icon on the start menu or program manager. You can add

many other types of action item as seen on the Action Item submenu. The action items have to be added to the packages like normal files and they will only be activated if they are part of the installing package. Action items will need to be moved into relevant parts of the project window as the installer will read from the top down when installing. For example, during an Omnis 7 patch install it is necessary to ensure that the patch files are installed into a correct Omnis 7 tree. Therefore, you would have a “Find” action item to search for the omnisres.dll in the tree. However, it is necessary for this action item to be called before you install the files. Therefore, the action item should be placed above the patch files in the project window. Another example is the displaying of billboards. Billboards are displayed using the “billboard” action item. These should be displayed from start to finish during the install. Multiple billboards should be given an equal part each of the install time. This means that multiple billboard action items should be created and added in different places throughout the installable files so as to change at regular intervals during the install. To add a billboard action item, select the Add Items menu and select Billboard. You then have to set the background color (Figure 17). It would be a good idea to set the same color as set in the background window so as to keep the same color all through the installation. Browse and select the bitmap you wish to display. Note that the dialog will only display the files that have been added to the Support Files package. Change the “Bitmap Placement” to “Center”. You will need to create a billboard action for each billboard.

Figure 17.



When this is complete you will need to spread the billboard action items amongst the tree so as to give an even display of each during installation. To add the Program icon to the Start-Menu or the Program Manager, select the “Program Item” from the Add menu. In the “Name” field enter “%ProductName%” (Figure 18). In the “Group” field add the name of the program folder. In the “File name” entry field enter “%TargetDir%\Omnis.EXE”. This action item should also be placed below the installable files in the project window.

When the project has been completed you will need to build the installer. The installer is a single executable file and you will be prompted to save the file. The file is usually called Setup.exe.

When all of these properties have been included in the project the installer can be built and tested. See Figure 19 for the completed project.

Figure 18.

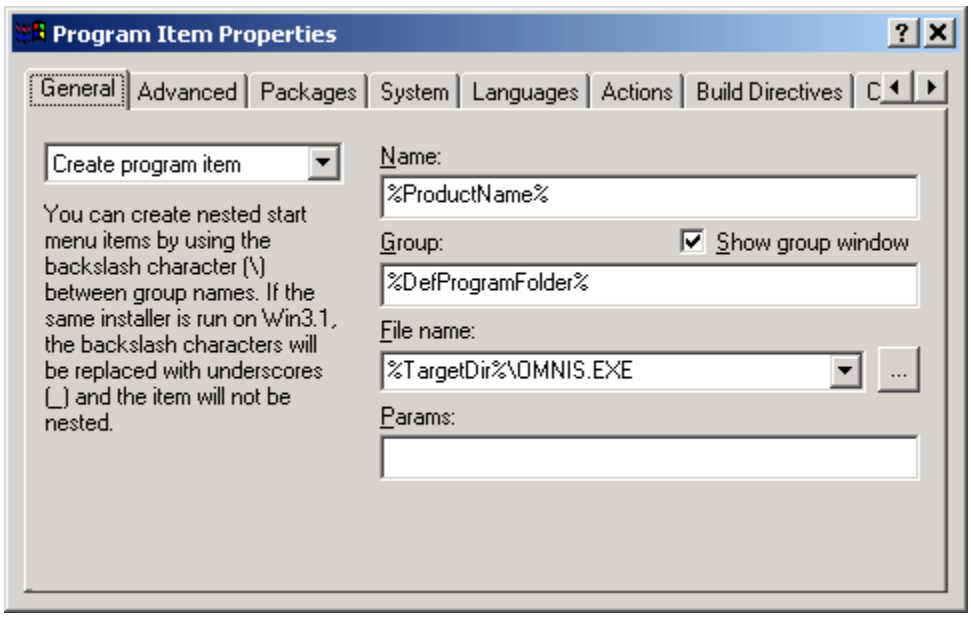
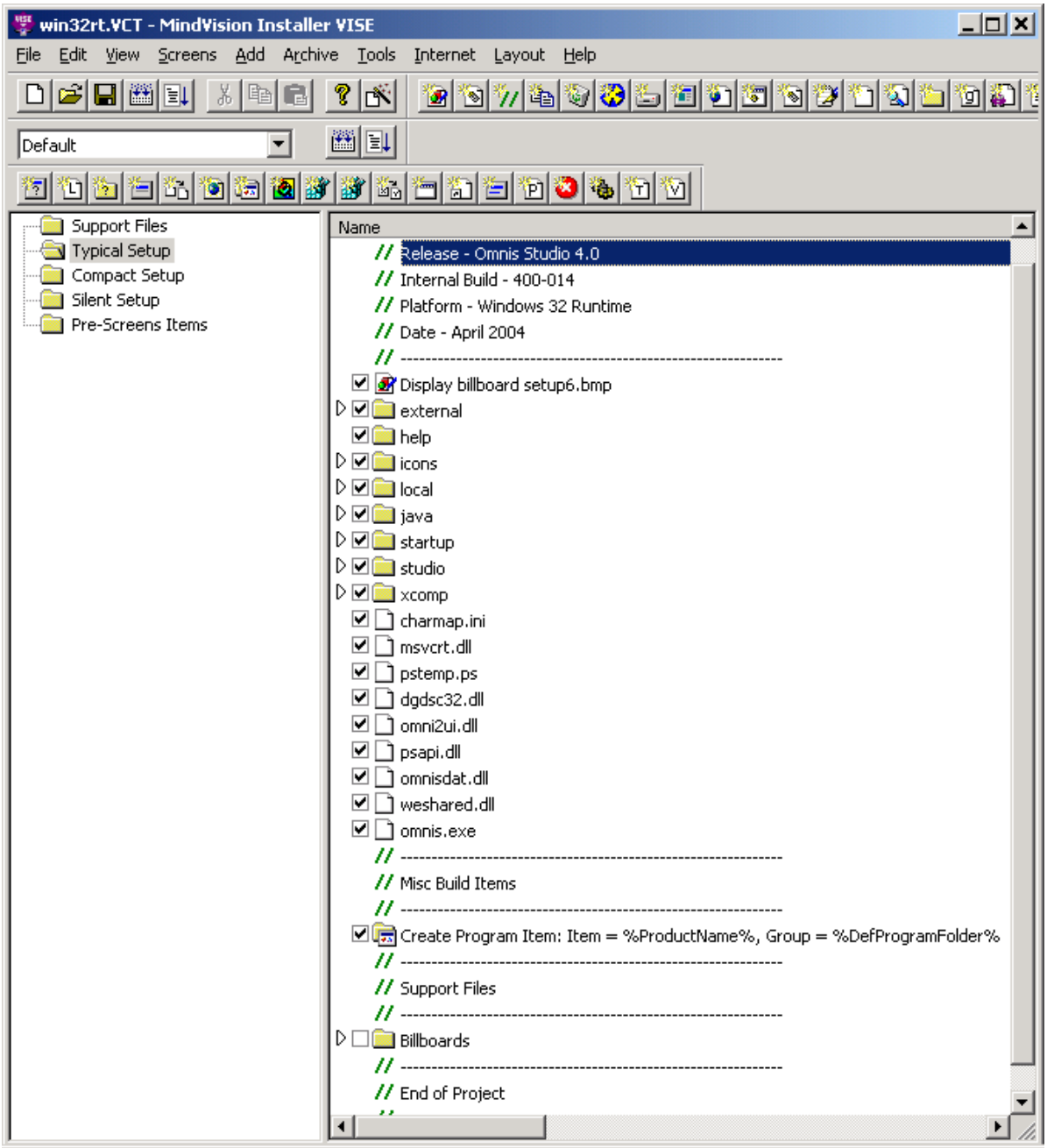


Figure 19.



5.3 macOS: Installer Vise

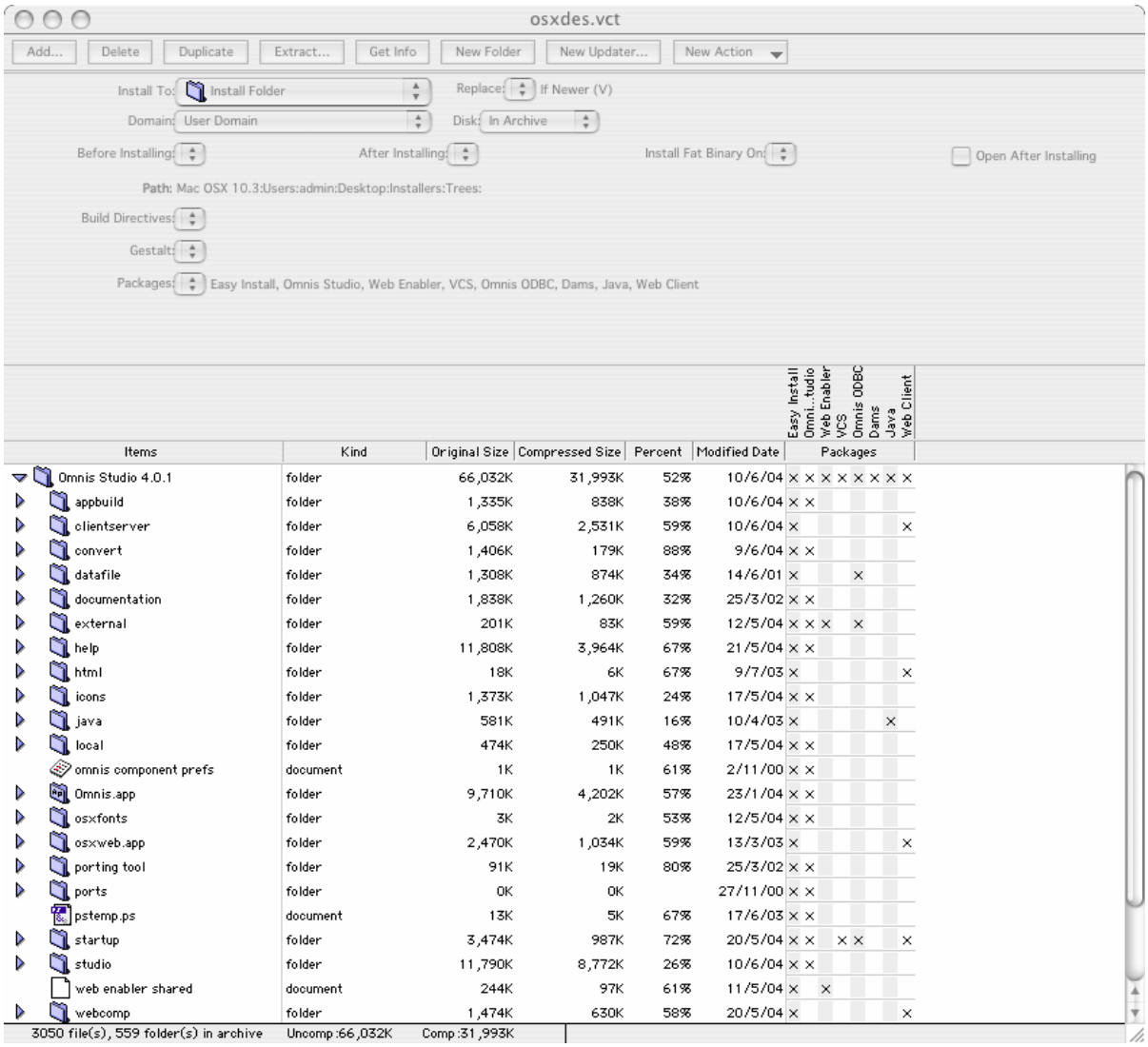
Once you have built the library from the Omnis VCS and added it to the runtime tree along with any other files relevant to the application, you will need to create an installer to install the tree onto the user's system. In this description of how to build the Macintosh Installer, we shall be using Installer Vise for macOS from Mindvision Software www.mindvision.com.

At this stage, you should make a full check of the trees, comparing with a previous release in order to ensure no unwanted differences.

When building the installer project for the final release, it would be a wise policy to update the existing release project. The advantages of this are the consistency of the installer program from one release to another, the elimination of minor errors, which can cause constant rebuilding, and also the saving of a considerable amount of time. We will create an installer from scratch for the benefit of the document.

Start Installer Vise and create a new project. The application tree on the local drive must then be neatly configured in a window before being dragged onto the Mac Installer project. The reason for this is the installer program will install the Omnis files to the user's local drive in exactly the same configuration as that with which they were dragged onto the project (Figure 20.). The files should all be in one relevantly named folder.

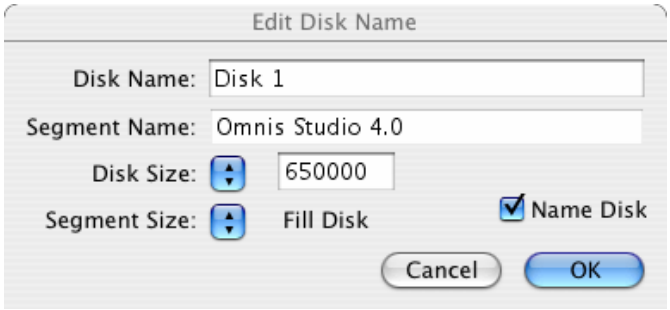
Figure 20.



For some files when dragged onto the archive window, the default location is set to the Extensions folder. This needs to be changed to the install folder by selecting the “Install To:” pop-up menu and selecting “Install Folder”.

From the Archive menu select “Edit Disk Names”. The number of disks required for an installation of your application will depend on which type of media you are using to deploy your application. For example if you are deploying using a CD then you will only need one disk, however if you are deploying using Floppy disks then you will need to add one disk per floppy. The disk name can be as simple as Disk 1 (Figure 21). The segment name is the name the user will see when they come to install the product, therefore it should be the name of the product followed by “Installer” e.g. Omnis Studio 4.0 Installer. The disk size for each disk should be the size of the media. The segment size needs to be “Fill Disk”.

Figure 21.



If you wish to display bitmaps while installing your application, you will need to add them to the project. To add the billboards, select “Billboards” from the Archive menu (Figure 22). The billboards can be displayed in various ways but we will use the “Sequential Billboard” method, so set the “Display Billboards” option to “Sequentially”. Next you will need to paste the billboards in display order into the “Sequential Billboards” field. This can be done by opening the billboard in Graphic Converter or another graphics application and copying the picture. Then click back on the installer project and paste in the billboard. Ensure the display time for each billboard is equal, it does not matter what the time is just so long as they are equal and Installer Vise will take care of the actual display time relative to the installation time.

The next stage is to ensure that certain settings are selected in the project. Open the “Archive Settings” window from the Archive menu (Figure 23). On each tab there are various settings that you may or may not require. These should be selected relevant to your application, installer and deployment system. It is always a good idea to display a text file before the installation. Select the Extras tab and locate the Readme.txt to be displayed at the beginning of the installation. The Readme.txt is automatically word wrapped by Installer Vise. The license agreement file should also be selected. If your application is very large (30M+) or has a great number of files (1000+) then you may need to increase the memory allocation for the installer, otherwise the installer will just crash when starting up. To change the memory allocation size, select the 'Attributes” tab and within the “SIZE Resource” group set the preferred size and minimum size to 3000k This should be enough for most applications. This only needs to be done for the large applications but will not cause any problems if done any size applications.

Figure 22.

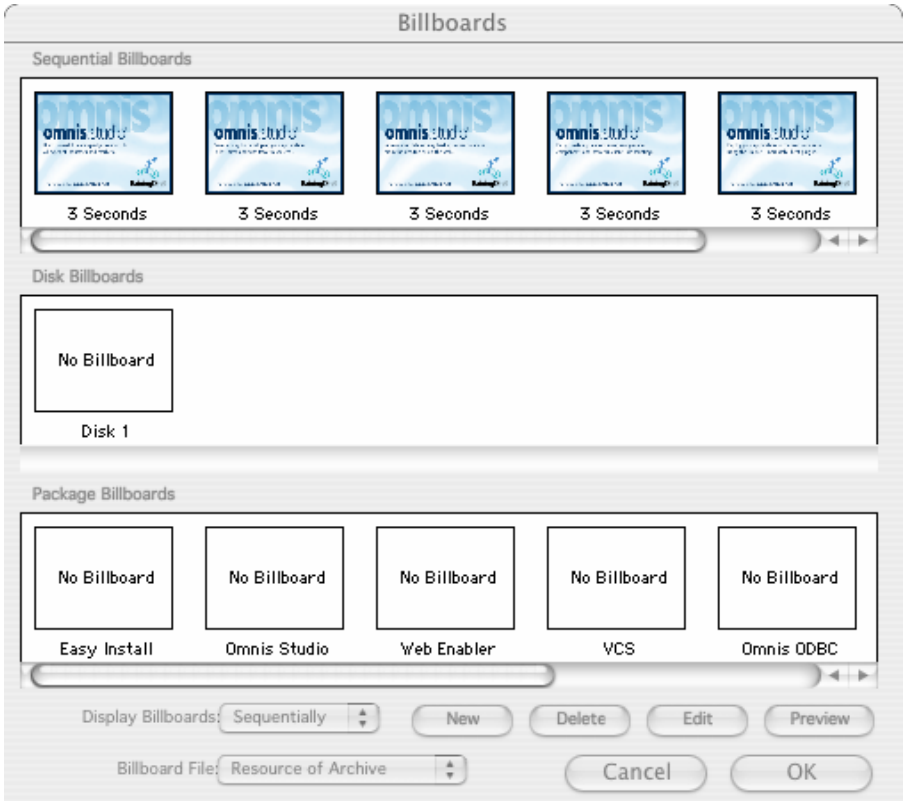
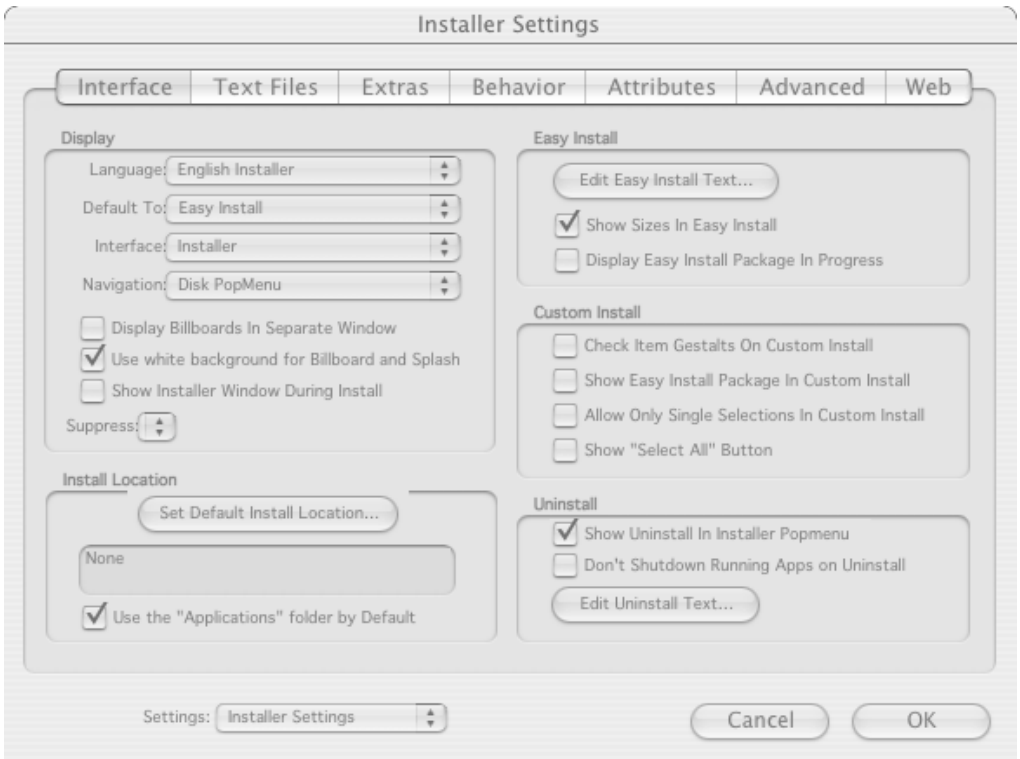


Figure 23.



The Installer is now ready to build. From the File menu select “Build Installer”. The build process will prompt you to save each disk in turn. When complete, you will have a number of files corresponding to the number of billboards. Disk 1 will be the installer executable while the other files will be called by Disk1. When copied onto CD the installation will be seamless.

Run the installer and check that the installation is satisfactory for a release standard installation.

6. Finishing Touches

Once all the installers have been completed, they have to be configured and copied onto the deployment media. The top level of the media should contain a Readme.txt and an Install.txt. The Readme.txt will be the same as that in the installer. The install.txt explains how to install the application. As well as the application folders for each platform, which will contain the installers, there may be additional folders or files depending on the product being released such as documentation, however if unsure it is wise to check the media of the previous release version.

It is advisable that you should record the size of the files on the media for future comparison, as this can be good indication as to whether there are any errors on the media.

A virus check should be made on your local drives on both Mac and Windows to ensure the files used to build the installer are virus free. Also a virus check should be made on the media. A copy of the media should be burnt for the retention as well as an archive of the release trees including the final installers. It is advisable that you archive all the folders, files, projects and installers involved in the release.