Advanced Options for the Application Builder

Presented by: Keisuke Miyako, Technical Account Manager – 4D Japan.

Introduction

The Build Application[1] tool allows the developer to quickly configure their build project with basic parameters. The tool automatically generates a **buildApp.4DSettings** XML project file in the *Settings* folder adjacent to the project or structure file and internally calls the BUILD APPLICATION[2] command.

Basic aspects of your final application related to branding, setup, and security can be configured using the application builder. However, as shown in the table below, the dialog exposes only a subset of options that is available in 4D.

Version	Key	Tool	Command
	BuildApplicationName	✓	✓
	ServerIconMacPath		✓
	ServerIconWinPath		✓
	RuntimeVLIconMacPath		✓
	RuntimeVLIconWinPath		✓
	CommonVersion		✓
	CommonCopyright		✓
	CommonCreator		✓
	CommonComment		✓
	CommonCompanyName		✓
	CommonFileDescription		✓
	CommonInternalName		✓
	CommonLegalTrademark		✓
	CommonPrivateBuild		✓
	CommonSpecialBuild		✓
	RuntimeVLVersion		✓
	RuntimeVLCopyright		✓
	RuntimeVLCreator		✓

[1] See Build Application in the official documentation.
[2] See BUILD APPLICATION in the official documentation.

Version	Key	Tool	Command
	RuntimeVLComment		✓
	RuntimeVLCompanyName		✓
	RuntimeVLFileDescription		✓
	RuntimeVLInternalName		✓
	RuntimeVLLegalTrademark		✓
	RuntimeVLPrivateBuild		✓
	RuntimeVLSpecialBuild		✓
	ServerVersion		✓
	ServerCopyright		✓
	ServerCreator		✓
	ServerComment		✓
	ServerCompanyName		✓
	ServerFileDescription		✓
	ServerInternalName		✓
	ServerLegalTrademark		✓
	ServerPrivateBuild		✓
	ServerSpecialBuild		✓
	ClientVersion		✓
	ClientCopyright		✓
	ClientCreator		✓
	ClientComment		✓
	ClientCompanyName		✓
	ClientFileDescription		✓
	ClientInternalName		✓
	ClientLegalTrademark		✓
	ClientPrivateBuild		✓

Version	Key	Tool	Command
	ClientSpecialBuild		✓
v19	ServerStructureFolderName[3]	✓	
v19	ClientServerSystemFolderName[4]	✓	
	BuildCompiled	✓	✓
	BuildComponent	✓	✓
	BuildApplicationSerialized	✓	✓
	BuildCSUpgradeable	✓	✓
	BuildServerApplication	✓	✓
	BuildMacDestFolder	✓	✓
	BuildWinDestFolder	✓	✓
	ServerIncludeIt	✓	✓
	ClientMacIncludeIt	✓	✓
	ClientWinIncludeIt	✓	✓
	RuntimeVLIncludeIt	✓	✓
	ServerMacFolder	✓	✓
	ServerWinFolder	✓	✓
	ClientMacFolderToMac	✓	✓
deprecated	ClientWinFolderToMac	✓	✓
	ClientWinFolderToWin	✓	✓
deprecated	ClientMacFolderToWin	✓	✓
	RuntimeVLMacFolder	✓	✓
	RuntimeVLWinFolder	✓	✓
deprecated	DataFilePath		✓
	IPAddress		✓

[3] See Multiple instances of merged server applications on the same machine on the official blog.
[4] See Multiple servers, one shared local resources on the official blog.

Version	Key	Tool	Command
	PortNumber		✓
v18	DatabaseToEmbedInClientMacFolder		✓
v18	DatabaseToEmbedInClientWinFolder		✓
v20	MacCompiledDatabaseToWinIncludeIt[5]	✓	✓
v20	MacCompiledDatabaseToWin[6]	✓	✓
	ArrayExcludedComponentName	✓	✓
	ArrayExcludedPluginID	✓	✓
	ArrayExcludedPluginName	✓	✓
	IncludeAssociatedFolders	✓	✓
v20	ArrayExcludedModuleName[7]	✓	✓
v16	LastDataPathLookup	✓	✓
v18	ClientWinSingleInstance		✓
v20	ShareLocalResourcesOnWindowsClient[8]		✓
v20	ClientUserPreferencesFolderByPath[9]		✓
	CurrentVers	✓	✓
	RangeVersMin	✓	✓
	RangeVersMax	✓	✓
v14	StartElevated		✓
v20	ServerDataCollection[10]		✓
	ArrayLicenseMac	✓	✓
	ArrayLicenseWin	✓	✓
	IsOEM		✓

[5] See Build a custom remote connection dialog on the official blog.
[6] See Simplified cross-platform client/server application building on Windows on the official blog.
[7] See Reduce your 4D apps' size with these new features on the official blog.
[8] See Share Local Resources Between Users with Windows Remote Desktop Services on the official blog.
[9] See Use duplicated merged client applications on the official blog.
[10] See Data Collection Insights on the official blog.

Version	Key	Tool	Command
deprecated	HardLink		✓
v14	MacSignature	✓	✓
v14	MacCertificate	✓	✓
v16	ServerSelectionAllowed		✓
v19	AdHocSign	✓	✓
v19	PackProject		✓
v20	HideDataExplorerMenuItem[11]		✓
v20	HideRuntimeExplorerMenuItem		✓
v20	ServerEmbedsProjectDirectoryFile[12]		✓
v20	UseStandardZipFormat[13]		✓
v20	HideAdministrationMenuItem[14]		✓

[11] See Disabling explorers on merged servers on the official blog.
[12] See Directory file management in merged server projects on the official blog.
[13] See Secure your app's resources with a new algorithm on the official blog.
[14] See Integrate your Own Administration Window for 4D Server on the official blog.

The build XML project file

To take advantage of the full potential of BUILD APPLICATION, one must specify keys in the build XML project file . The BuildApp class included with the demo project manages all options using an object which it internal converts to XML and back. This is done using 4D tags and a template file.

Embedding object notation in an XML template file

The 4DTEXT tag is like the String function but handles null differently.

```
<!--#4DTEXT String(null)--> is "null"

<!--#4DTEXT null--> is an empty string

<!--#4DTEXT Num(null)--> is error #59

<!--#4DTEXT Bool(null)--> is False/Faux
```

One must use the appropriate functions to avoid errors or suppress unwanted values.

Cross-platform build support

Earlier version of 4D supported building cross-platform client server applications on either platform. build keys such as **ClientWinFolderToMac** or **ClientMacFolderToWin** were used for that purpose. One could use a copy of 4D Volume Desktop runtime to build auto-upgrade clients for both platforms. That option is now deprecated. It is not possible to build macOS clients on Windows because a simple file copy with no code-signing results in an app invalid for distribution.

Client applications can only be built for the host platform. Once built and archived with the .4darchive file extension, the file can be moved to the server target platform and embedded using the build keys mentioned earlier. The original .4darchive file format uses a proprietary compression (same as BACKUP, you can decompress it with the RESTORE command) but one can use a renamed standard .zip file.

Data file path

Prior to v15, the last data file path was saved inside the 4DC stricture file. A freshly built structure file does not contain this information, which is why the application would always ask for the data file location on first launch or after an auto-upgrade. To avoid this one could use the **DataFilePath** build key and pre-set the last data file path during build. For instance, one could set a short file name to use the data file adjacent to the structure file (however the build key itself does not create or copy a data file). This option is now deprecated. It is now considered problematic to store the data file inside the application. It is also not recommended to modify the structure file after it has been installed to a system directory reserved for program files.

Starting with v15, it is recommended to activate the new architecture for deployment[15], which stores the last data file path outside of the application inside a folder that is safe for storing local data (AppData on Windows and Application Support on macOS). Similarly, it is now recommended to use the *Default Data* folder with a **Default.4DD** file inside to prevent the application from prompting for a data file on first launch.[16]

Default Data and journaling

The journal file path is recording in the data file. Since a journal file is only created after a backup, simply activating it in settings does not enable or disable journaling for the current data file. One must perform a backup, or else create a new data file after deactivating it to make sure the data file does not require a journal file to operate. This understanding it important in the context of creating the Default.4DD file which is supposed to be read-only, immutable and placed inside the application package on Mac.

Headless mode

The BUILD APPLICATION command does not work in headless mode[17]. It is not possible to build an application in utility mode or with tool4d[18].

Build4D[19] is an open-source substitute for BUILD APPLICATION. The demo project contains something similar, a project that accepts path to a project file and a build XML project file. A complementary project implements an enhanced build application dialog with all the advanced options which can invoke the BUILD APPLICATION command for the local project or the generic compile & build project via tool4d.

[15] See Compatibility page in the documentation.
[16] See Data file management in final applications in the documentation.
[17] See Headless 4D applications on the official blog.
[18] See A Tool for 4D Code Execution in CLI on the official blog.
[19] See Build your Compiled Structure or Component with Build4D on the official blog.

Printing to the command line interface

The CLI class included with the demo project implements ASCII escape code sequences to print coloured messages to the console. The *Terminal* app on macOS supports this feature by default. On Windows, it may need to be activated by editing the registry.

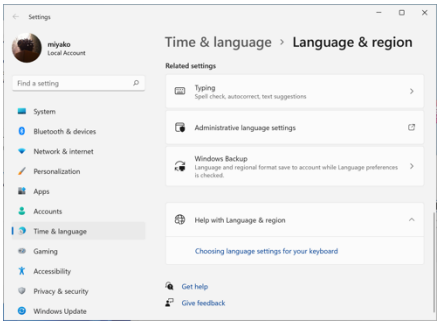
Key: HKEY_CURRENT_USER\Console\VirtualTerminalLevel

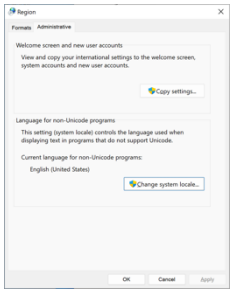
Type: DWORD

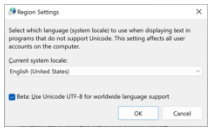
Value: 00000001

For historic compatibility reasons, the Windows Command Prompt (CMD.EXE) is a non-Unicode application. You can switch it to Unicode in Settings.

Time & language / Language & region / Administrative language







Some tips on printing to the CLI:

Print CR to move the cursor position to the start of line. This allows you to print over existing text.

Print LF to start a new line.

Print % twice to print the percent sign.

Port number

By default, the server application is published over TCP port number 19813 (hexadecimal 0x4D65). It uses the same port number to broadcast the UDP datagram "4D Server II"[20]. One can discover services available on the local subnet by listening for this signal.

The build key CS.PortNumber has the effect of modifying the EnginedServer.xml file created inside the *Database* folder of the client application. More specifically it adds a colon symbol followed by the port number to the **server_path** attribute (the domain or IP address is optional). This allows the client to connect to a server that is published over an alternative port number. The build key only has effect on the client. It does not alter the port number on the server side. To also change the port number of the server, one must modify the **publication_port** attribute in the **settings.4DSettings** XML file inside the *Sources* folder.

The port number is one of the few pieces of information that originates outside the build settings XML project file. The _SettingsXmlParser class included with the demo project implements a function to locate the settings file related to the specified project and changes its server port number.

SDI mode

Another critical piece of information missing in the build settings XML project file is the preference to run the application in SDI mode on Windows. Unlike the port number mentioned above, the user settings take precedence over structure settings for with regards to the **sdi_application** attribute. The _SettingsXmlParser class included with the demo project implements a function to determine the effective mode. If the attribute does not exist in user settings, the structure settings is respected. Only an explicit "false" value should cancel structure settings. For that reason, the function return value is Variant, which can be True, False or Null.

[20] See UDP Commands, Overview in the documentation.