Building Applications Examples – Part 2
General Parameters and Licenses

By Josh Fletcher, Technical Services Team Member, 4D Inc.

Technical Note 08-13

Abstract

4D 2004 features over 80 XML keys that can be used to build customized applications. The XML keys documentation gives basic information about what the keys do, but there are few examples. The goal of this series of Technical Notes is two-fold:

- Provide examples for each XML key, on Mac OS X and Windows, to aid the 4D developer in understanding what each key does.

- To make editing the XML project file, used in building applications, an easier task by providing a GUI editor.

Part 1 of this series provided a 4D database that can be used to edit project files.

In part 2 of this series, examples for the XML keys from the General Parameters and Licenses themes are presented.

Table of Contents

Introduction

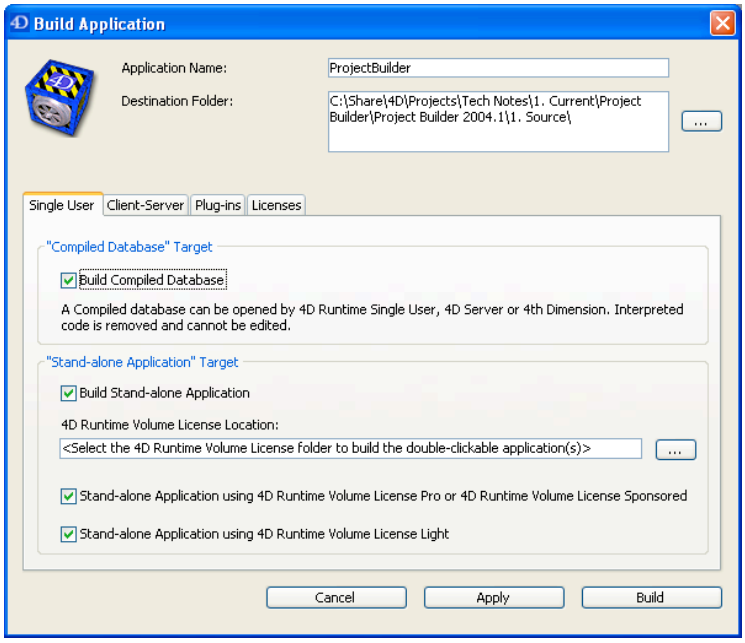
4D 2004 features the ability to control the application building process with the use of an XML project file. There are over 80 XML keys that can be used to build customized applications. The project file can contain some or all of these XML keys, as needed.

These XML keys allow many parts of a 4D application to be customized; for example the application name can be changed, or the path to the data file can be altered. The 4D 2004 XML Keys documentation can be downloaded from this page:

http://www.4d.com/support/documentation.html

(The title is listed as "XML Keys BuildApplication")

The XML keys documentation gives basic information about what the keys do, but there are few examples. Similarly the "Build Application" dialog found in 4D offers access to only a subset of the XML keys available (approximately 15 of the keys can be edited with this dialog):



If access to the other keys is needed, the **BUILD APPLICATION** command must be used and the XML project file must be edited manually. Documentation for the BUILD APPLICATION command can be found here:

http://www.4d.com/4ddoc2004/CMU/CMU00871.HTM

The goal of this Technical Note series is two-fold:

- Provide examples for each XML key, on Mac OS X and Windows, to aid the 4D Developer in understanding what each key does.

- To make editing the XML project file an easier task by providing a GUI editor for them.

The XML keys are organized into themes as follows:

- **General Parameters –** These are keys that are not specific to single-user, server, nor client applications.
- **Licenses –** These keys are used to specify the location of license files necessary for building applications.
- **CS –** These keys are specific to building client-server applications.
- **Sources Files –** These keys are concerned with the files that go into building applications, e.g. paths to 4D software, icon files, etc.
- **Versioning –** These keys are used to alter the meta-information for the merged application, e.g. version and copyright information.

This document will cover the examples for the XML keys from the General Parameters and Licenses themes. These examples are intended to better explain what each XML key does.

Terminology

This section defines some of the terms used in this Technical Note, for clarification.

- **XML key –** Refers to a single XML element within the list of XML elements that 4D has defined for use with the application building features.

- **Theme –** The XML keys available are organized into 5 themes: General Parameters, Licenses, CS, SourcesFiles, and Versioning.

- **Compiled structure –** A compiled 4D database (.4dc file).

- **Merged application –** Refers to a 4D database that has been "merged" with 4D software in order to create a single application. Also known as "stand-alone" or "built".

- **Merged single-user application –** Refers to a 4D database that has been merged with the 4D Runtime Volume License software.

- **Merged server application –** Refers to a 4D database that has been merged with the 4D Server software.

- **Merged client application –** Refers to the client portion of a merged client-server database. In this case there is no 4D database but the 4D Client software may have been customized as part of the build process.

- **Install image** – This term is used to describe all of the files that make up the installation of a merged application. For example a merged 4D Client install image contains the executable for the application, various libraries and resource files, etc. All of these files are collectively referred to as the "install image".

- **Install client –** The install image of a merged client application. This is the software that would be manually installed on any given computer that will be used to connect to the merged server application.

- **Upgrade client –** The install image that is downloaded to the client machine for the purposes of automatically upgrading. This is not the software that would be manually installed on a given client computer. It is used by the automatic upgrade feature only.

Structure of the Examples

The example sections are organized by theme. Furthermore the XML keys appear in the same order as they do in the *4th Dimension XML Keys BuildApplication* documentation.

For each XML key example the following information is presented:

- A short, general description of what the key does.
- Dependencies on other keys are listed, if applicable.
- Windows example:
  - Windows-specific description (if applicable)
  - Compiled structure example (if applicable)
  - Single-user example (if applicable)
  - Client example (if applicable)
  - Server example (if applicable)
- Mac OS X example:
  - Mac-specific description (if applicable)
  - Compiled structure example (if applicable)
  - Single-user example (if applicable)
  - Client example (if applicable)
  - Server example (if applicable)

Within each example the value of the XML key being tested – and optionally the values of other related keys – is shown. Please note that the XPATH of the keys is ignored. Keys from different themes will be presented side by side, as in:

```
<BuildApplicationLight>True</BuildApplicationLight>
<RuntimeVLIncludeIt>True</RuntimeVLIncludeIt>
```

The BuildApplicationLight key is not a sibling of the RuntimeVLIncludeIt key, in terms of XPATH notation. This is done to simplify the examples but remember that the keys must be placed at the correct location within the project file for them to work.

General Parameters

This section covers examples of the XML keys from the "General Parameters" theme.

BuildApplicationName

This key affects the name of the compiled database, single-user application, client application, and server application as follows:

- The name of the compiled database and single-user application is the value of this key.
- The name of the client application is the value of this key with " Client" appended to it.
- The name of the server application is the value of this key with " Server" appended to it.

Note that the resulting name is limited to 31 characters, including any extension. Also note that there is no error or warning if this limit is exceeded. The name will simply be truncated to fit within the bounds.

Example

In this example the BuildApplicationName key was set as follows:

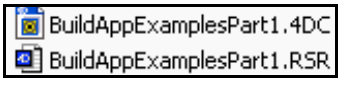
```
<BuildApplicationName>BuildAppExamplesPart1</BuildApplicationName>
```

Windows

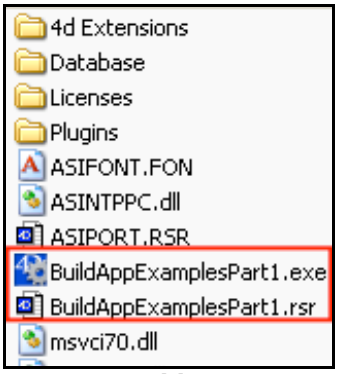
This key affects the name of the compiled structure and compiled structure resource file, as well as the executable and resource files of single-user, client, and server applications on Windows.

It also modifies the "Product Name" property for the executable files. To view this information, right-click on the executable file, select Properties, and select the "Version" tab in the Properties dialog.

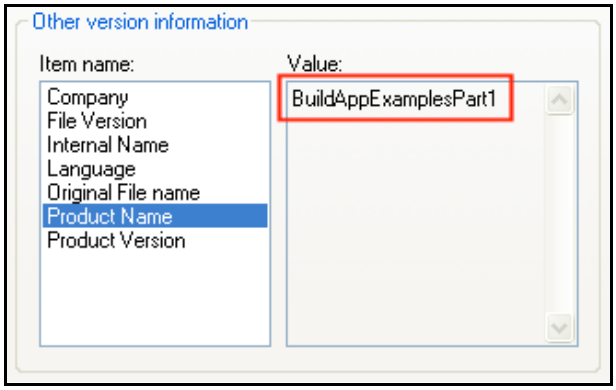
Compiled structure



Single-user application

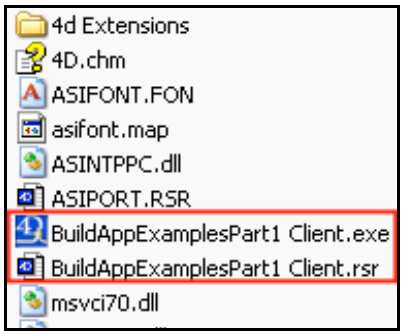


Executable name

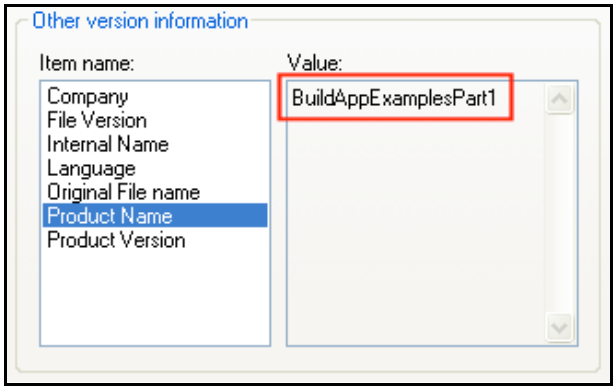


Product Name property

Client application

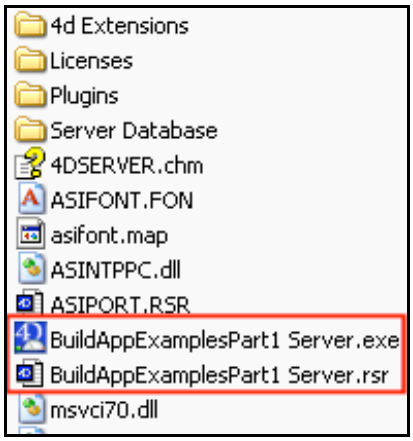


Executable name

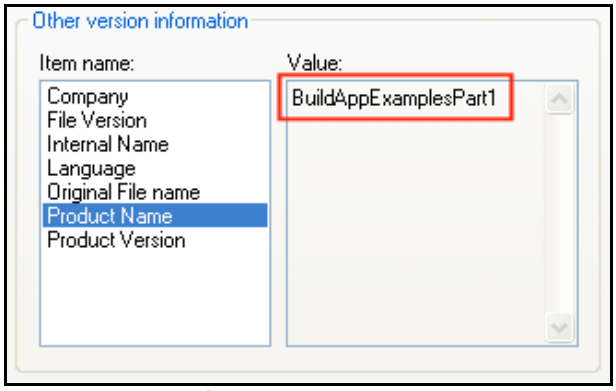


Product Name property

Server application



Executable name

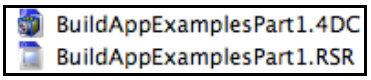


Product Name property

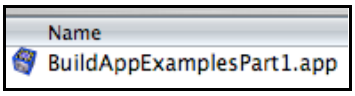
Mac OS X

This key affects the name of the compiled structure and compiled structure resource file, as well as the application package of single-user, client, and server applications on Mac OS X.

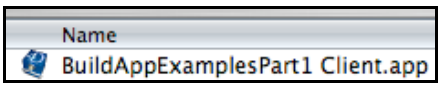
Compiled structure



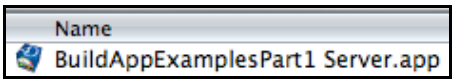
Single-user application



Client application



Server application



BuildWinDestFolder

This key specifies the location where the results of building the application will be placed on Windows. If no value is specified, the folder that contains the structure is used. The path can be relative to the structure file location, or absolute.

The compiled structure is placed in a folder called "Compiled Database". The single-user application is placed in a folder called "Final Application". The client and server applications are placed in a folder called "Client Server Executable". Furthermore the client is placed in a subfolder called "Client" and the server is placed in a subfolder called "Server".

Example

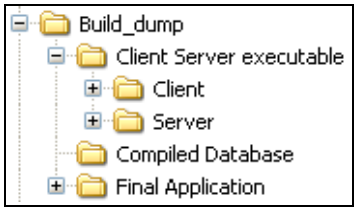
In this example the BuildWinDestFolder key was set as follows:

```
<BuildWinDestFolder>\Build_dump\</BuildWinDestFolder>
```

Note that this is a relative path.

Furthermore, a compiled structure, single-user, client, and server application were all built.

Windows



Mac OS X

This key has no effect on Mac OS X.

BuildMacDestFolder

This key specifies the location where the results of building the application will be placed on Mac OS X. If no value is specified, the folder that contains the structure is used. The path can be relative to the structure file location, or absolute.

The compiled structure is placed in a folder called "Compiled Database". The single-user application is placed in a folder called "Final Application". The client and server applications are placed in a folder called "Client Server Executable". Furthermore the client is placed in a subfolder called "Client" and the server is placed in a subfolder called "Server".

Example

In this example the BuildMacDestFolder key was set as follows:

```
<BuildWinDestFolder>:Build_dump:</BuildWinDestFolder>
```

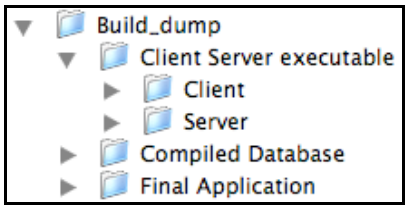
Note that this is a relative path.

Furthermore, a compiled structure, single-user, client, and server application were all built.

Windows

This key has no effect on Windows

Mac OS X



DataFilePath

This key is used to specify the location of the data file for merged applications at first launch. As with any 4D database, the path to the data file will be stored in the structure once it has been opened, so this key really only affects the first time you launch the database.

The path can be relative or absolute. Furthermore, it can be a Windows path, Mac OS X HFS path, or a POSIX path.

Note: *A data file is **not** placed or created at this location as part of the build process. This key is only concerned with specifying the location to search for the data file but the data file must be placed at that location by some other means.*

Example

Windows

On Windows by default 4D looks for a data file with the same name in the same folder as the structure. If it cannot find the data file there, it will display a Open File dialog.

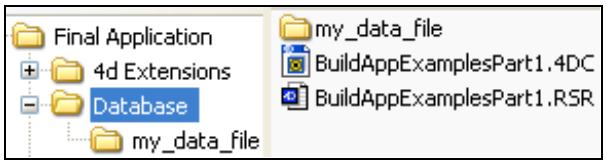
In this example the DataFilePath was set as follows:

```
<DataFilePath>\my_data_file\4DProjectEditor.4DD</DataFilePath>
```

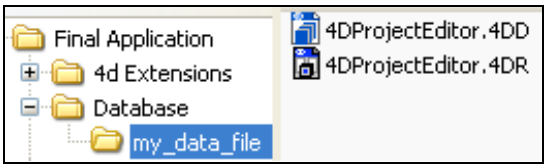
Thus the data file needs to be placed in a folder called "my_data_file", created next to the structure file of the merged application.

Single-user application

Tip: *In a merged single-user application on Windows, the structure is located in a folder called "Database" within the merged application.*



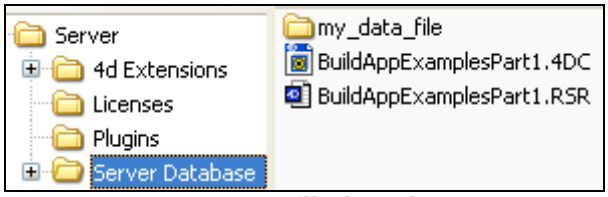
Structure file location



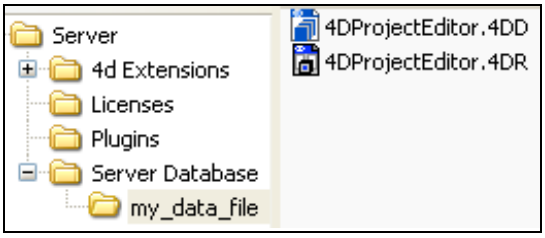
Data file location

Server application

Tip: *In a merged server application on Windows the structure is located in a folder called "Server Database" within the merged application.*



Structure file location



Data file location

Mac OS X

On Mac OS X, 4D looks for a data file with the same name in the same folder as the application package, by default. If it cannot find the data file there, it will display an Open File dialog.

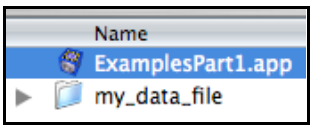
In this example the DataFilePath was set as follows:

```
<DataFilePath>:my_data_file:4DProjectEditor.4DD</DataFilePath>
```

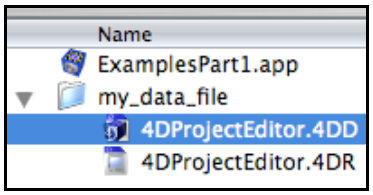
Thus the data file needs to be placed in a folder called "my_data_file", created next to the application package.

Single-user application

Tip: *In a merged single-user application on Mac OS X the structure is located in a folder called "Database" within the merged application package. To get inside the package, right-click on the application and choose "Show Package Contents".*



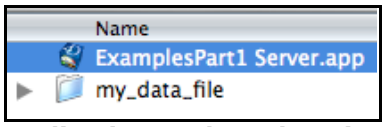
Application package location



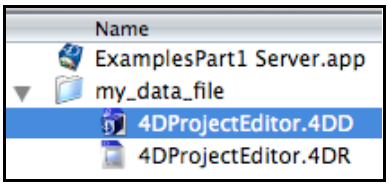
Data file location

Server application

Tip: *In a merged server application on Mac OS X, the structure is located in a folder called "Server Database" within the merged application package. To get inside the package, right-click on the application and choose "Show Package Contents".*



Application package location



Data file location

BuildCompiled

This key tells 4D whether or not to generate a compiled database. The compiled database is placed in a folder called "Compiled Database" at the location specified by the key BuildWinDestFolder or BuildMacDestFolder or, if those keys were not used, next to the structure file.

This key can be set to "True" or "False".

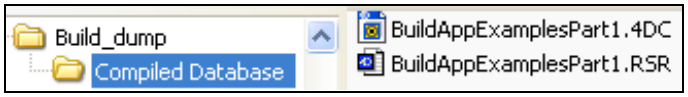
Note: *Even if BuildCompiled is set to "False", a compiled database may still be created. This is necessary in order build merged applications, for example. If this key is set to "False", 4D simply deletes the compiled database when it is done with it, at the end of the build process.*

Example

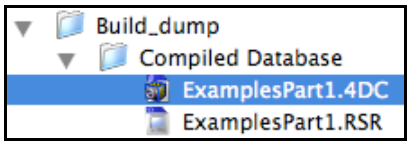
In this example, BuildCompiled was set as follows:

```
<BuildCompiled>True</BuildCompiled>
```

Windows



Mac OS X



BuildApplicationLight

This key tells 4D whether or not to build a **Light** application. A Light application is a special version of a merged single-user application. It is still merged with the 4D Runtime Volume License software, but it has no license so the resulting application will run in "demo" mode, with all of the limitations therein.

This key can be set to "True" or "False".

If set to "True", this key depends on the keys RuntimeVLWinFolder and/or RuntimeVLMacFolder. These keys must have a valid path to the 4D Runtime Volume License software or the build will fail. This key also depends on the RuntimeVLIncludeIt key. If RuntimeVLIncludeIt is not set to "True" the Light application will not be built.

The resulting application is placed in a folder named "Light Application" at the location specified by the key BuildWinDestFolder or BuildMacDestFolder or, if those keys were not used, next to the structure file.

Example

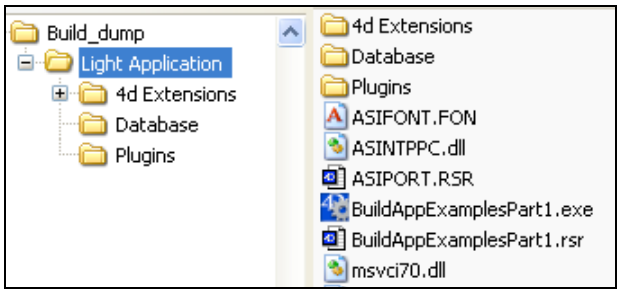
The keys were set as follows:

```
<BuildApplicationLight>True</BuildApplicationLight>
<RuntimeVLIncludeIt>True</RuntimeVLIncludeIt>
```

Windows

The RuntimeVLWinFolder key was set as follows:

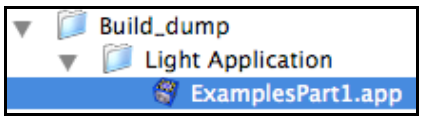
```
<RuntimeVLWinFolder>C:\4D Runtime Volume License 2004.7\</RuntimeVLWinFolder>
```



Mac OS X

The RuntimeVLMacFolder key was set as follows:

```
<RuntimeVLMacFolder>MacHD:4D Runtime Volume License.app</RuntimeVLMacFolder>
```



BuildApplicationSerialized

This key serves two purposes:

- It tells 4D to create a merged single-user application.
- It tells 4D to integrate a "4D Runtime Volume Pro" or "4D Runtime Volume Sponsored" license into the resulting merged application.

If set to "True", this key depends on the keys RuntimeVLWinFolder or RuntimeVLMacFolder for Windows or Mac OS X respectively. These keys must have a valid path to the 4D Runtime Volume License software or the build will fail.

This key also depends on the RuntimeVLIncludeIt key. If RuntimeVLIncludeIt is not set to "True", the merged application will not be built.

Of course a "4D Runtime Volume Pro" or "4D Runtime Volume Sponsored" license must be specified using the ArrayLicenseWin or ArrayLicenseMac keys in order for the license to be integrated. If no deployment license is specified, the build will still work but the resulting merged application will run in demo mode.

Once the build is complete the integrated license files will be placed in a folder called "Licenses" within the merged application.

The resulting application is placed in a folder named "Final Application" at the location specified by the key BuildWinDestFolder or BuildMacDestFolder or, if those keys were not used, next to the structure file.

Example

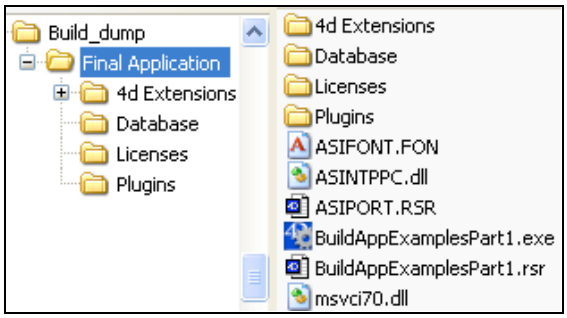
The keys were set as follows:

```
<BuildApplicationSerialized>True</BuildApplicationSerialized>
<RuntimeVLIncludeIt>True</RuntimeVLIncludeIt>
```

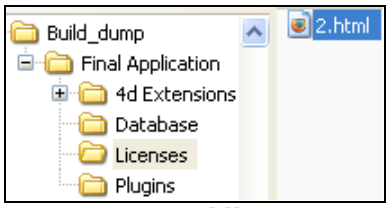
Windows

The Windows keys were set as follows:

```
<RuntimeVLWinFolder>C:\4D Runtime Volume License 2004.7\</RuntimeVLWinFolder>
<ArrayLicenseWin>
 <ItemsCount>2</ItemsCount>
 <Item1>D:\Documents and Settings\All Users\Application Data\4D\Licenses\4D
        Developer Edition V 2004 Windows.html</Item1>
 <Item2>D:\Documents and Settings\All Users\Application Data\4D\Licenses\4D
        Runtime VL Sponsored 2004 (Unlimited).html</Item2>
</ArrayLicenseWin>
```



Serialized application



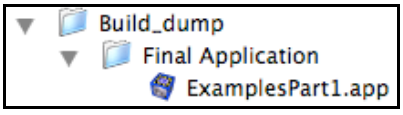
Integrated license

Mac OS X

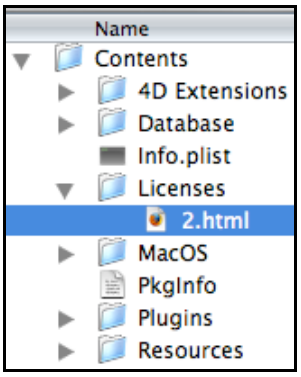
The Mac OS X keys were set as follows:

```
<RuntimeVLMacFolder>MacHD:4D Runtime Volume License.app</RuntimeVLMacFolder>
<ArrayLicenseMac>
 <ItemsCount>2</ItemsCount>
 <Item1>MacHD:Library:Application Support:4D:Licenses:4D Developer Edition V 2004
        MacOS.html</Item1>
<Item2>MacHD:Library:Application Support:4D:Licenses:4D Runtime VL Sponsored 2004
       (Unlimited).html</Item2>
</ArrayLicenseMac>
```

Tip: *In a merged application on Mac OS X, the "Licenses" folder is located within the application package. To get inside the package, right-click on the application and choose "Show Package Contents".*



Serialized application



Integrated license

ArrayExcludedPluginName

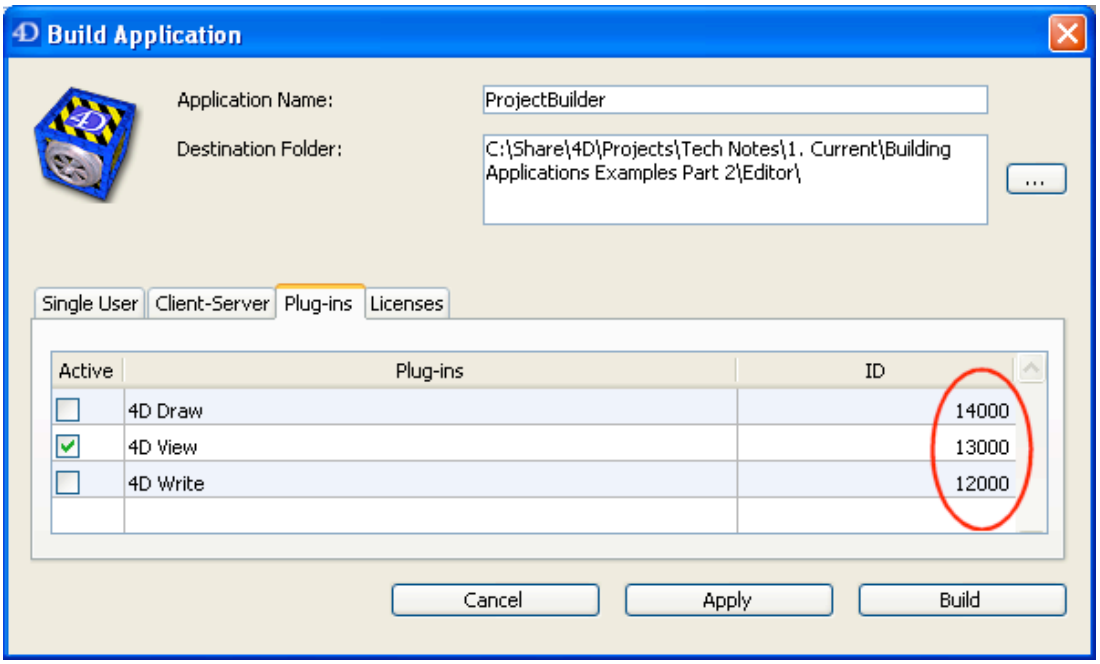
This key has no effect. Feel free to specify plug-in names if it makes the XML file easier to read, but ArrayExcludedPluginID must be used to exclude plug-ins.

ArrayExcludedPluginID

This key can be used to exclude plug-ins from merged applications. In this case the plug-ins are excluded by ID. Note that the plug-in IDs must be unique.

This key is an XML array and thus has two child elements to describe its contents: ItemsCount and ItemX, where "X" is the item number.

The easiest way to find the IDs, if not known, is to look in the Build Application dialog in 4D, in the "Plug-ins" tab:

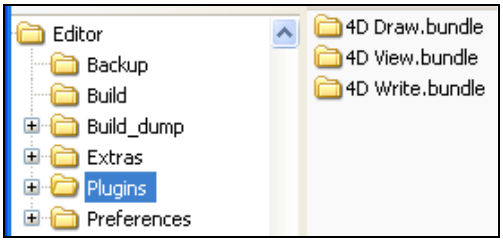


Example

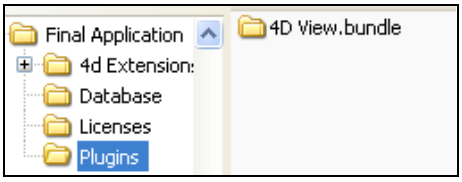
This example excludes 4D Write (14000) and 4D Draw (12000) from the merged application. 4D View is left included.

```
<ArrayExcludedPluginID>
  <ItemsCount>2</ItemsCount>
  <Item1>12000</Item1>
  <Item2>14000</Item2>
</ArrayExcludedPluginID>
```

Windows



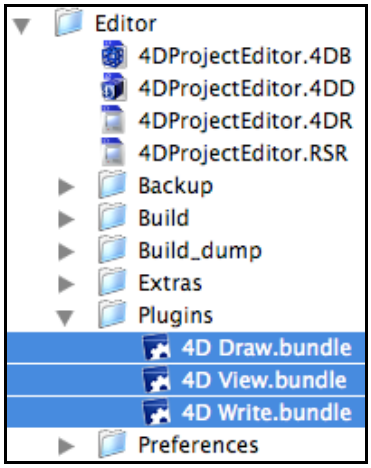
Source plug-ins



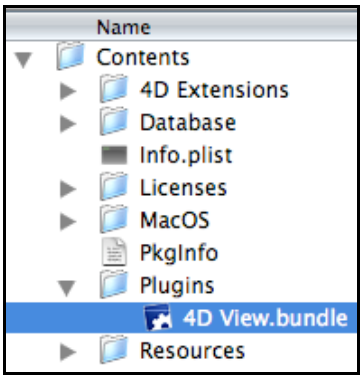
Plug-ins after build

Mac OS X

Tip: *In a merged application on Mac OS X, the "Plugins" folder is located within the application package. To get inside the package, right-click on the application and choose "Show Package Contents".*



Source plug-ins



Plug-ins after build

Licenses

This section covers examples of the XML keys from the "Licenses" theme.

ArrayLicenseWin

This key is used to specify necessary license files for building applications on Windows. Note that there are two types of licenses that will be typically specified:

- Development licenses
- Deployment licenses

The licenses for 4D Developer Edition and 4D Server Developer Edition are examples of Development licenses. Development licenses should always be specified when building applications.

Deployment licenses are those licenses that are integrated into the merged application at build time. These are usually "Volume" licenses, e.g. "4D Runtime Volume Pro" or "4D Runtime Volume Sponsored". Deployment licenses are optional, i.e. you can build Light applications without any Deployment licenses.

Integrated Deployment licenses are placed in a folder called "Licenses" within the merged application.

Note that "Single-User" licenses should not be specified here. An example of a Single-User license is "4D Runtime Single-User". Single-User licenses should be installed and registered on the deployment machine. They are not taken into account at build time.

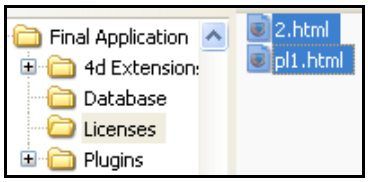
Example

In this example a merged single-user application was built that uses the 4D Runtime Volume Sponsored license and includes a volume plug-in license for 4D View. Note that two license files were created. While the content of these files is encoded, the presence of the files is a quick way to check if the build worked.

Windows

The keys were set as follows:

```
<ArrayLicenseWin>
 <ItemsCount>3</ItemsCount>
 <Item1>D:\Documents and Settings\All Users\Application Data\4D\Licenses\4D
Developer Edition V 2004 Windows.html</Item1>
 <Item2>D:\Documents and Settings\All Users\Application Data\4D\Licenses\4D Runtime
VL Sponsored 2004 (Unlimited).html</Item2>
 <Item3>D:\Documents and Settings\All Users\Application Data\4D\Licenses\4D View
Runtime VL Sponsored 2004 (Unlimited).html</Item3>
</ArrayLicenseWin>
```



Mac OS X

This key has no effect on Mac OS X.

ArrayLicenseMac

This key is used to specify necessary license files for building applications on Mac OS X. Note that there are two types of licenses that will be typically specified:

- Development licenses
- Deployment licenses

The licenses for 4D Developer Edition and 4D Server Developer Edition are examples of Development licenses. Development licenses should always be specified when building applications.

Deployment licenses are those licenses that are integrated into the merged application at build time. These are usually "Volume" licenses, e.g. "4D Runtime Volume Pro" or "4D Runtime Volume Sponsored". Deployment licenses are optional, i.e. you can build Light applications without any Deployment licenses.

Integrated Deployment licenses are placed in a folder called "Licenses" within the merged application.

Note that "Single-User" licenses should not be specified here. An example of a Single-User license is "4D Runtime Single-User". Single-User licenses should be installed and registered on the deployment machine. They are not taken into account at build time.

Example

In this example a merged single-user application was built that uses the 4D Runtime Volume Sponsored license and includes a volume plug-in license for 4D View. Note that two license files are created. While the content of these files is encoded, the presence of the files is a quick way to check if the build worked.

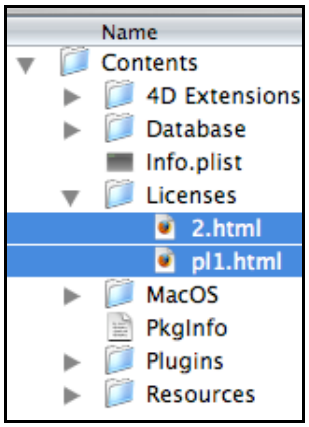
Windows

This key has no effect on Windows.

Mac OS X

The keys were set as follows:

```
<ArrayLicenseMac>
 <ItemsCount>3</ItemsCount>
 <Item1>Macintosh HD:Library:Application Support:4D:Licenses:4D Developer Edition V
2004 MacOS.html</Item1>
 <Item2>Macintosh HD:Library:Application Support:4D:Licenses:4D Runtime VL
Sponsored 2004 (Unlimited).html</Item2>
 <Item3>Macintosh HD:Library:Application Support:4D:Licenses:4D View Runtime VL
Sponsored 2004 (Unlimited).html</Item3>
</ArrayLicenseMac>
```



Conclusion

This Technical Note provided examples of the XML keys from the General Parameters and Licenses themes. These examples built upon the information presented in the XML keys documentation. Through these examples, the 4D developer should have a better understanding of what each XML key from these themes does.

Future editions of this Technical Note series will cover the remaining XML keys.

Related Resources

4D 2004 XML Keys Documentation:
http://www.4d.com/support/documentation.html

BUILD APPLICATION command:
http://www.4d.com/4ddoc2004/CMU/CMU00871.HTM

Tech Note: Building Applications Examples – Part 1 - 4D 2004 Project Editor
http://www.4d.com/support/technotes.html

Tech Note: Building Applications with 4D 2004: Automatic Client Upgrade
http://www.4d.com/support/technotes.html