External Component Version Numbers

As third-party Omnis component suppliers we are in the unfortunate position that we have to support numerous different versions of Omnis Studio for which we have to build and distribute different DLLs. Our support team has been contacted on numerous occasions when the wrong DLLs had been used for a particular version of Omnis Studio causing crashes, runtime errors, or less obvious compatibility issues. Some time ago we took the decision to alter our component version numbers so that one can identify the version of Studio for which a DLL was built.

All our components will be gradually updated as part of standard maintenance. The release notes for each product that has been updated will state so and link to this document.

This document will explain the new format, and how one can identify the DLL to prevent compatibility issues that may circumvent your internal QA. At the end of this document we will list the current supported versions of Studio and how they relate to our new version numbers. Included will also be a list of our products which have been updated to the new format. Consequently, this document will be updated on a regular basis as we release new versions of our products or when Tiger Logic releases new versions of Studio that require new builds.

IMPORTANT DOCUMENT CHANGES:

We have updated the supported studio versions table for Studio 8.0. Make sure you study this table if you are using this version.

The old format

All our Studio software now uses the new version number format thus we have removed instructions regarding old version numbers.

The new format

The new version number format includes the major and minor version of Omnis Studio for which the DLL was built and the three digit component version consisting of major, minor and patch digits. The complete version number will be “ab.cde” where ‘a’ and ‘b’ are the Studio major and minor digits and ‘c’, ‘d’ and ‘e’ are the component’s major, minor and patch digits. For example the patch release “3.5.1” for Studio 5.2 will be returned as “52.351” by $version. The same component when built for Studio 4.3 will return “43.351”.

Note: Version number strings displayed by the system may have a slightly different format. These strings will typically display the component’s version number but further info may display the Studio version number in parenthesis.

Checking the version programatically

It is possible to programmatically check component version numbers using the Omnis notation **$components.*component_name*.$version**. We strongly recommend that Omnis Studio applications always check the version numbers of Brainy Data external components against the minimum that is required for the application to operate correctly. The best time to check the version is during startup, perhaps in the startup task. Most of our examples provide sample code for this.

Example:

```
; this example tests for OWrite version 3.8.0 or better and
; Studio 5.2 (assuming that we are running a compatible Studio
; version 5, see section “Supported Studio versions”)
Calculate required_version as "52.380"
; check Studio compatibility
If ( int(required_version) <> int($components.OWrite.$version) )
  OK message  {OWrite is incompatible with this version of Studio}
End If
```

This example code is very simplistic and it requires the developer to update this code when switching between different versions of Omnis Studio. The section “Supported Studio versions” demonstrates a more sophisticated approach by checking the studio version utilizing the **sys(1)** function.

Checking the file version of the DLL

Brainy Data external component DLLs include system resources on Macintosh and Windows that allows one to inspect the version number of the DLL in the file system. Unfortunately, due to

technical limitations, these version numbers will only display the three digit component version number, but further system version info strings (typically following the copyright) may now display the Studio version number for which the component was build inside parenthesis.

To view the DLL version number on Macintosh, right-click the external component bundle and select “Get Info” from the context menu. The info window will display the version number as part of the Brainy Data copyright message.

On Windows, right-click the external component DLL and select “Properties”. On the properties window select the “Version” tab (“Details” tab in Windows 7). Both the “Copyright” and “Product Version” will display the component’s version number.

You may notice a capital letter ‘R’ or ‘D’ following the version number. The ‘D’ stands for time limited demo build and the ‘R’ stands for full release build which you will only receive when you have licensed our software.

Supported Studio versions

We do not build different DLLs for every version of Studio. Below is the list of separate builds that we provide and the versions of Studio that each build covers. This list will be updated as and when Tiger Logic release new software that requires new builds.

IMPORTANT NOTE: Since version 4.1.2.3, OWrite now has four digit version numbers following the decimal separator.

Component Version	Studio Versions Supported
43.xxx(x)	All Studio 4 versions after and including version 4.3.0
50.xxx(x)	All Studio 5 versions prior to version 5.2.0
52.xxx(x)	All Studio 5 versions after and including version 5.2.0
60.xxx(x)	All Studio 6 versions prior to version 6.1.0
61.xxx(x)	All Studio 6.1 versions prior to version 8.0 IMPORTANT NOTE: On MS Windows make sure to use the appropriate 32bit or 64bit DLLs.
80.xxx(x)	All Studio 8 versions prior to version 8.1.0 IMPORTANT NOTE: On MS Windows make sure to use the appropriate 32bit or 64bit DLLs. Studio 8 on Macintosh is 64bit only.
81.xxx(x)	All Studio 8.1 versions prior to version 10.0
100.xxx(x)	All Studio 10.0 versions prior to version 10.1

Component Version	Studio Versions Supported
101.xxxx	All Studio 10.1 versions (until further notice)

When downloading demo software from our demo download page or release software from our support pages, the downloaded folders containing the DLLs are arranged according to the Studio version. For example the fat-client component of OWrite version 3.8.0 for Studio 5 versions prior to version 5.2 on Macintosh is located in the folder

```
owrite_380r_mac/studio_500/xcomp
```

Studio version 5.2 components would be located in

```
owrite_380r_mac/studio_520/xcomp
```

Armed with the above table, we can now programmatically check the component's version numbers while catering for all currently known versions of Studio. When new versions of Studio are used that are currently unknown, the code will display an error. The following is an adaptation of the previous example code in the section "Checking the version programmatically".

Example:

```
; get the studio major and minor version digits using sys(1)
Calculate actual_studio_version as
                            con( mid(sys(1),1,1), mid(sys(1),3,1) )
; apply the above table
If (actual_studio_version = "100")
  Calculate required_studio_version as "100"
Else if (actual_studio_version >= "81")
  Calculate required_studio_version as "81"
Else if (actual_studio_version >= "80")
  Calculate required_studio_version as "80"
Else if ( actual_studio_version >= "61" )
  Calculate required_studio_version as "61"
Else if ( actual_studio_version >= "60" )
  Calculate required_studio_version as "60"
Else if ( actual_studio_version >= "52" )
  Calculate required_studio_version as "52"
Else if ( actual_studio_version >= "50" )
  Calculate required_studio_version as "50"
Else if ( actual_studio_version >= "43" )
  Calculate required_studio_version as "43"
End if
; now we can calculate our required version for OWrite
; (this example tests for OWrite version 3.8.0)
Calculate required_version as con(required_studio_version,".380")
; check Studio compatibility
If ( int(required_version) <> int($components.OWrite.$version) )
  OK message  {OWrite is incompatible with this version of Studio}
End If
```

Updated External Components

The following is the current list of external components that employ the new version number format. The “Version” column displays the version number in which the new format was introduced. This table will be updated as we release new versions of our software that implement the new version format.

IMPORTANT NOTE: Since version 4.1.2.3, OWrite now has four digit version numbers following the decimal separator.

Component	Version	Example Versions (Studio Interface)
OWrite JS-OWrite	4.2.0.0	43.4200 - 50.4200 - 52.4200 - 60.4200 - 61.4200 80.4200 - 81.4200 - 100.4200
PDFDevice	3.3.0	43.330 - 50.330 - 52.330 - 60.330 - 61.330 80.330 - 81.330 - 100.330
OSpell2	3.2.4	43.324 - 50.324 - 52.324 - 60.324 - 61.324 80.324 - 81.324 - 100.324
OGantt	4.1.0	43.410 - 50.410 - 52.410 - 60.410 - 61.410 80.410 - 81.410 - 100.410
OCal	2.1.1.	43.211 - 50.211 - 52.211 - 60.211 - 61.211 80.211 - 81.211 - 100.211 (please also read v1.7.0 release notes)
JS-Signature	1.1.1	60.111 - 61.111 80.111 - 81.111 - 100.111

Document History

09 April 2020: Updated for Studio 10.1
02 July 2019: Updated for Studio 8.1 and 10.0
08 June 2016: Updated for Studio 8.0
02 September 2015: Updated for Studio 6.1
05 September 2013: Updated for OGantt version 4
14 August 2013: Updated for PDFDevice version 3
29 April 2013: minor grammatical corrections and signpost to OCal release notes
25 April 2013: first publication