Release Notes

Go Up to What's New

These Release Notes contain important information that might not appear in the main product documentation. We recommend that you read this page in its entirety.

Contents

Before you install, uninstall, or upgrade the product, read the following files, for your chosen language (`xx` can be en, de, fr or ja):

- `radstudio_install_xx.htm` : lists the system requirements, as well as installation and upgrade procedures.
- `radstudio_license_xx.rtf` : contains your Software License and Support agreement.
- `radstudio_deploy_xx.htm` : contains information about deployment.

Read the latest version of the `radstudio_install_xx.htm` file, available at the following locations:

- **Installation Notes (http://docwiki.embarcadero.com/RADStudio/Sydney/en/Installation_Notes)**.
- The installation directory. The default location is: `C:\Program Files (x86)\Embarcadero\Studio\21.0`

Offline Installer

- Download the installer and the `.gof` file to the same folder
- Start the installler

It will automatically detect the offline mode and it will use the offline image to install the RAD Studio features.

If you use the offline installer, RAD Studio will not be able to access the GetIt Package Manager until you manually switch to "online mode". To do this, we recommend using the GetItCmd.exe tool: `GetItCmd.exe -c=useonline`

Previous Versions and Migration

RAD Studio Includes Licenses for Previous Versions

Licenses for previous versions of the product are included with your 10.4 Sydney license.

- Delphi 10.4 Sydney includes licenses for Delphi Rio, Tokyo, Berlin, Seattle, XE8, XE7, XE6, XE5, XE4, XE3, XE2, XE, 2010, 2009, 2007, and 7.
- C++Builder 10.4 Sydney includes licenses for C++Builder Rio, Tokyo, Berlin, Seattle, XE8, XE7, XE6, XE5, XE4, XE3, XE2, XE, 2010, 2009, 2007, and 6.

For more information about licenses, see the `radstudio_license_xx.rtf` file in the RAD Studio installation directory.

Upgrading Projects When Opening

When using RAD Studio 10.4 Sydney to open a project that was created with a previous version of RAD Studio, RAD Studio automatically upgrades the project. Upgraded projects are no longer compatible with previous versions of RAD Studio.

Always create a backup of a project before you open it with a newer version of RAD Studio.

Firewalls May Block GetIt Installation Files

If you have a firewall blocking .7z files, GetIt installation will fail.

Windows

DLLs are exporting RTL methods

When linking a 64-bit application to a DLL's import library in the Project Manager, the library is placed on the linker's command line, connecting the linker to the `__CPPdebugHook` symbol in the DLL's import library, rather than its own internal RTL symbol. To prevent this from happening, link to the import library using the `#pragma link` or `#pragma comment` and remove the reference to the import library in the Project Manager. This forces the linker to look at the symbols in the import library at the end, hence linking to the correct `__CPPdebugHook` symbol in it's own RTL.

Issues when rebuilding an MSBuild for a second time

When rebuilding an MSBuild with or without the additional resource file for a second time, the check on whether to `build` the project version resource file or not happens before the `clean` step of rebuild.

To prevent this from happening, perform a `clean` followed by a `build` :

```
> msbuild /t:clean project.cbproj

> msbuild /t:build project.cbproj
```

Linux

Wrong PAServer Error Message shown on Ubuntu Server 16.04 LTS

Enabling **Use launcher application** in the Run|Parameters dialog uses xterm as the default launcher application for Linux targets. When running PAServer on Ubuntu Server 16.04 LS, an error message is shown indicating that a file cannot be created. The error message is incorrect since it cannot launch xterm on Ubuntu Server 16.04 LTS which is expected. Xterm can be installed separately and is part of the client distributions of Ubuntu.

FireMonkey

Names Must Be Assigned to All Components Before Creating a New View in the Form Designer

If you open an older FireMonkey app in 10.4 Sydney, you might encounter the following error message:

```
 Cannot inherit from form 'FormName".  It contains a component
with a blank name property.
```

The Views Inheritance system requires that you assign a name to all components before you can create a new View in the Form Designer.

macOS 64-bit

Inspecting/evaluating structures on macOS 64-bit

Structures passed as parameters by value into a method on macOS64 may display the wrong values in the debugger.

Expanding changed inspected values on macOS 64-bit

A variable which can be expanded when inspected to see the values of fields inside it will not show updated sub-values in the debugger UI after the value of the variable has changed.

Evaluation on macOS 64-bit is case-sensitive

For technical reasons, evaluation on macOS 64-bit (such as expressions in the Evaluate/Modify dialog) is case-sensitive. This is the same as iOS 64-bit. You may also need some C++-style syntax in some situations, such as using a 0x prefix for hex values instead of $.

FireDAC

Using Run-Time Packages with C++ FireDAC DataSnap Driver

If you use the FireDAC DataSnap driver in 64-bit C++ applications built with run-time packages, you might receive an error at run time. To avoid the error, you need to add DataSnapClient to the run-time packages, following these steps:

1. **Project > Options > Packages > Runtime Packages**.
2. In the **Runtime package import libraries** field, add **DataSnapClient** to the list.

This setting resolves the problem and adds DataSnapClient.bpi to the link line rather than DataSnapClient.a.

64-bit Windows C++ Application with a FireDAC Component Might Raise Error

The following error is raised when attempting to compile an application using a single FireDAC component, when the platform is Windows x64 with static linking:

```
c:\program files
(x86)\embarcadero\studio\<n.n>\Bin\CodeGear.Cpp.Targets :
warning : Warning: Out of memory
c:\program files
(x86)\embarcadero\studio\<n.n>\Bin\CodeGear.Cpp.Targets(2751,5)
: error MSB6006: "ilink32" exited with code 2.
```

The workaround is not to link FireDAC statically in Win64 C++ apps.

Database

Repopulating datasets

Repopulating a bound `TClientDataSet` or `TFDMemTable` may be slower than the initial time. To work around this issue, we have added a new `EmptyDataLink` method to `TBindSourceDB`.

Android

JDK Path Issues

Due to a bug (https://code.google.com/p/android/issues/detail?id=147561) of the Java Development Kit (JDK), the deployment path of a project must not contain non-ASCII characters for the Android target platform.

Patching `string.h` file for Android

While using the C++ Builder for creating an FMX application for Android, especially if you use the Offline Installer. Please follow these steps to patch the `string.h` file manually:

1. Double click the error message.
2. The IDE opens the `string.h` file.
3. Go to the line containing the following comment:

```
/* Const-correct overloads. Placed after FORTIFY so we call those
functions, if possible. */
```

4. Change the line after the comment mentioned above as shown below:

Before	`#if defined(__cplusplus) && defined(__clang__)`
After	`#if defined(__cplusplus) && defined(__clang__) && ((__clang_major__ > 3) || (__clang_minor__ > 4))`

Please refer to the example below as it displays how does your code should change when generating a `.diff`:

```
--- string.h.orig
+++ string.h.modified
@@ -188,7 +188,7 @@
```

```
 #endif

 /* Const-correct overloads. Placed after FORTIFY so we call
those functions, if possible. */
-#if defined(__cplusplus) && defined(__clang__)
+#if defined(__cplusplus) && defined(__clang__) &&
((__clang_major__ > 3) || (__clang_minor__ > 4))
 /*
  * Use two enable_ifs so these overloads don't conflict with +
are preferred over libcxx's. This can
  * be reduced to 1 after libcxx recognizes that we have const-
correct overloads.
```

Android Manifest File Updates in 10.4

New FireMonkey projects created in a new directory do not need the step mentioned below for using the built-in IDE support for the Android platform. If you have an existing FMX project for the Android platform from a previous RAD Studio version, you need to delete AndroidManifest.template.xml from your project directory and it will be recreated automatically in 10.4.

iOS

Working with the iOS Storyboard

For new FireMonkey projects created in a new directory, no additional steps are required for using the built-in IDE support for the storyboard.

If you have an existing FMX project for the iOS platform from a previous RAD Studio version, you need to delete the info.plist.TemplateiOS.xml from your project directory.

There are two different steps to customize the storyboard launch screen support for iOS in 10.4, depending on whether you're using your own compiled storyboard or starting with a new one using what's provided in 10.4.

Option 1) User supplies his own compiled storyboard and asset catalog (*.storyboardc and Assets.car). In order to use that with 10.4, follow the steps below:

- Uncheck *.launchscreen from being deployed in Project | Deployment
- If the name of the storyboard file is not LaunchScreen.storyboard, you will need to edit the info.plist.TemplateiOS.xml, and replace the "<%StoryboardInfoPListKey%>" with the updated "UILaunchStoryboardName key/value"
- Open Project | Deployment and add the compiled storyboard and asset catalog files

Option 2) User modifies the default storyboard and asset catalog provided by the IDE. If you're using this option, follow the steps below:

- Edit the LaunchScreen.storyboard and Assets files in LaunchScreen.TemplateiOS as needed
- Open Project | Deployment and add additional launch screen images to be deployed if any

The info.plist.TemplateiOS.xml and LaunchScreen.TemplateiOS templates can be found in the following location:

```
a. <bds>\ObjRepos\<locale>\iOS
b. %APPDATA%\Embarcadero\BDS\21.0
c. the location where the project is located.
```

The IDE looks for these templates in the following order: Project directory, %APPDATA% directory, and ObjRepos directory.

C++ Toolchains

Specifying Dependent and Required Packages When Using a Component in a C++ Package

When using an existing component in a C++ package, the IDE does not automatically add the dependent (required) package for the component. You need to perform this step manually.

Without the relevant dependent (required) packages, the C++ package might fail to link because the linker cannot locate .obj files that must be initialized when that component is used. To avoid such linker errors, you need to add the package of the component and its dependencies to the **Requires** node of your C++ package (in the Projects Window).

The following example illustrates how to determine the required packages for a component and eliminate the **Unable to open** 'xxx.obj linker errors by creating a dummy VCL Forms Application that uses the component.

By default, a newly-created C++ Multi-Device Application links with the .rtl and .fmx packages, as shown by the *.bpi files listed on the linker line (viewable in View > Messages):

Ilink32 command line:

```
c:\BDSLoc\bin\ilink32.exe ... -aa -V5.0 -Tpe  c0wfmx32w rtl.bpi
fmx.bpi memmgr.lib sysinit.obj .\Win32\Debug\Project30.obj
.\Win32\Debug\Unit26.obj , .\Win32\Debug\Project30.exe ,
.\Win32\Debug\Project30.map , import32.lib cp32mti.lib , ,
Project30.res
```

When you add new components, the IDE calculates dependent packages and adjusts the link line. To see this, add the component you want to use in a C++ Package to the Multi-Device Form. This example uses the component TFDSchemaAdapter. Now rebuild the application. You will see that several new .bpi files have been added to the link line:

Ilink32 command line:

```
c:\BDSLoc\bin\ilink32.exe ... -aa -V5.0 -Tpe  c0w32w rtl.bpi
fmx.bpi FireDACCommonDriver.bpi FireDACCommon.bpi xmlrtl.bpi
dbrtl.bpi FireDAC.bpi FireDACSqliteDriver.bpi memmgr.lib
sysinit.obj .\Win32\Debug\Project30.obj
.\Win32\Debug\Unit26.obj , .\Win32\Debug\Project30.exe ,
.\Win32\Debug\Project30.map , import32.lib cp32mti.lib , ,
Project30.res
```

These new .bpi files are the packages on which your component depends. In the case of TFDSchemaAdapter, the .bpi files are:

```
FireDACCommonDriver.bpi FireDACCommon.bpi xmlrtl.bpi dbrtl.bpi
FireDAC.bpi FireDACSqliteDriver.bpi
```

So to use the **TFDSchemaAdapter** component in a C++ Package, you need to explicitly add each one of these bpi files as 'Required' references in the IDE (see Packages (Delphi)).

Downloading DirectX Header Files

We only ship DirectX headers inside the Microsoft Windows Platform SDK. If you encounter errors such as "**d3d.h file not found**", the HPP generated for that Delphi unit contains `#include <D3D*.hpp>`. You can do either of the following:

- Add a `{$NOINCLUDE Winapi.D3DX9}` directive to the unit that uses the D3D unit and regenerate the HPP file of that unit. The regenerated HPP will not contain the `#include <Winapi.D3DX9.hpp>`.
- Download the DirectX SDK that the D3D header relies upon. You can download the DirectX SDK for free from Microsoft. For more information, see Where is the DirectX SDK? (Windows) (http://msdn.microsoft.com/en-us/library/windows/desktop/ee663275(v=vs.85).aspx).

IDE

Renamed iOS Project Might Fail at Run Time or When Debugging

Renaming a project in the IDE can cause an iOS app to fail at run time and debug time. To correct the issue, do the following:

1. Select **Project > Deployment**.
2. In the Deployment Manager, click the **Revert To Default** speed button.

Version Info Settings Inheritance

To include version information in a project targeting Windows, do not change the key values on the All Platforms target. Instead, change the values individually for each platform (32-bit and 64-bit). The targets do not inherit the values properly from the All platforms target.

InterBase

InterBase 2020 Edition in RAD Studio 10.4 Sydney

This note pertains to users who have both an earlier version of RAD Studio and RAD Studio 10.3 Rio installed on the same machine. In this note, the mention of RAD Studio also implicitly includes Delphi and C++Builder editions.

Each version of RAD Studio includes a license for a version of InterBase:

Product	InterBase Version
RAD Studio XE3	InterBase XE3 Developer Edition
RAD Studio XE4	InterBase XE3 Developer Edition
RAD Studio XE5	InterBase XE3 Developer Edition
RAD Studio XE6	InterBase XE3 Developer Edition
RAD Studio XE7	InterBase XE3 Developer Edition
RAD Studio XE8	InterBase XE7
RAD Studio XE9	InterBase XE7
RAD Studio Berlin	InterBase XE7
RAD Studio Tokyo	InterBase XE7
RAD Studio Rio	InterBase 2017
RAD Studio Sydney	InterBase 2020

Since all these RAD Studio license suites are visible system-wide, only one license of InterBase can be used at any time.

For example, if you are running the "InterBase XE3 Developer Edition" that comes with RAD Studio XE6, you cannot start a simultaneous instance of the "InterBase XE3 Developer Edition" that comes with RAD Studio XE7. You cannot start a simultaneous instance of the InterBase 2020 edition that comes with RAD Studio 10.4 Sydney either. When you try, you receive an error dialog that states "InterBase licensing error", and the InterBase log shows "Registration file error: License is in use by another instance of InterBase".

For more information about running multiple instances of InterBase, see **Multiple Instances** section in the Operation Guide (http://docwiki.embarcadero.com/images/InterBase/2020/e/3/30/OpGuide.pdf) document.

Workaround for InterBase Licensing Errors

1. Stop all your instances of InterBase. Also, if you have set up this instance as a Windows Service, please disable it in the system Service Control Panel.
2. Start the instance of InterBase that you want to use. It should now launch successfully with the proper license.

The aforementioned RAD Studio versions and the applications built by them, can also work with the updated InterBase 2020 edition installed with RAD Studio 10.4 Sydney. Have your older IDE tools and applications connect to your database via TCP loopback to this InterBase instance. For example:

```
localhost/gds_db:<dbpath>
```

In older versions of RAD Studio, you might also want to select Tools > Options > Environment Options > Environment Variables and add the following new "User overrides" entries for making local client connections.

Variable	Value
`IB_Protocol`	`gds_db`
`InterBase`	`C:\Program Files (x86)\Embarcadero\Studio\21.0\InterBase2020`

Long Term Known Issues

HP Computers include an environment variable named **PLATFORM** which if set to any other value other than "**Win32**", "**Win64**", or "**OSX32**" causes the following error when compiling:

Invalid PLATFORM variable "(value)". PLATFORM must be one of the following: "Win32", "Win64", "Android32Arm", "iOSSimulator32", "iOSDevice32" "iOSDevice64", or "OSX32", or "OSX64", or "Linux64".

Follow this steps to remove the PLATFORM environment variable from your system:

1. On your desktop, right click on **My Computer**.
2. Select **Properties**.

3. Depending on your OS, select either the **Advanced** tab, or **Advanced system settings**
4. Click the **Environment Variables** button
5. Find the Platform environment variable and click the **Delete** button

External Software

Issues might occur for applications debugged in an environment that includes a third-party keyboard switcher such as the Yandex Punto Switcher. If the application is closed by Windows, and an access violation is raised by the application, please ignore this access violation error.

Translation Tools

As of RAD Studio 10.3 Rio the translation tools available in the product, both for VCL (Integrated Translation Editor) and for FireMonkey (TLang), are officially deprecated and removed from active support. Both features are still available in Rio but it is recommended to reduce your project reliance on them. RAD Studio is teaming with third-party vendors to offer alternative options.

See Also

- Installation Notes
- New features and fixed issues