Contents

- Requirements
- Getting started
- Queries
- Release build
- Other known issues
- Additional resources

You can build Windows 10 Universal apps beginning with DevForce 7.4.0. We'll show you how to get started here.

First, some nomenclature. We generally use "UWP" (for Universal Windows Platform) to indicate a Windows 10 Universal app. You'll also sometimes see the abbreviation "UAP" (Universal App Platform) in other documentation.

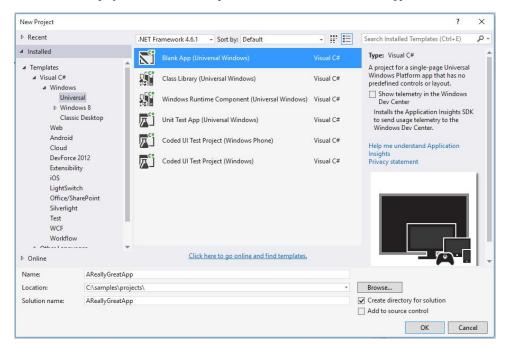
Requirements

- Windows 10
- Visual Studio 2015 Update 1 with Visual Studio Tools for Universal Windows Apps Although UWP apps can be developed with any edition of Visual Studio 2015, DevForce 2012 requires the Professional edition or above.
- Windows 10 SDK This will be installed with VS 2015 Update 1.
- To build a UWP app with DevForce you'll need either a "Universal" or "Windows Store" license.

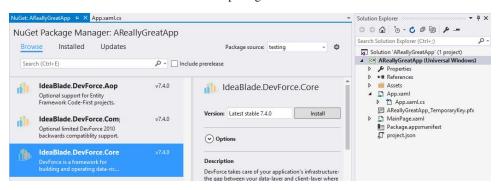
Getting started

DevForce does not offer a project template to begin development of a DevForce UWP app, but it's simple to get started.

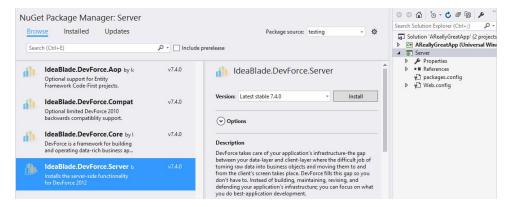
1. Create a new project and select the template for Windows Universal Blank App:



2. Install the IdeaBlade.DevForce.Core NuGet package:

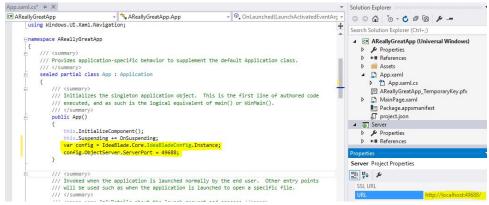


3. Like other DevForce mobile environments, DevForce UWP apps are n-tier and require an *EntityServer*. The most common way to implement this is to add an empty web application project to your solution, and install the IdeaBlade.DevForce.Server package to it.



4. Configure the app to communicate with the *EntityServer*.

You can use either an app.config or simple programmatic configuration. During development, it's easy to get started by specifying only the server port, allowing other settings to default, like we do here in the app startup logic:



(Hmm, Someone doesn't know how to use a highlighter.)

5. See the <u>development guide</u> for what you can do next.

Queries

- DevForce uses <u>QueryStrategy.Normal</u> by default when querying. This translates into an <u>Optimized</u> fetch strategy, which generally means that DevForce will determine how best to satisfy the query using local cache, the data source, or a combination of the two. In UWP apps, at this time queries may not be executed against both cache and the data source, which means that a <u>FetchStrategy</u> of <u>DataSourceThenCache</u> or <u>DataSourceAndCache</u> is not supported.
 In UWP apps, <u>QueryStrategy.Normal</u> means that DevForce will use the <u>DataSourceOnly</u> fetch strategy when the <u>EntityManager</u> is connected and the <u>CacheOnly</u> fetch strategy when disconnected. You can still use custom query strategies, but an exception will be thrown if you use an unsupported fetch strategy.
- There is reduced support for the *Nullable*<*T*> type in UWP apps. Most importantly, the List<Nullable<*T*>> cannot be used in queries. For example, you might want to use a List<int?> when using a query with a contains clause:

```
var pids = new List<int?> {1, 2, 3};
var query = _entityManager.Products
.Where(p => pids.Contains(p.ID));
```

This will fail, however, because the List<int?> cannot be serialized. To work around this restriction, use a List for the non-nullable value type, in this case List<int>.

Note that you will only encounter an error when doing a *release* build or enabling the .NET Native tool chain. You may also encounter other edge conditions when using Nullable<T> in release builds. It's very important to enable the .NET Native tool chain at times during development and testing to ensure there will be no unpleasant surprises at release.

Release build

When building a UWP app in release mode the .NET Native tool chain is used to compile your app to native code for the target platforms specified. This build removes most dependencies on external runtimes and libraries and heavily optimizes code for maximum performance. To ensure your DevForce app still works in .NET native you must use runtime directives to specify its serialization and reflection requirements.

Specifically, for every entity type in your model you must include a Type directive for List<TEntity>, RelatedEntityList<TEntity> and EntityQueryProxy<TEntity>. This is required even if you never query or save the entity.

For example, here's the runtime directive file, Default.rd.xml, for an app whose entity data model contains an entity type named "Customer":

Other known issues

- Code First models are not currently supported.
- Projection into an anonymous type is not supported. See the return part of an entity topic for how to work around this.

Additional resources

- DevForce Windows Universal app tour
- Get Started with Windows Apps
- Build UWP Apps with Visual Studio
- Getting Started with .NET Native
- How-to Guides for UWP apps
- Runtime Exceptions in .NET Native apps
- Runtime Directives Reference