# ArcGIS Runtime and Xamarin

By Euan Cameron and Rex Hansen, Esri ArcGIS Runtime Team

Over the last year, there have been a number of requests from developers for information regarding Xamarin technology in relation to the ArcGIS Runtime.

Xamarin is attractive to many of our developers because it allows them to leverage their existing Microsoft .NET development skills when building Android, iOS, Mac, Windows, and Windows Phone apps.

## What Is Xamarin?

Xamarin is an evolution of the Mono project, which is a cross-platform, open-source .NET development framework. When Microsoft created the .NET technology, it did two things: it defined a specification and implemented that specification for the Windows operating system. The specification defined and subsequently published was, in fact, composed of several specifications. Microsoft implemented these and released them to the world as the Microsoft .NET Framework. Microsoft, Intel, and Hewlett-Packard then worked to standardize the specifications, which was accomplished in 2003.

## Xamarin Implementation

The situation with user interface (UI) components is more complex. Xamarin Forms, released in May 2014, is a technology similar to Microsoft's XAML that allows developers to capture a form layout within a visual designer. Xamarin will bind UI components in the layout to native UI widgets appropriate for the platform the UI is executing on.

Since the various platforms that Xamarin supports have very different UI capabilities, a subset of UI controls are supported by Xamarin Forms. Xamarin Forms can coexist with native UI controls that are also accessible through Xamarin, although when using the native controls the UI code is not portable between platforms.

Third-party components that extend the capabilities of the operating system, such as ArcGIS Runtime, can expose their capabilities to Xamarin by creating platform-specific Xamarin bindings or enabling access to functionality in C/C++ libraries.

Xamarin bindings are a mechanism by which Objective-C or Java components can be utilized from .NET code. Since code for a binding depends on a platform-specific library, the binding code itself is platform specific. The .NET API on top of the binding will usually match the underlying structure of the library but can be adjusted. One major benefit of this approach is that Xamarin tools are provided to autogenerate bindings for a given library.

On the other hand, building functionality in C/C++, compiling for the Mono platform, and invoking from .NET code also enable sharing logic across platforms. Since the C/C++ source code is the same, the .NET API surface is the same regardless of the target platform (e.g., iOS, Android). Since Xamarin technologies rely on Mono, these native libraries can be used in Xamarin solutions. While this provides the most refined and consistent developer experience, it does require managing interops and referencing platform-specific dependencies to build a native library.

## ArcGIS Runtime and Xamarin in Practice

To expose the capabilities of ArcGIS Runtime to the Xamarin developer today, Xamarin bindings need to be created. These Xamarin bindings must be created for each platform by binding onto the appropriate ArcGIS Runtime API for the respective platform.

While this architecture allows Xamarin developers to access the ArcGIS Runtime, it does not present the developer with a way to easily write cross-platform code, since the differences in ArcGIS Runtime APIs are exposed directly to the developer. Developers must learn the specific ArcGIS Runtime API, and with that knowledge, they can code against that API using C#. In addition, since the bindings map very closely to the underlying API, as the API evolves, the bindings must be maintained. This can be a time-consuming task.

To provide a more consistent and stable experience for ArcGIS Runtime .NET developers across the Windows, iOS, and Android platforms, Esri needs to build an API for Xamarin that invokes native capabilities in the C++ core that underlies ArcGIS Runtime. No bindings will be required, but the process will take time. With that in mind, Esri plans to have a commercial release of ArcGIS Runtime supporting Xamarin in the latter half of 2015. Esri is also seeking to provide a beta version in the first half of 2015.

## What Can You Do Now?

Developers can create their own Xamarin bindings on the current ArcGIS Runtime SDKs for iOS and Android. While this is technically possible, Esri will not directly support any development work via third-party Xamarin bindings. Of course, Esri will continue to support use of the underlying native API.

Keep in mind, third-party bindings will need to be re-created when Esri releases updates to the underlying ArcGIS Runtime SDKs. This will be required to enable developers to take advantage of new features or pick up bug fixes and performance enhancements. Also consider that any application code written against third-party bindings will need to be rewritten to work with Esri's commercial product that supports Xamarin next year.

## What Should You Do Now?

If possible, Esri recommends waiting until the official release of an ArcGIS Runtime product that supports Xamarin. If you are unable to wait, please take into account the time and effort that will be necessary to migrate ArcGIS Runtime applications you create today to Esri's commercial product available later this year.