F# Versioning in dev11

# Inventory of F# 2.0 Releases and Compatibility Spec

In 2010, alongside VS2010, we shipped F# 2.0. We have multiple sets of tools targeting F# 2.0, shipped on different release vehicles:

* **F# 2.0 VS2010 Pro** (April 2010). Our flagship, official, supported release, in the box with VS2010.
  + Users can target .NET 2.0 (+ 3.0, 3.5), 4.0, Silverlight
    - Silverlight 2&3 out of the box
    - Silverlight 4 support included with VSTools for Silverlight 4, an official VS2010 pack to support Silverlight (Not updated for VS2010 SP1)
* **F# 2.0 CTP** (November 2010). Our free-as-in-beer F# offering. Includes: barebones compiler (installable on Mono), tools for VS2008 (shell & Pro+), tools for VS2010 Integrated Shell
  + Users can target .NET 2.0 (+ 3.0, 3.5), 4.0, Silverlight (2 &3), Compact Framework, Windows Phone
* **F# 2.0 OpenSource** (TBD). Our free-as-in-speech F# offering. Includes: barebones compiler, all runtimes included in F# 2.0 CTP (in source form).
* **F# 2.0 SP1 CTP (April 2011) –** Includes updated compiler and runtime DLLs for Parity with VS2010 SP1. Also includes SL4 and SL5 runtimes.

## Compatibility – language

All three F# 2.0 releases support the same version of F# language – F# 2.0

## Compatibility – horizontal (same target, different releases)

For the same target platform, binaries produced by F# 2.0 VS2010Pro and F# 2.0 CTP are entirely binary compatible and interoperable.

*Note:this decision impacts (and greatly simplifies) vertical compatibility story. We do not need to consider vertical compat for different F# releases.*

## Compatibility – vertical (different targets)

F# 2.0 programs targeting .NET 2.0 (+ 3.0, 3.5) can be run on .NET 4.0. For all other platforms, we do not have any form of compatibility – the users will need to recompile their binaries if they want to target different versions of Silverlight or Compact Framework. This policy is consistent with policies on respective target frameworks.

## Compatibility – project files

In general, project files cannot be shared between VS2008 and VS2010. In F# 2.0 VS2010Pro release includes required engineering to support using VS2010 F# project files in VS2008. However, in the absence of general VS round-tripping support between VS2008 and VS2010 this feature is limited.

## SxS Requirements

VS2008 Shell/Pro and VS2010 Shell/Pro tooling should install side-by-side seamlessly. SxS Installation of VSxyzw Pro and VSxyzw Shell is a non-requirement.

# Inventory of F# 3.0 releases and Compatibility Spec

The next version of F# language, F# 3.0, will ship in sync with VS dev11 (called VS2012 elsewhere in this document)

* **F# 3.0 VS2012 Pro**. Our flagship, official, supported release, in the box with VS2012.
  + Users can target 2.0 (+3.0, 3.5), 4.0, 4.5
* **F# 3.0 Free Tools release**. Our free-as-in-beer F# offering. Includes: barebones compiler (installable on Mono), VS2010 (Shell + Pro+), VS2012 Integrated Shell
  + Users can target 2.0 (+3.0,3.5), 4.0, 4.5
* **F# 3.0 OpenSource**. TBD

## Compatibility – language, horizontal

* F# 3.0 releases for .NET 4.0 (and by association, 4.5) can make use of queries and Type Providers
* F# 3.0 releases for .NET 2.0 **cannot** make use of queries and Type Providers

## Compatibility – language, backward

We expect F# 2.0 programs to be compileable with F# 3.0 compiler with the same semantics. We might take some minor and heavily considered breaks.

Interesting possibility to consider is enabling using F# 3.0 compiler with F# 2.0 runtime. There is something immensely simplifying about our F# 2.0 design decision that “the characteristics of the compiler are dependent on the FSharp.Core and mscorlib you reference”.  That mostly covers .NET profile and F# codegen characteristics, but extending it to include the small set of things we expect to be covered by the F# 2.0/3.0 characteristic may help us keep the user picture simple.

We are not finalizing this decision yet but we are aiming for that in our compiler work

### Minor breaks

* const keyword (reserved keyword identifier became an actual language keyword)

## Compatibility – horizontal (same target, different 3.0 releases)

For the same target platform, binaries produced by F# 3.0 VS2012Pro and F# 3.0 CTP are entirely binary compatible and interoperable for a given .NET Framework version.

*Note:this decision impacts (and greatly simplifies) other compatibility stories. We do not need to consider compat for different F# releases.*

## Compatibility – vertical (different targets)

F# 3.0 programs targeting .**NET 2.0 (+ 3.0, 3.5) can be run on .NET 4.0/.NET 4.5**. F# 3.0 programs targeting **.NET 4.0 can be run on .NET 4.5**. For all other platforms, we do not have any form of compatibility – the users **will need to recompile their binaries if they want to target different versions of Silverlight or Compact Framework**. This policy is consistent with policies on respective target frameworks.

## Compatibility – backward (F# 2.0 on F# 3.0)

On .NET 2.0 (+3.0,3.5), 4.0, 4.5, it is consistent with general platform expectations to **maintain backwards compatibility**. All F# 2.0 programs should be able to run on F# 3.0 runtime, and to intoperate with F# 3.0 binaries.

On Silverlight/Windows Phone, backward compatibility is less of an expectation, particularly because every released application carries all its dependencies with itself.

**Action Item**: make decision on backwards compatibility for Silverlight/WP

## Compatibility – forward (F# 3.0 on F# 2.0)

The F# 2.0 runtime will not be forward-compatible with F# 3.0 (F# 3.0 programs will not run on F# 2.0 runtime).

## Compatibility – project files

The VS2010/VS2012 project compatibility is achieved by extending conditional logic for Microsoft.FSharp.targets placement.

VS2010 project files are backward compatible, in that opening them in VS2012 is possible; the project will undergo in-place conversion, which adds conditional logic so that the same project can be opened in either VS2010 or VS2012. A project file newly created in VS2012 will not work on VS2010 (due to the VS2012 project referencing the specific version FSharp.Core 4.3.0.0 – users could remove the specific version in the VS reference properties, in which case it behaves like an upgraded project that can be opened in VS2010, provided no new features (.Net 4.5-specific references for example) are used).

## SxS Requirements

Design time: VSabcd Pro/Shell and VSxyzw Pro/Shell tooling should work side-by-side. VSabcd Pro and VSabcd Shell SxS is a non-requirement.

# Versioned components of F# release – design time

1. F# language version number. Used in
   1. Banners: "Microsoft (R) F# x.y Compiler build %s"
   2. Reference Assembly target dir: C:\Program Files\Reference Assemblies\Microsoft\FSharp\x.y
   3. Registry key for F# runtime in reference assemblies: @"Software\Microsoft\FSharp\x.y\Runtime\v4.0"

F# 2.0 VS2010Pro: 2.0

F# 2.0 CTP: 2.0

F# 3.0 VS2012Pro: 3.0

F# 3.0 CTP: 3.0

1. F# compiler installation location in $(ProgramFiles)(non-VS bits):

F# 2.0 VS2010Pro: Microsoft F#\v4.0

F# 2.0 CTP: FSharp-$(build number)

F# 3.0 VS2012Pro: Microsoft SDKs\F#\3.0\Framework\4.0  
F# 3.0 CTP: FSharp-$(build number)

1. Build number (used in banners
   1. Microsoft (R) F# x.y Compiler build a.b.c.d

F# 2.0 VS2010Pro: assembly file version for compiler dlls.  
F# 2.0 CTP: 2.0.0.0  
F# 3.0 VS2012Pro: assembly file version for compiler dlls.  
F# 3.0 CTP: 3.0.0.0

Note: for interim beta CTPs we will use numbers between 2.0.0.0-3.0.0.0 (e.g. 2.9.7.1)

1. Assembly versions and assembly file versions
   1. VS dlls (FSharp.ProjectSystem &c) (version, file version)  
      F# 2.0 VS2010Pro: 4.0.0.0, 4.0.30319.1  
      F# 2.0 CTP: 2.0.50727.0, 2.0.50727.0  
      F# 3.0 VS2010Pro: 11.0.0.0, 11.0.dailybuild  
      F# 3.0 CTP: TBD
   2. Compiler dlls (version, file version)  
      F# 2.0 VS2010Pro: 4.0.0.0, 4.0.30319.1  
      F# 2.0 CTP: : 2.0.50727.0, 2.0.50727.0  
      F# 3.0 VS2012Pro: 4.3.0.0, 4.3.dailybuild  
      F# 3.0 CTP: TBD

# F# Runtime versions

Versions of F# runtime are identical across different releases of the same language version (see “Compatibility – horizontal”).

For F# 2.0, the versions we ship are as follows (assembly version, file version):

* .Net 2.0: 2.0.0.0, 2.0.30319.1
* .Net 4.0: 4.0.0.0, 4.0.30319.1
* Silverlight, *all versions*: 2.0.0.0, 2.0.30319.1
* Windows Phone, : 2.0.0.0, 2.0.0.0

*For F# 3.0, .Net 2.0 & 4.0:*

There is no forward compatibility. FSharp.Core for F# 3.0 is versioned as follows:

* .Net 2.0: 2.3.0.0, 2.3.0.dailybuild
* .Net 4.0: 4.3.0.0, 4.3.0.dailybuild
* Silverlight, Windows Phone: 3.0.0.0

**The “F# binary format revision number”:** the format used for optdata/sigdata

*Remains at 2.0.0.0.* As yet we don’t expect this to change: we have no plans as yet to rev. the binary metadata format and it would be really great if we could avoid it. Certainly we must be backwards compatible. This is an absolute requirement for forward compatibility.

## F# Runtime dependencies

F# 2.0:

.Net 2.0/3.0/3.5: Only depends on mscorlib.dll and System.dll, 2.0 subsets thereof

.Net 4.0: Only depends on mscorlib.dll, System.dll and System.Numerics.dll

F# 3.0

For queries language feature, we will take a dependency on System.Core. One question with that is whether we then forego .Net 2.0 (one option is to ship FSharp.Core for 2.0 in F# 3.0 CTP).

# Open items:

1. Review and possibly simplify version numbers – we currently have too many and they are not very consistent, particularly on CTP side.