X# Version history

*Note: When an item has a matching GitHub ticket then the ticket number is behind the item in parentheses prefixed with #. You can find these tickets by going to: https://github.com/X-Sharp/XSharpPublic/issues/**nnn** where **nnn** is the ticket number.*
If you find an issue in X# we recommend that you report it on GitHub. You will be notified of the progress on the work on your issue and the ticket number will be included in the what's new documentation

This document lists the changes to X# since build 2.12. For a complete list of changes that were made in earlier builds, please have a look at the help file.

Changes in 2.16.0.5

Compiler

New Features Xbase++ dialect

We have made several changes in the way how Xbase++ class definitions are generated. Please check your code extensively with this new build!

- We now generate a class function for all classes. This returns the same object as the ClassObject() method for Xbase++ classes.
  This class function is generated, regardless of the /xpp1 compiler option.
  The Class function depends on the function __GetXppClassObject and the XSharp.XPP.StaticClassObject class that both can be found in the XSharp.XPP assembly (#1235).
  From the Class function you can access class variables and class methods.
- In Xbase++ you can have fields (VAR) and properties (ACCESS / ASSIGN METHOD) with the same name, even with same visibility. Previously this was not supported.
  The compiler now automatically makes the field protected (or private for FINAL classes) and marks it with the [IsInstance] attribute.
  Inside the code of the class the compiler will now resolve the name to the field. In code outside of the class the compiler will resolve the name to the property.
- For derived classes the compiler now automatically generates a property with the name of the parentclass, that is declared as the parent class and returns the equivalent to SUPER.
- We have fixed an issue with the FINAL, INTRODUCE and OVERRIDE keywords for Xbase++ methods (#1244)
- We have fixed some issues with accessing static class members in the XBase++ dialect (#1215)
- You can now use the "::" prefix to access class variables and class methods inside class methods.
- When a class is declared as subclass from another class then the compiler generates a (typed) property in the subclass to access the parent class, like Xbase++ does. This property returns the value "super".
- We are now supporting the READONLY clause for Vars and Class Vars. This means that the variable must be assigned in the Init() method (instance variables) or InitClass() method (Class vars)

New Features other dialects

- Inside Visual Objects you could declare fields with the INSTANCE keyword and add ACCESS/ASSIGN methods with the same name as the INSTANCE field.
  In previous builds of X# this was not supported.

The compiler now handles this correctly and resolves the name to the field in code inside methods/properties of the class and resolves the name to the property in code outside of the class.

- The PPO file now contains the original white space from user defined commands and translates.

Bug fixes

- Fixed some method overload resolution issues in the VO dialect (#1211).
- Fixed internal compiler error (insufficient stack) with huge DO CASE statements and huge IF ELSEIF statements (#1214).
- Fixed a problem with the Interpolated/Extended string syntax (#1218).
- Fixed some issues with incorrectly allowing accessing static class members with the colon operator or instance members with the dot operator (#1219, #1220).
- Fixed Incorrect visibility of MEMVARs created with MemVarPut() (#1223).
- Fixed problem with _DLL FUNCTION with name in Quotes not working correctly (#1225).
- If the preprocessor generated date and/or datetime literals, then these were not recognized. This has been fixed (#1232).
- Fixed a problem with the preprocessor matching of the last optional token (#1241).
- Fixed a problem with recognizing the ENDSEQUENCE keyword in the Xbase++ dialect (#1243).
- Using a default parameter value of NIL is now only supported for parameters of type USUAL. Using NIL for other parameter types will generate a (new) warning XS9117.
  Also assigning NIL to a Symbol or using NIL as parameter to a function/method call that expects a SYMBOL will now also generate that warning (#1231)
- Fixed a problem in the preprocessor where two adjacent tokens were not merged into one token in the result stream. (#1247)
- Fixed a problem in the preprocessor where the preprocessor was not detecting an optional element when the element started with a Left parenthesis (#1250)
- Fixed a problem with interpolated strings that contained literal double quotes like in `i"SomeText""{iNum}"" "`
- Fixed a problem that was introduced in 2.16 with local functions / procedures.
- A warning generated at parse time could lead to another warning about a preprocessor define even when that is not needed. This has been fixed.
- Fixed issue with default parameter values for parameters declared as "`a := NIL,b := NIL as USUAL`" introduced in an earlier build of 2.16.
- Fixed issue with erratic debugger behavior introduced in an earlier build of 2.16.
- When you are referring to a type in an external assembly that depends on another external assembly, but you did not have a reference to that other external assembly, then compilation could fail without proper explanation. Now we are producing the normal error that you need to add a reference to that other assembly.
- Omitting types for parameter from a function that did not have the CLIPPER calling convention was allowed. Previously these parameters were assumed to be of type USUAL. This now produces a new error XS9118. If you want the parameters to be of type USUAL you will have to explicitly declare them with that type.

Breaking changes

- If you are using our parser to parse source code, please check your code. We have made some changes to the language definition for the handling of if ... else statements as well as for the case statements (a new condBlock rule that is shared by both rules). This removes some recursion in the language. Also, some of the Xbase++ specific rules have been changed. Please check the language definition online

Runtime

New Features

- Added the DOY() function
- Addeding missing ADS_LONG and ADS_LONGLONG defines
- Improved the speed of CDX skip operations on network drives (#1165)

Bug fixes

- Fixed a problem with DbSetRelation() and RLock() (#1226).
- Adjusted implicit conversion from NULL_PSZ to string to now return NULL instead of an empty string.
- Some initialization code is now moved from _INIT procedures to the static constructor of the SQLConnection Class, in order to make it easier to use this class from non-X# apps.
- Fixed an issue with the visibility of dynamic memory variables that were created with the MemVarPut function (#1223).
- Fixed a problem with the DbServer class in exclusive mode (#1230).
- Implicit conversions from NULL_PSZ to string were returning an empty string and not NULL (#1234)
- Improved the speed of CDX skip operations on network drives (#1165).
- Fixed a problem in the CTOD() function when the day, month or year were prefixed with spaces
- Fixed an issue with OrderListAdd() in the ADS RDD. When the index is already open, then the RDD no longer returns an error.
- Fixed an issue with MemRealloc where the second call on the same pointer would return NULL_PTR (#1248).

VOSDK

- Global arrays in the SDK classes are now initialized from the class constructor of the SQLConnection class to fix problems when the main app does not include a link to the SQL Classes assembly.

Visual Studio integration

Debugger

- The debugger expression evaluator now also evaluates late bound properties and fields (if that compiler option is enabled inside your project).
- If this causes negative side effects then you can disable that in the "Tools/Options Debugging/X# Debugger options screen".
- The debugger expression evaluator now is initialized with the compiler options from your main application (if that application is an X# project). The settings on the Debugger Options dialog are now only used when debugging DLLs that are loaded by a non-X# startup project.
- The debugger expression evaluator now always accepts a '.' character for instance fields, properties and methods, regardless of the setting in the project options.
- This is needed because several windows in the VS debugger automatically insert '.' characters when adding expressions to the watch window or when changing values for properties or fields.

New Features

- Added support for importing Indexes in the DbServer editor.
- The X# project system now remembers which Windows were opened in the Windows editor in design mode and reopens them correctly when a solution is reopened.
- We have added templates for a Harbour console application and Harbour class library.
- We have added item templates for FoxPro syntax classes and Xbase++ syntax classes.
- The Class templates for the FoxPro and XBase++ dialect now include a class definition in that dialect.

- We have improved the support for PPO files in the VS Editor.
- We have updated some of the project templates.

Bug fixes

- Fixed a problem with incorrectly showing member list in the editor for the ":=" operator (#1061)
- Fixed VOMED generation of menu item DEFINE names that were different to the ones generated by VO (#1208)
- Fixed VOWED incorrect order of generated lines of code in some cases (#1217)
- Switched back to our own version of Mono.Cecil to avoid issues on computers that have the Xamarin (MAUI) workload in Visual Studio.
- Fixed a problem opening a form in the Form Designer that contains fields that are initialized with an XBase function call (#1251).
- Windows that were in [Design] mode when a solution is closed, are now properly opened in [Design] mode when the solution is reopened.

Changes in 2.15.0.3

Compiler

New Features

- Implemented the STACKALLOC syntax for allocating a block of memory on the stack (instead of the heap) (#1084)
- Added ASYNC support to XBase++ methods (#1183)

Bug fixes

- Fixed missing compiler error in a few specific cases when using the dot for accessing instance members, when /allowdot is disabled (#1109)
- Fixed some issues with passing parameters by reference (#1166)
- Fixed some issues with interpolated strings (#1184)
- Fixed a problem with the macro compiler not detecting an error with incorrectly accessing static/instance members (#1186)
- Fixed incorrect line number reported for error messages on ELSEIF and UNTIL statements (#1187)
- Fixed problem with using an iVar named "Value" inside a property setter, when option /cs is enabled (#1189)
- Fixed incorrect file/line info reported in error message when the Start() function is missing (#1190)
- Fixed bogus warning about ambiguous methods in some cases (#1191)
- Fixed a preprocessor problem with nested square brackets (#1194)
- Fixed incorrect method overload resolution in some cases in the VO dialect (#1195)
- Fixed erratic debugging while stepping over code in some cases (#1200)
- Fixed a problem where a missing "end keyword", such as ENDIF, NEXT, ENDDO was not reported when the code between the start and end contained a compiler warning (#1203)
- Fixed a problem in the build system where sometimes an error message about an incorrect "RuntimeIdentifier" was shown

Runtime

Bug fixes

- Fixed runtime error in DBSort() (#1196)
- Fixed error in the ConvertFromCodePageToCodePage function

- A change in the startup code for the XSharp.RuntimeState could lead to incorrect codepages

Visual Studio integration

New Features

- Added VS option for the WED to manually adjust the x/y positions/sizes in the generated resource with multipliers (#1190)
- Added new options page to control where the editor looks for identifiers on the Complete Word (Ctrl+Space) command.
- A lot of improvements to the debugger expression evaluator (#1050). Please note that this debugger expression evaluator is only available in Visual Studio 2019 and later
- Added a debugger options page that controls how expression are parsed by the new debugger expression evaluator.
  You can also change the setting here that disallows editing while debugging.
- We have added context help to the Visual Studio source code editor. When you press F1 on a symbol then we inspect the symbol. If it comes from X# then the relevant page in the help file is opened. When it comes from Microsoft then we open the relevant page from the Microsoft Documentation online.
  In a next build we will probably add an option for 3rd parties to register their help collections too.
- When a keyword is selected in the editor that is part of a block, such as CASE, OTHERWISE, ELSE, ELSEIF then the editor will now highlight all keywords from that block.
- The Jump Keywords EXIT and LOOP are now also highlighted as part of the repeat block that they belong to.
- When a RETURN keyword is selected in the editor, then the matching "Entity" keyword, such as FUNCTION, METHOD will be highlighted too.
- Added a warning to the Application project options page, when switching the target framework.

Bug fixes

- Fixed previously broken automatic case synchronization, when using the cursor keys to move to a different line in the editor (#722)
- Fixed some issues with using Control+Space for code completion (#1044, #1140)
- Fixed an intellisense problem with typing ":" in some cases (#1061)
- Fixed parameter tooltips in a multiline expressions (method/function calls) (#1135)
- Fixed problem with Format Document and the PUBLIC modifier (#1137)
- Fixed a problem with Go to definition not working correctly with multiple partial classes defined in the same file (#1141)
- Fixed some issues with auto-indenting (#1142, #1143)
- Fixed a problem with not showing values for identifiers in the beginning of a new line when debugging (#1157)
- Fixed Intellisense problem with LOGICs in some cases (#1185)
- Fixed an issue where the completionlist could contain methods that were not visible from the spot here the completionlist was shown (#1188)
- Fixed an issue with the display of nested types in the editor (#1198)
- Cleaned up several X# project templates, fixing problems with incorrect placement of Debug/Output folders (#1201)
- Undoing a case synchronization in the VS editor was not working, because the editor would immediately synchronize the case again (#1205)
- Rebuilding the intellisense database no longer restarts Visual Studio (#1206)
- VOXporter now writes the menu ids from VO menus to the exported .xsmnu files and these are

reused inside X# (#1207)

- A Change to our project system and language service could lead to broken "Find in Files" functionality in some versions of Visual Studio. This has been fixed.
- Fixed an issue where goto definition was not working for protected or private members
- Fixed an issue where for certain files the Dropdown combo boxes on top of the editor were not correctly synchronized.

Documentation

Changes

- Some methods in the typed SDK were documented as Function. They are now properly documented as Method
- Property Lists and Method lists for classes now include references to methods that are inherited from parent classes. Methods that are inherited from .Net classes, such as ToString() from System.Object are NOT included.

Changes in 2.14.0.2, 3 & 4

Visual Studio Integration

Bug fixes

- Fixed an exception in the X# Editor when opening a PRG file in VS 2017
- Selecting a member from a completion list with the Enter key on a line immediately after an entry that has an XML comment could lead to extra triple slash (///) characters to be inserted in the editor
- The triple slash command to insert XML comments was not working. This has been fixed.
- Fixed a problem with entity separators not shown on the right line for entities with leading XML comments
- Fixed a peek definition problem with types in source code that do not have a constructor
- Fixed a problem with the Implement Interface action when the keyword case was not upper case
- Fixed a problem that the keyword case was prematurely synchronized in the current line.
- Fixed a problem with indenting after keywords such as IF, DO WHILE etc
- Fixed a problem with selecting words at the end of a line when debugging
- Fixed a problem where Format Document could lock up VS
- Fixed a problem that accessors such as GET and SET were not indented inside the property block
- Fixed a problem that Format Document was not working for some documents
- Changed the priority of the background scanner that is responsible for keyword colorization and derived tasks inside VS.

Changes in 2.14.0.1

Compiler

Bug fixes

- Fixed a problem with date literals resulting in a message about an unknown alias "gloal" (#1178)
- Fixed a problem that leading 0 characters in AssemblyFileVersion and AssemblyInformationalVersion were lost. If the attribute does not have the wildcard '*' then these leading zeros are preserved (#1179)

Runtime

Bug fixes

- The runtime DLLs for 2.14.0.0 were marked with the TargetFramework Attribute. This caused problems. The attribute is no longer set on the runtime DLLs (#1177)

Changes in 2.14.0.0

Compiler

Bug fixes

- Fixed a problem resolving methods when a type and a local have the same name (#922)
- Improved XML doc messages for methods implicitly generated by the compiler (INITs, implicit constructors) (#1128)
- Fixed an internal compiler error with DELEGATEs with default parameter values (#1129)
- Fixed a problem with incorrect calculation of the memory address offset when obtaining a pointer to a structure element (#1132)
- Fixed problematic behavior of #pragma warning directive unintentionally enabling/.disabling other warnings (#1133)
- Fixed a problem with marking the complete current executing line of code while debugging (#1136)
- Fixed incompatible to VO behavior with value initialization when declaring global MEMVAR (#1144)
- Fixed problem with compiler rule for DO not recognizing the "&" operator (#1147)
- Fixed inconsistent behavior of the ^ operator regarding narrowing conversion warnings (#1160)
- Fixed several issues with CLOSE and INDEX UDC commands (#1162, #1163)
- Fixed incorrect error line reported for error XS0161: not all code paths return a value (#1164)
- Fixed bogus filename reported in error message when the Start() function is missing (#1167)
- The PDB information for a command defined in a UDC now highlights the entire row and not just the first keyword
- Fixed a problem in the CLOSE ALL and CLOSE DATABASES UDC.

Runtime

New Features

- Added 2 new values to the DbNotificationType enum: BeforeRecordDeleted and BeforeRecordRecalled. Also added AfterRecordDeleted and AfterRecordRecalled which are aliases for the already existent RecordDeleted and RecordRecalled (#1174)

Bug fixes

- Added/updated several defines in the Win32API SDK library (#696)
- Fixed a problem with "SkipUnique" not working correctly (#1117)
- Fixed an RDD scope problem when the bottom scope is larger than the highest available key value (#1121)
- Fixed signature of LookupAccountSid() function in the Win32API SDK library (#1125)
- Improved exception error message when attempting to use functions like Trim() (which alter the key string length) in index expressions (#1148)
- Fixed a Macro Compiler runtime exception when there is an assignment in an IIF statement (#1149)
- Fixed a problem with resolving the correct overloaded method in late bound calls (#1158)
- Fixed a problem with parametrized SQLExec() statements in the FoxPro Dialect
- Fixed a problem in the Days() function where the incorrect number of seconds in a day was used.

- Fixed a problem in the Advantage RDD when a FieldGet returned fields with trailing 0 characters. These are now replaced with a space.
- Fixed a problem with DBI_LUPDATE in the ADS RDDs
- Fixed the Debugger display of the USUAL type.

Visual Studio integration

New Features

- Now using the "Reference Manager" instead of the "Add Reference Dialog Box" for adding References (#21, #1005)
- Added an option to the Solution Explorer context menu to split a Windows Form in a form.prg and form.designer.prg (#33)
- We have added an options page to the Tools / Options TextEditor/X# settings that allows you to enable/disable certain features in the X# source code editor, such as "Highlight Word", "Brace Matching" etc.
- Tooltips for all source code items now contain the Location (file name and the line/column).

Bug fixes

- Fixed a problem renaming files when a solution is under SCC with Team Foundation Server (#49)
- The WinForms designer now ignores differences in the namespaces specified in the form.prg and designer.prg files (the one from form.prg is used) (#464)
- Fixed incorrect mouse tooltip for a class in some cases (#871)
- Fixed a code completion issue on enum types with extension methods (#1027)
- Fixed some intellisense problems with enums (#1064)
- Fixed a problem with Nuget packages in VS 2022 causing first attempts to build projects to fail (#1114)
- Fixed a formatting problem in XML documentation tooltips (#1127)
- Fixed a problem with including bogus extra static members in the code completion list in the editor (#1130)
- Fixed problem with Extension methods not included in Goto Definition, Peek definition, QuickInfo tips and Parameter Tips (#1131)
- Fixed a problem in determining the correct parameter number for parameter tips when a compiler pseudo function such as IIF() was used inside the parameter list (#1134)
- Fixed a problem with selecting words with mouse double-click in the editor with underscores while debugging (#1138)
- Fixed a problem with evaluating values of identifiers with underscores in their names while debugging (#1139)
- Fixed identifier highlighting causing the VS Editor to hang in certain situations (#1145)
- Fixed indenting of generated event handler methods in the WinForms designer (#1152)
- Fixed a problem with the WinForms designer duplicating fields when adding new controls (#1154)
- Fixed a problem with the WinForms designer removing #region directives (#1155)
- Fixed a problem with the WinForms designer removing PROPERTY declarations (#1156)
- Fixed a problem that the type lookup for locals was failing in some cases (#1168)
- Fixed a problem where the existence of extension methods in code was causing a problem filling the member list (#1170)
- Fixed a problem when completing the member completion list without selecting an item (#1171)
- Fixed a problem with showing member completion on types of static members of a class (#1172)
- Fixed a problem with the indentation after single line entities, such as GLOBAL, DEFINE, EXPORT etc. (#1173)

- Optional tokens in UDCs were not colored as Keyword in the source code editor
- Fixed a problem in the CodeDom provider that failed to load on a Build Server because of a dependency to Microsoft.VisualStudio.Shell.Design version 15.0 when generating code for WPF projects.

Changes in 2.13.2.2

Compiler

Bug fixes

- Class members declared with only the INSTANCE modifier were generated as public. This has been changed to protected, just like in Visual Objects (#1115)

Runtime

Bug fixes

- IVarGetInfo() returned incorrect values for PROTECTED and INSTANCE members. This has been fixed.(#1116)
- The Default() function was changing usual variables initialized with NULL_OBJECT to the new value. This was not compatible with Visual Objects (#1119)

Visual Studio integration

New Features

- The Rebuild Intellisense Database menu option now asks for confirmation before restarting Visual Studio (#1120)
- The "Include Files" node in the solution explorer can now be hidden (Tools/ Options X# Custom Editors/Other Editors)

Bug fixes

- The type information for variables declared in a CATCH clause was not available. This has been fixed (#1118)
- Fixed several issues with parameter tips (#1098, #1065)
- Fixed a performance issue when the cursor was on a undeclared identifier in a "global" entity such as a function or procedure in VERY large projects
- The "Include Files" node could contain duplicate references when the source code for an #include statement contained relative paths, such as
  ```
  #include "..\GlobalDefines.vh"
  ```
- Suppressed the expansion of the Include Files node in the Solution Explorer when a solution is opened.
- Single character words (like i, j, k) were not highlighted with the 'highlight word' feature
- The type 'ptr' was not marked in the keyword color in quickinfo tooltips
- The nameof, typeof and sizeof keywords were not synchronized in the keyword case

Changes in 2.13.2.1

Compiler

New Features

- The parser now recognizes AS <type> clause for PUBLIC and PRIVATE memory variable declarations

but ignores these with a warning
- We have added support for AS <type> for locals declared with LPARAMETERS. The function/procedure is still clipper calling convention, but the local variable is of the declared type.

Bug fixes

- The PUBLIC and PRIVATE keywords are sometimes misinterpreted as memvar declarations when the /memvar compiler option is not even selected. We have added parser rules to prevent this from happening: when /memvar is not selected then PUBLIC and PRIVATE are only used as visibility modifiers
- Fix to an issue with selecting function and method overloads (#1096, #1101)
- Build 2.13.2.0 introduced a problem that could cause a big performance problem for VERY large source files. This has been fixed in 2.13.2.1.

Runtime

Bug fixes

- When the runtime cannot resolve a late bound call to an overloaded method it produces an error message that includes a list of all relevant overloads (#875, #1096).
- The .NULL. related behavior that was added for the FoxPro dialect was breaking existing code that involves usuals. In the FoxPro dialect DBNull.Value is now seen as .NULL. but in the other dialects as a NULL_OBJECT / NIL
- Several internal members of the PropertyContainer class in the VFP library are now public

Visual Studio integration

Bug fixes

- The lookup code for Peek definition, Goto definition etc. was filtering out instance methods and only returning static methods. This has been fixed (#1111, #1100)
- Several changes to fix issues with indentation while typing (#1094)
- Fixed several problems with parameter tips (#1098, #1066, #1110)
- A recent change to support the wizard that converts packages.config to package references has had a negative impact on nuget restore operations during builds inside Visual Studio. This was fixed. (#1113 and #1114)
- Fixed recognition of variables in lines such as CATCH, ELSEIF, FOR, FOREACH etc (#1118)
- Fixed recognition of types in the default namespace (#1122)

Changes in 2.13.1

Compiler

New Features

- The PUBLIC and PRIVATE statements in the FoxPro dialect now support inline assignments, such as in
```
PUBLIC MyVar := 42
```
Without initialization the value of the PUBLIC will be FALSE, with the exception of the variable with the name "FOXPRO" and "FOX". These will be initialized with TRUE in the FoxPro dialect

Bug fixes

- Fixed a problem with initialization of File Wide publics in the foxpro dialect
- Column numbers for error messages were not always correct for complex expressions. This has been fixed (#1088)

- Corrected an issue in the lexer where line numbers were incorrect when the source contains statements that span multiple lines (by using a semicolon as line continuation character) (#1105)
- Fixed a problem in the overload resolution when one or more overloads have a Nullable parameter(#1106), such as in

```
class Dummy_
   method Test (param as usual) as int
   .
   method Test(param as Int? as int
   .
end class
```

- Fixed a problem with the code generation for late bound method calls and/or array access in the FoxPro dialect with the /fox2 compiler compiler option ("compatible array handling") for variables of unknown type (#1108).
  An expression such as

  ```
  undefinedVariable.MemberName(1)
  ```

  was interpreted as an array access but it could also be a method call.
  The compiler now generates code that calls a runtime function that checks at runtime if "MemberName" is either a method or a property.
  If it is a property then the runtime will assume that it is an array and access the first element.
  Code with more than 2 parameters or with non-numeric parameters, such as

  ```
  undefinedVariable.DoSomething("somestring")
  ```

  was not affected, since "somestring" cannot be an array index.
  TIP: We recommend however, to always declare variables and specify their type. This helps to find problems at compile time and will generate MUCH faster code.

Runtime

New Features

- Added functions to resolve method calls or array access at runtime (#1108)
- Added GoTo record number functionality to the WorkareasWindow in the XSharp.RT.Debugger library

Visual Studio Integration

New Features

- Now the VS Project tree shows (in a special node) include files that are used by a project (#906).
  This includes include files inside the project itself but also include files in the XSharp folder or Vulcan folder (when applicable).
- We are using the built-in images of Visual Studio in the project tree and on several other locations when possible.
- Our background parser inside VS is now paused during the built process to interfere less with the build.
- We have added a setting to the indentation options so you can control the indentation for class fields and properties separately from methods.
  So you can choose to indent the fields and properties and to not indent the methods. This has also been added to the .editorconfig file

Bug Fixes

- Fixed problems with Peek Definition and Goto Definition
- When looking up Functions we were (accidentally) sometimes also including static methods in other classes.
- When parsing tokens for QuickInfo and Peek Definition then a method name would not be found if there was a space following the name and before the open parenthesis.
- Fixed a problem where project wide resources and settings (added from the project properties page) did not get the code behind file when saving
- Quick Info and Goto definition on a line that calls a constructor will now show / goto the first constructor of the type and no longer to the type declaration
- When the build process of a project was failing due to missing resources or other resource related problems, then the error list was not properly updated. This has been fixed (#1102)
- The XSharpDebugger.DLL was not properly installed in VS2017 and VS2019

Changes in 2.13.0.7

Compiler

New Features

- We have implemented a new compiler option /**allowoldstyleassignments**, which allows using the "=" operator instead of ":=" for assignments.
  This option is enabled by default in the VFP dialect and disabled by default in all other dialects.
- We have **revised the behavior** of the /vo4 and /vo11 command line options that are related to **numeric conversions**.
  Before /vo4 only was related to conversions between integral numbers. It has now been extended to also include conversions between fractional numbers (such as float, real8, decimal and currency) and integral numbers.
  In the original languages (VO, FoxPro) you can assign a fractional number to a variable with integral value without problems.
  In .Net you can't do that but you will have to add a cast to the assignment:

  ```
  LOCAL integerValue as INT
  LOCAL floatValue := 1.5 as FLOAT
  // no conversion: this will not compile in .Net without conversion
  integerValue := floatValue
  // explicit conversion: this does compile in .Net
  integerValue := (INT) floatValue
  ? integerValue
  ```

  If you enable the compiler option /vo4 then the assignment without the cast will also work.
  The /vo4 compiler option adds an implicit conversion. In both cases the compiler will produce a warning:
  warning XS9020: Narrowing conversion from 'float' to 'int' may lead to loss of data or overflow errors.
  The **value of the integer** integerValue above is controlled by the /vo11 compiler option:
  By default in .Net conversions from a fractional value to an integer value will round towards zero, so the value will then be 1.
  If you enable the compiler option /vo11 then the fractional number will be rounded to the nearest even integral value, so the value of integerValue in the example will be 2. This is not new.
  We have made a change in build 2.13, to make sure that this difference is no longer determined at

runtime for the X# numeric types but at compile time.
In earlier builds this was handled inside conversion operators from the FLOAT and CURRENCY types in the runtime.
These classes choose the rounding method based on the /vo11 setting from the main program which is stored in the RuntimeState object.
However that could lead to unwanted side effects when an assembly was compiled with /vo11 but the main program was not.
This could happen for example with ReportPro or bBrowser.
If the author of such a library now chooses to compile with /vo11 then he can be certain that all these conversions in his code will follow rounding to zero or rounding to the nearest even integer, depending on his choice.
- The DebuggerDisplay attribute for Compile Time Codeblocks has changed. You now see the source code for compile time codeblocks in the debugger.

Bug fixes

- Fixed a code generation issue with ASYNC/AWAIT (#1049)
- Fixed an Internal compiler error with Evaluate() in CODEBLOCK in VFP dialect (#1043)
- Fixed an Internal compiler error with UDCs incorrectly inserted after an END FUNCTION statement
- Fixed a problem in the preprocessor with #region and #endregion in nested include files (#1046)
- Fixed some problems with evaluating DEFINEs based on the order they appear (#866, #1057)
- Fixed a compiler error with nested BEGIN SEQUENCE .. END SEQUENCE statements (#1055)
- Fixed some problems with codeblocks containing complex expressions (#1056)
- Fixed problem assigning function to delegate, when /undeclared+ is enabled (#1051)
- Fixed a bogus warning when defining a LOCAL FUNCTION in the Fox dialect (#1017)
- Fixed a problem with the Linq Operation Sum on FLOAT values (#965)
- Fixed a problem with using SELF in an anonymous method/lamda expression (#1058)
- Fixed an InvalidCastException when casting a Usual to a Enum defined as DWord (#1069)
- Fixed incorrect emitted code when calling AScan() with param nStart supplied and similar functions (#1062, #1063)
- Fixed a problem with resolving the expected function overload (#1079)
- Fixed unexpected behavior of the preprocessor with #translate for specific XBase++ code (#1073)
- Fixed a problem with unexpected behavior of "ARRAY OF" (#885)
- Fixed some issues with calling specific overloads of functions accepting an ARRAY as a first argument (#1074)
- Fixed a bogus XS0460 error when using the PUBLIC keyword on a method (#1072)
- Fixed incorrect behavior when enabling Named Arguments option (#1071)
- Fixed Access violation when calling a function/method with DECIMAL argument with default value (#1075)
- Fixed some issues with #xtranslate not recognizing the Regular match marker in the preprocessor. Also fixed an issue with recognizing the double colon (::) inside expression tokens in the preprocessor. (#1077)
- Fixed some issues with declaring arrays in the VFP dialect (#848)
- Fixed a problem with column numbers in compiler error messages

Runtime

Bug fixes

- Fixed some incompatibilities with VO in the Mod() function

- Fixed an exception with Copy to array in the VFP dialect when dimensions do not match (#993)
- Fixed a seeking problem with SetDeleted(TRUE) and DESCEND order (#986)
- Fixed a problem with DataListView incorrectly showing (empty) deleted records with SetDeleted(TRUE) (#1009)
- Fixed problem with SetOrder() failing with SYMBOL argument (#1070)
- Reverted a previous incorrect change in the SDK in DBServer:FieldGetFormatted() (#1076)
- Fixed several issues with StrEvaluate(), including not recognizing MEMVARs with underscores in their names (#1078)
- Fix for a problem with InList() and string values (#1095)
- The Empty() function now returns false for the values .NULL. and DBNull.Value to be compatible with FoxPro
- Fixed a problem with GetDefault()/ SetDefault() to make them compatible with Visual Objects (#1099)

New Features

- Enhancements for Unicode AnyCpu SQL classes (#1006):
- Added a property to open a Sqlselect in readonly mode. This should prevent Append(), Delete() and FieldPut()
- Implemented delay creating InsertCmd, DeleteCmd, UpdateCmd until really needed
- Added callback mechanism so customers can override the commandtext for these command (and for example route them to stored Procedures)
- When a late bound method call cannot be resolved because the method is overloaded then a better error message is now generated that also includes the prototypes of the methods found (#1096)

FoxPro dialect

- Added ADatabases() function

Visual Studio Integration

New features

- You can now control how indenting is done through the Tools/Options Text Editor/X# option pages. We have added several options that control indenting of your source code. You can also set these from an .editorconfig file if you want to enforce indenting rules inside your company.
- We have implemented the option "Identifier Case Synchronization". This works as follows: The editor picks up the first occurrence of an Identifier (class name, variable name etc) in a source file and make sure that all other occurrences of that identifier in the same source file use the same case. This does NOT enforce casing across source files (that would be way too slow)
- We have added color settings to the VS Color dialog for Matched braces, Matched keyword and Matched identifiers. Open the Tools/Options dialog, Choose Environment/Fonts and Colors and look for the colors in the listbox that start with the word "X#". You can customize these to your liking.
- X# projects that use the Vulcan Runtime now have a context menu item that allows you to convert them to the X# runtime. Standard Vulcan assemblies will be replaced with the equivalent X# runtime assemblies. If you are using 3rd party components such as bBrowser or ReportPro then you need to replace the references to these components yourself.(#32)
- We have added an option to the language page of the project properties to set the new /allowoldstyleassignments commandline option for the compiler

Bug fixes

- Fixed a problem with Get Latest Version for solution that is under TFS (#1045)

- Fixed WinForm designer changing formatting in main-prg file (#806)
- Fixed some problems with code generation in the WinForms designer (#1042, #1052)
- Fixed a problem with formatting of DO WHILE (#923)
- Fixed problem with Light Bulb "Generate default constructor" feature (#1034)
- Fixed problem with ToolTips in the Debugger. We now parse the complete expression from the first token until the cursor location. (#1015)
- Fixed some remaining intellisense issues with .Net array locals defined with VAR (#569)
- Fixed a problem with indenting not working correctly in some cases (#421)
- Fixed a problem with auto outdenting (#919)
- Several improvements to keyword pair matching (#904)
- Fixed a problem with Code Completion showing also static members after typing a dot in "ClassName{}." #1081
- Fixed a performance Issue when typing . for .and. (#1080)
- Fixed a problem with the navigation bar while typing new classes/methods (#1041)
- Fixed incorrect info tooltips on keywords (#979)
- Right Click on a packages.config file and choosing the option "Migrate to packagereferences" did not work because inside Visual Studio there is a hardcoded list of supported project types. We are now "faking" the projecttype to make VS happy and enable the wizard.

Build System

Bug fixes

- The XSharp.Build.Dll, which is responsible for creating the command line when compiling X# projects in VS, was not properly passing the /noconfig and /shared compiler options to the compiler. As a result the shared compiler was not used, even when the project property to use the Shared Compiler was enabled. Also the compiler was automatically including references to all the assemblies that are listed inside the file xsc.rsp, which is located inside the XSharp\bin folder.
  You may experience now that assemblies will not compile because of missing types. This will happen if you are using a type that is inside an assembly that is listed inside xsc.rsp. You should add explicit references to these assemblies in your X# project now.
-

The What's new for older builds can be found in the X# documentation