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 2.8. For a complete list of changes that were made in earlier builds, please have a look at the help file.*

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

- 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.

Changes in 2.12.2.0

Compiler

Bug fixes

- Fixed a bug in the code generation for handling FoxPro array access with parenthesized indices (#988, #991)
- The compiler was generating incorrect warnings for locals declared with IS. This has been fixed.
- The compiler was not reporting an error on invalid usage of the OVERRIDE modifier on ACCESS/ASSIGNs, this has been fixed (#981)
- Fixed inconsistent behavior in reporting warnings and errors in several cases when converting from various numeric data types to another (#951, #987)
- Fixed some "failed to emit module" issues with iif() statements in some cases (#989)
- Fixed a problem with compiling X# code scripts (#1002)
- Fixed a problem with using classes for some specific assemblies in the macro compiler (#1003)

- Fix incorrect error message when adding an INT to a pointer in AnyCPU mode (#1007)
- Fixed a problem with casting STRING to PTR syntax (#1013)
- Fixed a problem with PCount() when passing a single NULL argument to a CLIPPER function/method (#1016)

New Features

- We have added support for the TEXT .. ENDTEXT command in all dialects. Please note that there are several variations of this command. One variation work in ALL dialects (TEXT TO varName). Other variations depend on the dialect chosen. We have moved the support for TEXT .. ENDTEXT now also from the compiler to the preprocessor. This means that there are also 2 new preprocessor directives, #text and #endtext (#977)
- Implemented new compiler option /vo17, which implements a more compatible to VO behavior for the BEGIN SEQUENCE..RECOVER command (#882, #916):
  - For code that contains a RECOVER USING, a check is made for wrapped exceptions. When the exception is not a wrapped exception then a function in the runtime is called (FUNCTION _SequenceError(e AS Exception) AS USUAL) that can process the error. It can for example call the error handler, or throw the error
  - When there is no RECOVER USING clause , then the compiler generates one and from within this generated clause detects if the RECOVER was reached with a wrapped exception or a normal exception. For wrapped exceptions it gets the value and calls a special function in the runtime (FUNCTION _SequenceRecover(uBreakValue AS USUAL) AS VOID). When the generated recover is called with a 'normal' exception then _SequenceError function from the previous bullet is called
- We have added support for CCALL() and CCALLNATIVE()
- The #pragma directives are now handled by the preprocessor. As a result you can add #pragma lines anywhere in your code: between entities, inside the body of an entity etc.

Runtime

Bug fixes

- Changed the prototype for AdsGetFTSIndexInfo (#966)
- Fixed a problem with TransForm and decimal types (#1001)
- Added several missing return types in the VFP assembly
- Fixed a problem with browsing a DBFVFP table in the FoxPro dialect
- Fixed an inconsistency with handling values provided by BREAK commands inside surrounding BEGIN...RECOVER statements, depending on if early or late bound call is used (#883)
- Fixed a problem with floating point format when assigning a System.Decimal value to a USUAL var (#1001)
- Fixed a runtime error with DbCopyToArray() when copying to an array that has more columns than the table, in the FoxPro dialect (#993)
- Fixed a problem with the typecast expression and numeric literals with the +/- sign in the macro compiler (#1025)
- Fixed problem in the Late binding code where a string was sometimes passed in and not properly converted to symbol
- IVarPut()/IVarGet() now throw an appropriate exception when trying to use an inaccessible (due to limiting visibility modifiers) property getter/setter (ACCESS/ASSIGN) (#1023)
- Fixed an issue with IVarGet() and IVarPut() for properties that are redefined in a subclass with the NEW modifier (#1030)
- DbDataSource now tries to lock a record when deleting or recalling the record
- Foreach was not working correctly on properties containing collections that were returned from a

late bound property access such as IVarGet()(#1033)

New Features

- You can now register a delegate in the runtime state that allows you to control how the macro compiler caches types from loaded assemblies(#998).
- This delegate has to have the format:

```
DELEGATE MacroCompilerIncludeAssemblyInCache(ass as Assembly) AS LOGIC
```

Example:

```
XSharp.RuntimeState.MacroCompilerIncludeAssemblyInCache := { a  =>
DoNotCacheDevExpress(a)}
FUNCTION DoNotCacheDevExpress(ass as Assembly) AS LOGIC)
   // do not cache DevExpress assemblies
   RETURN ass:Location:IndexOf("devexpress",
StringComparison.OrdinalIgnoreCase) == -1
```

Compatible VO SDK

- Fixed an issue where SetAnsi(FALSE) causes SingleLineEdit controls with pictures to show random characters when entering umlauts (#1038)

Typed VO SDK

There are 2 new properties for the SQLSelect class.

- ReadOnly - which makes the SQLSelect Readonly
- BatchUpdates - which controls how updates are handled

ReadOnly cursors and delayed creation of command objects

Previously the SQLSelect class created DbCommand objects to update, insert and delete changes made to a cursor immediately when the result set was opened.
That could cause problems when a complex query was used to select data, because the DbCommandBuilder object could not figure out how to create these statements.
We are now delaying the creation of these commands until the first time they are needed.
At the same time we have now added a ReadOnly property with a default value of FALSE.
If you set ReadOnly to true then:

- Calling FieldPut(), Delete() and Append() will generate an error with Gencode EG_READONLY.
- No Command objects will be created for the SQLSelect, because the cursor cannot be updated.

If ReadOnly remains FALSE then the command objects to update, insert and delete will be created the first time they are needed.
These commands are created in the __CreateDataAdapter() method.
You can override this method and create the commands in your own subclass when you want.
The command creation and the updates work as follows:

- First a DataAdapter (of type DbDataAdapter) is created using the CreateDataAdapter method from the SQLFactory class
- Then a CommandBuilder object (of type DbCommandBuilder) is created from the CreateCommandBuilder method of the SQLFactory class
- Then the Insert, Delete and Update Command objects (all of type DbCommand) are created from the GetInsertCommand() etc methods from the DbCommandBuilder object. The DBCommandBuilder object takes the Select statement and creates commands with parameters based on the SQLSelect command
- These command objects are assigned to the DataAdapter and then the DataAdapter:Update()

method is called with the DataTable that is behind the SQLSelect as argument.

Batch Updates

Normally updates in a SQLSelect will be sent to the server when you move the record pointer to a new row, or when you call Update()
If you set the BatchUpdates property to TRUE then the SQLSelect will delay sending updates to the server and will not do that for each record movement, but will wait until you call the Update() method with an argument TRUE. This will then write all the buffered changes to the server. This may then also trigger the creation of the DBCommand objects (see before).
If your table has autoincrement fields then you may want to call Requery() afterwards to see the newly assigned key values.

Visual Studio Integration

Bug fixes

- Fixed the handling for project property pages for flavoured projects (#992)
- When trying to start the debugger with a non existing working directory or program file name an error is displaid (#996)
- Fixed a problem with the form designer generating sometimes invalid code with #regions (1020)
- The WinForms designer now by default adds the OVERRIDE keyword modifier to the generated Dispose() method (was added in the template) (#1004)
- Due to a changed threading model inside the latest VS2022 releases, error messages were sometimes not shown in the output window and the error list. This has been fixed.
- Fixed a problem in the windows forms designer code generation with nested classes inside the main form class (#1031)
- Fixed problem with windows forms editor failing to open form with command based on UDC (#1037)

Sourcecode Editor

- Type lookups on full names were sometimes failing because the fullname was defined as case sensitive (#978)
- Nested type lookup was sometimes failing. This has been fixed.
- The indenting options can now also be overridden in the .editorconfig file (#999)
- When a source file was loaded in the editor then the combo boxes with types and members were not activated until the caret was moved in the buffer (#995)
- The member combobox in the editor was getting confused for code that contains local functions or local procedures.
- Fixed a lookup problem for expressions inside a conversion or cast with a keyword, such as DWORD(SomeExpression). There were no quick info tips for the expression inside the parentheses for the conversion (#997)
- Fixed an intellisense problem with DATATYPE(<expression>) conversion expressions (#997)
- Fixed a problem with properties declared with the => symbol in their implementation causing Navigation Bar contents to be incomplete (#1008)
- Fixed several issues with code folding and formatting (#975)
- Fixed problem with typing a comma inside an argument list did not invoke the Parameters Tooltip (#1019)
- Fixed some issues with the detection of variable types for the VAR keyword (#903)
- Fixed an Intellisense problem with typing ":" or "." inside a string literal (#1021)
- Fixed a problem with unknown identifiers sometimes causing bogus member completion list to show (#1022)
- Pressing CTRL+SPACE in the editor now always invokes a code completion list (#957)

New Features

- Added options to insert page and reorder pages in a tabcontrol, in the VOWED (#1024)
- We have updated the WPF Application template. The Main window is now called "MainWindow".
- Added the following new settings to the .editorconfig file to set indentation options (#999).

```
indent_entity_content (true or false)
indent_block_content (true or false)
indent_case_content (true or false)
indent_case_label (true or false)
indent_continued_lines (true or false)
```

VOXporter

- The VOXporter now correctly enabled or disables the Allow MEMVAR/Undeclared vars compiler options, if they were enabled in the VO app (#1000)

Changes in 2.11.0.1

Compiler

Bug fixes

- Fixed an internal compiler error with CLIPPER calling convention delegates (#932)
- Fixed an AccessViolationException at runtime with the Null-conditional operator ?. on a usual property (#770)
- [XBase++ dialect] Fixed a problem with parsing method declarations with parentheses (#927)
- [XBase++ dialect] Fixed a problem with parsing the (obsolete in X#) ANNOUNCE and REQUEST statements (#929)
- [XBase++ dialect] Fixed a problem with parsing INLINE ACCESS and ACCESS ASSIGN statements (#926)
- [VFP dialect] Fixed a problem with parsing FOR EACH statements containing "M." variables usage where the variable was not typed in the FOR EACH line (#911) .
- Fixed a problem where the PPO files contains some output twice, when a single UDC was producing several statements (#933)
- Fixed some issues with the "FIELDS" clause in several UDCs (#931, #795)
- Fixed a problem in the preprocessor with parentheses in #xtranslate directives (#963)
- Fixed several more issues with #command and #translate directives (#915)
- In some cases, the compiler would emit code that does not throw a runtime exception, when casting/converting from one type to an incompatible one. This has been fixed (#961, #984)
- The compiler was not reporting narrowing conversion warnings in several cases, this has been fixed (#951)
- The compiler was not reporting signed/unsigned conversion warnings. This has been fixed (#971)
- Fixed a problem that could lead to the "Could not emit module" error message, caused by NULL values inside IIF() expressions(#989)

New features

- Added compiler option /noinit to not generate $Init calls for libraries without INIT procedures for the sake of postponed loading (#854)
- Added preprocessor support for #stdout and #if. (#912)
- The full contents of #include files is now written to the ppo file (#920)
- When a parser error occurs because an identifier was replaced by a define with the same name, then the compiler will now generate a second warning.

- If a header file contains actual code and this code is called during debugging then the debugger will now step into the header file when debugging this code.
  Previously all statements were linked to the #include line from the place where the header was included. (#967)
- When you are suppressing compiler errors with the /vo11 (Compatible numeric conversion) compiler option you will now see a XS9020 "narrowing" warning indicating that a runtime error may happen or that data may be lost.

Visual Studio Integration

New features

- The source code editor now also supports the new #if and #stdout preprocessor commands (#912)
- There is new "Lightbulb" option to generate constructors for classes.

Bug Fixes

- Fixed a problem with specifying custom preprocessor defines in the project properties (#909)
- The VO-style editors now retain existing "CLIPPER" clause to methods/constructors when generating code (#913)
- Fixed incorrect parsing of classes as nested to each other (#939)
- Fixed a problem with using embedded variables in the form of $(SomeName) in the project settings (#928)
- Fixed a problem where deleting items from a project would fail.
- Fixed a problem resolving the DLL produced by project files from other development languages, in particular SDK style C# projects (#950)
- Fixed a problem with quick info tooltip after an unrecognized identifier (#894)
- Fixed a problem with the editor incorrectly adding parentheses after auto typing a property (#974)
- Fixed extremely slow editor response when creating a new line after an #endif directive (#970)
- Fixed some intellisense issues with .Net array types (#569)
- Fixed a problem with the DevExpress DocumentManager control at design time (#976)
- Fixed an ArgumentNullException in the Output window when "Show output from" is set to "Extension" (#940)

Runtime

New features

- Added a constructor with IEnumerable to the array class (#943)
- Implemented missing functions AdsSetTableTransactionFree() and AdsGetFTSIndexInfo() (#966)
- Moved functions GetRValue(), GetGValue() and GetBValue() from the Win32API library to XSharp.RT, so they can be used by AnyCPU code (#972)
- [VFP dialect] Implemented function APrinters() (#952)
- [VFP dialect] Implemented function GetColor() (#973)
- [VFP dialect] Implemented functions Payment(), FV() and PV() (#964)
- [VFP dialect] Implemented commands MKDIR, RMDIR and CHDIR (#614)

Bug fixes

- Fixed a problem with the ListView TextColor and TextBackgroundColor ACCESSes in the SDK (#896)
- Fixed a problem with soft Seek not respecting order scope when to strict key is found (#905)
- Fixed DBUseArea() search logic for files in various folders. Also SetDefault() is no longer initialized with the current directory (for VO compatibility) (#908)

- Fixed problem with creating dbfs with character fields with length > 255 (#917)
- Fixed a problem with the buffered read system in some cases when a dbf was being read, closed, overwritten and then reopened (#968)
- Fixed a VO compatibility problem with how DBSetIndex() changes the active order when opening index files (#958)
- Fixed a problem with db append, copy etc, when both source and destination files have the same structure and include a memo file (#945)
- Fixed an incorrect result of DBOrderInfo(DBOI_ORDERCOUNT) with a non existing or not open index file (#954)
- [VFP dialect] Added optional parameter to Program([,lShowSignature default=.f.]) (#712)
- [VFP dialect] Fixed several issues with the Type() function (#747, #942)
- [VFP dialect] Fixed a problem with ExecScriptFast() (#823)
- [VFP dialect] Fixed a problem with SQLExec() not putting the record pointer on the first record (#864)
- [VFP dialect] Fixed a problem with SQLExec() with null values (#941)
- [VFP dialect] Fixed a write error in the buffer returned from SqlExec() (#948)
- [VFP dialect] Fixed a problem with the DBFVFP RDD and null columns (#953)
- [VFP dialect] Fixed a problem with SCATTER TO and APPEND FROM ARRAY (#821)

Typed SDK

- Fixed a problem with the FileName property of standard open dialogs
- Fixed a problem with a FOREACH inside the Menu constructor causing handled exceptions

RDD

Bug fixes

- Fixed a problem in the DBFVFP RDD with the calculation of the keysize of nullable keys (#985)

VOXporter

Bug fixes

- Fixed incorrectly detecting pointers to functions inside literal strings and comments (#932)

Changes in 2.10.0.3

Compiler

Bug fixes

- Fixed some problems with COPY TO ARRAY command in the FoxPro dialect (#673)
- Fixed a problem with using a System.Decimal type on a SWITCH statement (#725)
- Fixed an internal compiler error with Type() in the FoxPro dialect (#840)
- Fixed a problem with generating XML documentation (#783, #855)
- Prevented a warning from appearing for members of SEALED classes when /vo3 (all members VIRTUAL) is enabled (#785)
- Fixed problems with assigning and comparing "ARRAY OF <type>" vars to NULL_ARRAY (#833)
- Fixed some issues with passing arguments by reference with the @ operator and/or using it as the AddressOf operator (#810, #899, #902)
- Fixed a problem resolving parameters passed by reference with the @ operator when the function/

method had a parameter of the pointer type (#899, #902)

New features

- Added compiler option (/enforceoverride) to make the OVERRIDE modified mandatory when overriding a parent member (#786, #846)
- The compiler now reports an error when using String2Psz() and Cast2Psz() in a non local context (since such PSZs are being released on exiting the current entity) (#775)
- FUNCTIONs and PROCEDUREs now support the ASYNC modifier (#853)
- You can now suppress the automatic generation of the $Init1() and $Exit() functions by passing the compiler commandline option /noinit (#854). This is NOT yet supported in the VS Properties dialog
- Added support for the ASTYPE operator also for USUAL vars (#857)
- Allowed specifying AS <type> clause in PUBLIC var declarations (ignored by the compiler, but used by the editor in the future for intellisense) (#853)
- The AS <datatype> OF <classlib> clause is now also supported for several other FoxPro compatible commands, such as PARAMETERS and PUBLIC.
- Since these variables are untyped at runtime by nature, the clause is ignored by the compiler and a warning is shown.

Build System

Bug Fixes

Running MsBuild on a X# WPF project could fail (#879)

Visual Studio Integration

New features

- We have added Visual Studio integration for VS 2022
- We have added support for Package References
- Now XML comments are automatically inserted in the editor when the user types "///". (#867, #887)
  Conditions:
  - Cursor must be on a line before the start of an entity
  - Cursor must NOT be before a comment line
- Now the tooltip on a class includes also information about the parent class and implemented interfaces (if any) (#860)
- We have added tooltips, parameter completion etc for the pseudo functions that are built into the compiler, such as PCount() and String2Psz().
- We have added a first version of Lightbulb tips. For now to implement missing interface members and to convert a field to a Property. More implementations and configuration options will follow
- We have added a new dialog to configure source code formatting with visual examples of the effects of the options.
- We have added the ability to log operations of the X# VS integration to the Windows debug window and/or a logfile.
  If you are experiencing unexplainable problems we will contact you and tell you how to enable these options, so you can send us a log file that shows what happened before a problem occurred inside Visual Studio. We have used Serilog for this.
- The Highlight Word feature now is case insensitive and no longer hightlights words that are part of a comment, string or inactive editor region
- We have added 'Brace Completion' to the editor

Bug Fixes

- Fixed some problems with the Format Document command (#552)
- Fixed several issues with Parameter Tooltips (#728, #843)
- Fixed problem with code completion list showing even for not defined vars/identifiers (#793)
- Fixed member completion and parameter tooltips with chained expressions (#838)
- Fixed recognition of type for VAR locals in some cases (#844)
- Fixed member completion and tooltip info problems with VOSTRUCT vars (#851)
- Fixed a problem with ignoring Line Breaks in XML Comments (#858)
- Fixed some WinForms designer problems with CHAR properties (#859)
- Fixed a problem with Goto Definition not working correctly with SUPER() constructor calls (#862)
- Fixed an error with the Rebuild Intellisense Database command, when the solution contains a space in the path (#865)
- Goto Definition for types from external assemblies was failing when there was more than one copy of VS running at the same time.
- Fixed a problem with a VOSTRUCT some times confusing the parser (#868)
- Fixed some more problems with quickinfo and member completion (#870)
- Fixed a problem in the Windows Forms designer (#873)
- Fixed an intellisense problem with ENUMs using no MEMBER keywords (#877)
- Fixed a member completion problem with inherited exception types (#884)
- If an XML topic had sub elements of type <see> or other these were not shown in the editor. This has been fixed (#900)
- Unbalanced braces were sometimes matched in the editor with keywords. This has been fixed (#892)
- Line separators were sometimes flickering. This has been fixed (#792)
- When parsing for local variables we were not processing the include files. This could lead to a situation where a local that was declared in a conditional block (#ifdef SOMEVAR) was not found. This has been fixed. The editor parser now includes the header files and #defines and #undefines found in the code even when parsing a part of the source file (#893)
- #include lines are now included in the fields/members combobox in the editor (when fields are shown). They are also saved to the intellisense database.
- The editor was trying to show QuickInfo tooltips when the cursor was over an inactive preprocessor region (#ifdef). This no longer happens.

Runtime

Bug fixes

- Fixed DBFCDX corruption that could happen with simultaneous updates (#585)
- Fixed a problem opening FoxPro tables with indexes on nullable fields (#631)
- The BlobGet() function was returning a LOGIC instead of the actual field value (#681)
- Greatly improved speed of index creation with large number of fields in the index expression (#711)
- Fixed some problems with FieldPutBytes() and FieldGetBytes() (#797)
- DBSeek() with 3rd param (lLast) TRUE had incorrect behavior in some cases (#807)
- Fixed a potential NullreferenceException that could happen when creating indexes (#849)
- Improved indentation in the text produced by the method Error.WrapRawException() (#856)
- Fixed a runtime problem when converting .Net Array <-> USUAL (#876)
- DbInfo() was returning TRUE even when an info enum was not supported.(#886)
- Fixed also a possible DBFNTX corruption problem (#889)
- DbEval() could fail in FoxPro when the codeblock was returning NIL or was VOID (#890)
- Fixed a problem with Softseek and descending indexes.

- Fixed a problem where incorrect scope expressions could lead to unexpected results. Now the server goes to (and stays at) EOF with an incorrect scope.
- Fixed a problem with accessing FoxPro arrays with the parenthesis operators in a macro expression (#805).
  Please note that for this to work you have to compile the main program with /fox2

Changes in 2.9.1.1 (Cahors)

Compiler

Bug fixes

- Fixed a problem introduced in 2.9.0.2 with define symbols not respecting the /cs compiler option in combination with the /vo8 compiler option (#816)
- Fixed an internal compiler error with assignment expressions inside object initializers when the /fox2 compiler option is enabled (#817)
- Fixed some problems with DATEs in VOSTRUCTs (#773)
- Fixed a problem in the preprocessor that would occur when using a list rule like `FIELDS <f1> [,<fn> ]` in the middle of a UDC.
- Fixed a problem compiling UDCs such as `SET CENTURY &cOn` because `cOn` was not parsed as an identifier but as a keyword.

New features

- There is a new result marker (the NotEmpty result marker) in the preprocessor that does the same as the regular result marker, but writes a NIL value to the output when the (optional) match marker is not found in the input.
  This can be used when you want to make the result a part of an IIF() expression in the output, since the sections inside an IIF expression may not be empty.
- The result marker looks like this: `<!marker!>`
- Using a Restricted match marker as the first token in an UDC was not allowed before. This has been fixed. You can now write a rule like this, which will output the keyword (SCATTER, GATHER or COPY) followed by the stringified list of options.

```
#command <cmd:SCATTER,GATHER,COPY> <*clauses*> => ?  <"cmd">, <"clauses">
FUNCTION Start AS VOID
    SCATTER TO TEST   // is preprocessed into ? "SCATTER" , "TO TEST"
    RETURN
```

Visual Studio Integration

Bug Fixes

- Fixed a problem introduced in 2.9.0.2 with code generation for WPF projects (#820)
- Fixed a VS freezing problem after building (#819)
- Fixed some problems with code collapsing and the navigation bar for source files that contains a SELF property (#825)
- Fixed a problem with the form designer emitting invalid code when the form prg contains nested classes (#828)
- Fixed a problem with code completion showing the wrong members when opened just left to a closing paren (#826)
- Fixed a VS crash when clicking on a generic class (#827)
- Fixed a problem with the keyword colorization for expressions such as `SET CENTURY &cOn`, where

`&cOn` was colored in the keyword color.
- Parameter tips for nested function calls required an extra space before the name of the nested function (#728)
- Fixed a problem with the form designer deleting delegates and other nested types in the form.prg (#828)
- The background process to load the types in the ClassView / ObjectView windows was slowing down the VS performance. This has been disabled for now.
- Fixed type lookup for Generic types.
- Hovering the mouse over a constructor keyword was showing a tooltip for the class and not for the constructor. This has been fixed.
- Fixed an issue in the code generator for Windows Forms for literal characters with special values (such as '\0') (#859)
- Fixed an exception in the project system when the project system was initialized in the background (for example when no X# projects were opened) (#852)
- Fixed missing code completion for the LONGINT and SHORTINT keywords (#850)
- The context menu option "View in Disassembler" is now only shown for X# projects
- Fixed code generator problem with ARRAY OF <type> (#842)
- Fixed a performance problem when clicking on code in the editor (#829)
- Fixed a problem with loading Windows Forms when the lookup of a nested type failed.

New features
- We have added a context item to the project context menu in the solution explorer to edit the project file. This will unload the project when needed and then open the file for editing.
- The Rebuild Intellisense Database menu option in the Tools/XSharp menu now unloads the current solution, deletes the intellisense database and reopens the solution to make sure that the database is recreated correctly.
- We have made some changes to the process that parses the source code for a solution in the background.
- Generic Typenames are now stored in the Name`n format in the Intellisense database, for example IList`1 for IList<T>

Runtime

New features
- Added missing ErrorExec() function (#830)
- Added support for BlobDirectExport, BlobDirectImport, BlobDirectPut and BlobDirectGet (#832)
- When you do a numeric operation on two USUALs of different types we now make sure that decimal values are no longer lost (#808). For example a LONG + DECIMAL will result in a DECIMAL. See the table in the USUAL type page in this help file for the possible return values when mixed types are used.

Bug Fixes
- Fixed a problem with _PrivateCount() throwing an InvalidateOperationException (#801)
- Fixed a problem with member completion in the editor sometimes showing methods of the wrong type (#740)
- Fixed some problems with the ACopy() function (#815)
- Fixed a few issues that were remaining related to DATEs in VOSTRUCTs (#773)

Macro compiler

New features

- Added support for the & operator (#835).
- Added support for parameters by reference (both @ and REF are supported) for late bound method calls (#818)

VOXporter

Bug Fixes

- Fixed problem with incorrectly prefixing PUBLIC declarations with "@@"

Changes *in 2.9.0.2 (Cahors)*

Compiler

New features

- The parser now supports class variable declarations and global declarations with multiple types(#709)
```
EXPORT var1 AS STRING, var2, var3 as LONG
GLOBAL globalvar1 AS STRING, globalvar2, globalvar3 as LONG
```
- If you are using our parser you should be aware that the ClassVarList rule has disappeared and that the ClassVars, VoGlobal and ClassVar rules have changed.
- We have added a command to fill a foxpro array with a single value
```
STORE <value> TO ARRAY <arrayName>
```
- When you create a VOSTRUCT or UNION that contains a DATE field, then the compiler will now use the new __WinDate structure that is binary compatible with how DATE values are stored inside a VOSTRUCT or UNION in Visual Objects (#773)
- It is now possible to use parentheses for (instead of brackets) accessing ARRAY elements in the FoxPro dialect. The compiler option /fox2 must be enabled for that to work (#746).
- We have added support (for the FoxPro dialect only) for accessing WITH block expressions inside code of a calling function / method. So you can type .SomeProperty and access the property that belongs to a WITH BLOCK expression inside the calling code. To use this Late Binding must be enabled, since the compiler does not know the type of the expression from the calling code (#811).

Bug fixes

- When you use the NEW or OVERRIDE modifier for a method where no (virtual) method in a parent class exists an error will now be generated (#586, #777)
- Fixed a problem with LOGICAL AND and OR for USUAL variables in an array (#597)
- Error messages and Warnings for some compiler generated code (such as Late bound code) were not always pointing to the right line number, but to the first line in the body of the method or function. This has been fixed. (#603)
- Fixed a problem incorrect return values for IIF expressions (#606)
- Fixed a problem in the compiler when parsing multiple method names on a DECLARE METHOD line (#708)
- Fixed a problem in the FoxPro dialect with assigning a single value to an array to fill the array (#720)
- Fixed a problem with the calculation of VOSTRUCT sizes when the structure contained a member of type DATE (734)
- The previous problem caused runtime errors (#735)
- Fixed a problem in code like this (#736)
```
var aLen := ALen(Aarray)
```

- Fixed a compiler crash when overriding CLIPPER method with STRICT for methods with typed return value (#761)
- When the interface implementation had different casing then the definition then an incorrect error message was shown (#765)
- Fixed a compiler crash with incorrect function parameters inside a codeblock (#759)
- Recursive definitions of DEFINEs could result in an infinate loop inside the compiler causing a StackOverflowException(#755)
- Fixed a problem with late bound calls and OUT parameters (#771)
- If you compile with warning level 4 or lower then certain warnings for comparing value types to null are not shown. We have changed the default warning level to 5 now. (#772)
- Fixed a compiler crash with multiple PRIVATE &cVarName statements in the same entity (#780)
- Fixed a problem with possibly corrupting the USUAL NIL value when passing USUAL params by reference (#784)
- Fixed a problem with declared PUBLIC variables getting created as PRIVATE in the FoxPro dialect (#753)
- Fixed a problem with using typed defines as default arguments (#718)
- Fixed a problem with typed DEFINEs that could produce constants of the wrong type (#705)
- Fixed a problem with removing whitespace from #warning and #error directive texts (#798)

Runtime

New Features

- We have added several strongly typed overloads for the Empty() function that should result in a bit better performance (#669)
- We have added an event handler to the RuntimeState class. This event handler is called "StateChanged" and expected a method with the following signature:
```
Method MyStateChangedEventHandler(e AS StateChangedEventArgs) AS VOID
```
- The StateChangeEventArgs type has properties for the Setting Enum, the OldValue and the NewValue.
- You can use this if you have to synchronize the state between the X# runtime and an external app, for example a Vulcan App, VO App or for example (this is where we are using it) with an external database server, such as Advantage.
- We have added a new (internal) type __WinDate that is used when you store a DATE value into a VoStruct or Union. This field is binary compatible with the Julian date that VO stores inside structures and unions.
- We have added an entry to the RuntimeState in which the compiler stores the current /fox2 compiler setting for the main app.
- Added runtime support to support filling FoxPro arrays by assigning a single value.

Bug Fixes

- Fixed a problem (incompatibility with VO) in the Descend() function (#779) - IMPORTANT NOTE: If you are using Descend() in dbf index expressions, then those indexes need to be reindexed!
- Late bound code that was returning a PSZ value was not correctly storing that inside a USUAL (#603)
- Fixed a problem in the Cached IO that could cause problems with low level file IO (#724)
- The VODbAlias() function now returns String.Empty and not NULL when called on an area where no table is open. (#733)
- Fixed a compatibility problem with the MExec() function (#737)
- The M-> prefix was not recognized correctly inside codeblocks (#738)
- The Explicit DATE -> DWORD cast was returning an incorrect value for NULL_DATE.

- Fixed a problem with late bound calls and OUT parameters (#771)
- Added a new __WinDate type that is used to store DATE values inside a VOSTRUCT or UNION. (#773)
- Fixed several problems with FoxPro arrays
- Removed TypeConstraints on T for functions that manipulate __ArrayBase<T>
- Fixed a problem with Directory() including files that match by shortname but not by longname (#800)

RDDs

- When creating a new DBF with the DBFCDX driver an existing CDX file is not automatically deleted anymore (#603)
- Fixed a problem with updating memo contents in DBFCDX (#782)
- Fixed a runtime exception when creating DBFCDX index files with long filenames (#774)
- Fixed a problem with with DBSeek() with active OrderDescend() finding even deleted records
- Fixed a problem with a missing call to AdsClearCallbackFunction() in the ADS RDD in OrderCreate() (#794)
- Fixed a problem with VODBOrdCreate function failing it the cOrder parameter contains an empty string (#809)

Macro compiler

- Fixed a problem in the Preprocessor
- Added support for parameters passed by reference with the @ operator
- Added support for M->, _MEMVAR-> and MEMVAR-> prefixes in the macro compiler
- When the Macro compiler finds 2 or more functions with the same name it now uses the same precedence rules that the compiler uses:
  - Functions in User Code are used first
  - Functions in the "Specific" runtimes (XSharp.VO, XSharp.XPP, XSharp.VFP, XSharp.Data) take precedence over the ones inside XSharp.RT and XSharp.Core
  - Functions in XSharp.RT take precedence over functions inside XSharp.Core

Visual Studio Integration

- In this build we have started to use the "Community toolkit for Visual Studio extensions" that you can find on GitHub. This toolkit contains "best practices" for code for VS Extension writers, like we are. As a result more code is now running asynchronously which should result in better performance.
- We have also started to remove 32 bit specific code that would become a problem when migrating to VS 2022 which is a 64 bits version if Visual Studio that is expected to ship in November 2021.

New features

- Added several new features to the editor
- The editor can now show divider lines between entities. You can enable/disable this in the options dialog (#280)
- Keyword inside QuickInfo tooltips are now colored (#748)
- Goto definition now also works on "external" types. The editor generates a temporary file that contains the type information for the external type. In the options dialog you can also control if the generated code should contains comments (as read from the XML file that comes with an external DLL). (#763)
- You can control which keyword is used for PUBLIC visibility from the Tools/Optons menu entry (PUBLIC, EXPORT or No modifier at all)
- You can control which keyword is used for PRIVATE visibility from the Tools/Optons menu entry

(PRIVATE or HIDDEN).
- The various code generators inside VS now follow the capitalization rules from the source code editor.
- The intellisense database now has views that return the unique namespaces in the source code and in the external assemblies
- The X# specific menu points in the Tools menu have been moved to a separate submenu.
- Added option for the WinForms designer to generate backup (.bak) files of form.prg and form.designer.prg files when saving (#799)

Bug Fixes

- Fixed several problems in the editor:
  - We have made several improvements to increase the speed inside the editor (#689, #701)
  - Fixed a problem in the type lookup of variables for FOREACH loops (#697)
  - Parameter tips were not shown for methods selected from a completion list (#706)
  - Keyword case synchronization did not work when the keyword was not followed by a space (#722)
  - Goto definition always went to line 1 / column 1 in the file where a function was defined (#726)
  - Code completion for Constant members of classes (#727)
  - Parameter tips for nested function calls required an extra space before the name of the nested function (#728)
  - QuickInfo for DEFINES (#730, #739)
  - VOSTRUCT Member completion with the '.' operator (#731)
  - The ENUM and FUNC keywords are now recognized as identifier and not case synchronized in these cases.(#732)
  - Fixed a problem when opening files (#742)
  - Fixed parameter tip display for default values NULL, NULL_DATE and NULL_OBJECT (#743)
  - Fixes broken parameter tips for constructors (#744)
  - Nested classes were not always handled correctly by the intellisense (#745)
  - Fixed a problem in the type lookup of variables declared with ARRAY OF <something> (#749)
  - The Editor could sometimes "freeze" when the buffer contained invalid code (#751)
- The code generator for Windows Forms was replacing tab characters with spaces. This has been fixed.(#438)
- Fixed a problem with the Form Designer corrupting code that contains EXPORT ARRAY OF <type>
- Fixed a problem with the Form Designer that when removing an event handler in the editor, some code was deleted (#812)
- Fixed a problem with the Form Designer converting EXPORT, INSTANCE and HIDDEN keywords to PUBLIC and PRIVATE (#802)

VO-Compatible Editors

- Now all VO-compatible editors support full Undo/Redo functionality. Also added cut/copy/paste functionality to the Menu editor
- Fixed several visual problems with VOWED controls in Design and Test mode (#741)
- Fixed a VS crash when Alt-Tabbing out of the editors, with the Properties window having focus (#764)
- Adjusted ComboBoxEx controls to have the same fixed height, as in VO. Also allowed the previous behavior, when the user has manually increased the height by more than 50 pixels, then this height is being used instead (#750)
- Added a bitmap thumbnail for the "Button Bmp" property of the Menu Editor in the Properties Window
- Added support for specifying a Ribbon in the Menu Editor. The ribbon (bitmap) to be used needs to

be specified as a filename in the properties of the Menu's main item (#714)
- Fixed some issues with event code generation in the Window Editor (#441, #46)

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