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

Changes in XSharp 3.1.0

Compiler

New Features

- Added the XBaseNet dialect and the /fox3 compiler option for FoxPro-compatible dot syntax; /fox3 is also available as a pragma option and defines the __FOX3__ preprocessor symbol
- Added support for FoxPro-style Cursor.Field expressions inside codeblocks when /fox3 is enabled, including better tracking of codeblock parameters during late binding
- Improved compiler handling of file-wide memvars and fields, which makes FoxPro-style name resolution more reliable in larger source files
- Added support for primary constructors and static interface constructors

Bug Fixes

- Fixed several /fox3 compatibility problems involving dot access to static and instance members, codeblocks, PUBLIC variables, and the special THISFORM/FindForm rewriting rules (#2025, #2054, #2055, #2056, #2059, #2060)
- Fixed a "Failed to emit module" error that could occur after an undefined symbol was followed by a CLIPPER-style call inside a method with a local procedure (#2031)
- Fixed PTR and NULL_PTR code generation so pointer comparisons and assignments work correctly on newer .NET versions (#2069)
- Fixed parser state corruption in shared compiler-server builds (/shared), which could cause random dialect and reference errors during parallel builds (#2076)
- Fixed primary-constructor edge cases, including empty primary constructors and preserving the primary constructor when a CLIPPER constructor is added
- Fixed duplicate import warnings for namespaces supplied through ImplicitNameSpace/ClassLibrary attributes, and corrected problems with LOCAL ARRAY, PUBLIC ARRAY and DIMENSION declarations
- Fixed a NullReferenceException in SemanticModel queries on compound assignments (#2096)
- Fixed an IndexOutOfRangeException in the compiler when analyzers and .editorconfig files are used (#2098)
- Fixed an issue with _DLL FUNCTION with PSZ arguments in .Net 5 or later (#2097)

- Fixes two issues when including C# files in the source files for X# (#2151 and #2153)
- Fixed an issue when assigning a lambda expression to a variable declared as USUAL or OBJECT. A lambda without a real target type is now converted to its natural delegate type (Func<...> or Action<...>) first and that delegate is then converted to the destination type
- Fixed random XS0266, XS1503 and XS0121 errors when compiling with /vo4 caused by the SuppressVo4 setting being shared between compiler threads (#2173)

Build System

- Added support for the /fox3 compiler option
- Values set in Directory.Build.Props for DefaultItemExcludes are now preserved
- Output folders are now excluded from the default globs
- Prevent automatic setting from Windows platform for SDK projects
- The Rules folder with XAML files is now included to support the new Rules based property dialog for SDK style projects

Runtime

New Features

- The target framework of the X# runtime is now .NET Framework 4.7
- Added GETPRINTER(), PUTFILE(), LOCFILE() and INPUTBOX(), and completed WAIT WINDOW support (#1997, #1996)
- Added VFP introspection functions and improved command coverage by absorbing CLEAR and RELEASE support (#1999, #2001)
- Implemented ATAGINFO(), XMLTOCURSOR(), CURSORTOXML(), KEYMATCH(), LOOKUP() and basic NEWOBJECT() support in the VFP runtime (#2087, #2070, #2016, #2004, #2003)
- Added _TALLY support for COUNT, REPLACE, DELETE, RECALL, SUM and AVERAGE (#2062, #2074)
- Expanded VFP database support with CLOSE DATABASES/CLOSE TABLES and DBC commands such as ADD TABLE, REMOVE TABLE and RENAME TABLE, plus support for long DBC field names
- Expanded XSharp.VFP.UI with ON KEY LABEL support, FormSet constructors, additional control properties, dynamic colors, and broader Grid and Menu support
- Implemented the VFP functions CNTBAR() and CNTPAD() (#2172)

Bug Fixes

- Fixed VFP command compatibility issues in CAST(), REPLACE default scope, SET DATE, SET SECONDS, COPY MEMO, IN-clause parsing, WAIT WINDOW message ordering, and SYSMETRIC() (#2013, #2008, #2038, #2030, #2017, #2029, #2053, #2065)
- Fixed SQLRDD crashes and data-integrity issues around reconnects, paging and EOF handling, delete and flush behavior, NULL and nullable ADS compatibility, tag lookup, index creation, Date and DateTime detection, and null-DataTable handling (#2035, #2044, #2050, #2058, #1984)
- Fixed FoxPro member-access and macro-compiler problems involving protected, private and static members, qualified static references, and NEWOBJECT() initialization parameters (#2056, #2057, #2039)
- Fixed Dbc.FindTable() to accept either an alias or a file name, and corrected long field-name collisions in DBFs
- Fixed several XSharp.VFP.UI problems in grids, record-pointer preservation, transparent controls, shapes, shortcut popups, property casing, InputMask and Format handling, and icon and form behavior
- Fixed wrong file offsets during shared file access on .NET 6 and later (#2082)

- Fixed field and member lookup in __FieldGetWa2 for objects that support dynamic properties, and improved the error that is thrown when a variable cannot be found (#2092)
- Fixed the usCulture used for conversions so it ignores Windows user overrides; this fixes Val(), Str() and Transform() when the user has a custom decimal symbol or digit grouping (#2100)
- Fixed an issue with rounding small negative numbers (-0.0) with new .Net versions #2104
- Fixed an issue with FindClass for types that are not yet loaded #2109
- Nested codeblocks in macro scripts can now access locals from the surroundng code #2113
- Allow implicit interface conversions in macros (#2114)
- Fixed an issue with the FoxPro directory function (#2137)
- Fixed an issue with late bound assigns to class properties in the FoxPro dialect
- Changed the return type of the internal function that is called for the SCATTER ARRAY command from Array to FoxPro compatible array (__FoxArray)
- SET EXACT ON in the VFP Dialect now ignores trailing blanks (#2160)
- CURDIR() in the VFP dialect returns the directory formatted as in VFP (#2160)
- Fixed STRTOFILE() additive clause for files that do not exist (#2160)
- DELETE FROM without a WHERE clause now deletes every record (#2167)
- CursorAdapter.DataSource in XSharp.VFP.UI now accepts non-string values (#2169)
- DELETE and RECALL with the IN clause now accept the clauses in any order (#2174)
- SET PATH TO with the ADDITIVE clause now adds the folders to the existing path (#2175)

Visual Studio Integration

New Features

- (Experimental) Added a WinForms Designer shadow-project bridge for SDK-style projects, including automatic synchronization on save and cleanup when projects close. This will be used for projects targeting .Net 8 and later. For .Net 4 the internal designer is used. The IDE will create a C# project with the same references and will cross compile the X# code to C# so it can be designed. Changes in the form will be cross compiled back to X# and double clicking event handlers will open the X# source files.
- Added SDK-style project reference management in the IDE, including "Add Assembly Reference", "Add Project Reference" and Dependencies-node integration (#2041, #2048)
- Added Target OS and runtime-identifier support to the SDK-style project property pages for modern .NET targets
- Updated Visual Studio integration for the new FoxPro and XBaseNet compiler options, including /fox3-aware settings and parser support for newer language constructs
- Added support for primary constructors and static interface constructors
- Implemented the Project context menu option "Convert to SDK format" for projects in the legacy format. The old project file is saved with a .BAK extension.
- The project Property pages are now based on the XAML files in the MsBuild Rules folder
- Added support for several new Intellisense features, contributed by one of our customers (Thanks HansJoerg)
    - Added support for Find All References in the source code editor (#2148)
    - Added support for Renaming types and members (#2154)
    - Added support for Navigate To (Ctrl-T) for types and members (#2155) (only in VS2022 and later)
    - Added support for Goto Implementation types and members (#2156)
    - Added Inlay hints for parameter names and VAR types (#2164)
    - Added Quick actions to add a USING statement, generate a method, generate overrides

and convert a field to a property (#2166)
- Goto Definition now also works for types and members in the C# and VB projects of the solution. F12 picks the correct overload (#2170)
- The Code Search window (Ctrl-T) now shows a preview for X# results and the preview no longer takes the focus (#2171)

Bug Fixes

- Fixed deadlocks and hangs when opening solutions or resolving references, including startup and open-from-command-line cases and UI-thread reference resolution (#2046, #2066)
- Reduced UI-thread marshalling and code-model database lock contention, which improves responsiveness and SDK-style project load performance (#2084, #2043)
- Improved handling of missing project files and prevented several exceptions and hierarchy hangs in the project system and designer support
- Fixed the completion "Max Entries" option resetting to 25 (#2088)
- Fixed threading issues in the VS Projectsystem (#2124, #2125, #2127)
- Fixed an issue with the type lookup for the loop variables of a FOREACH loop (#2129)
- Fixed an issue when closing a solution (#2138)
- Fixed an issue with the navigation bar in the source code editor (#2144)
- Fixed an issue when a project file contains an invalid configuration (#2145)
- For multi-targeting projects Goto Definition could show the same definition twice. This has been solved
- Fixed some issues when switching between Sqlite databases engines for the intellisense database. We now delete the database after changing the engine, so it will be rebuilt when the solution is reloaded.
- Fixed an issue with Goto Definition for partial classes in other projects (#2110)
- Fixed the Find References target selection after combining Find All References and Goto Implementation (#2161)
- The Rename dialog is now implemented in XAML and renaming a file in the project tree no longer throws an exception (#2163)
- Fixed the WinForms Designer shadow project: form events are now migrated from C# to X#, stale references to the shadow designer project were removed and the C# form is saved to X# before debugging starts
- SDK style projects now also add references to projects in other languages to the code model
- The error squiggle now has the length of the token at the error position
- Fixed an exception for an incomplete VAR statement
- The IDE no longer uses EnvDTE to find the startup project

VFPExporter

New Features

- Added a KeepFolderStructure option so exported projects can preserve the original VFP folder tree
- Added support for saving exported projects in XIDE format
- Added options to export each VFP library as its own project, with dependency-aware ordering, and added basic Function, Form and Menu start-point generation
- Added WITH/ENDWITH expansion during export, plus better ClassName-prefix handling and prefix-aware CreateObject() generation
- Added PageFrame and Page typed-class export, plus ReportListener support
- Added DBC metadata injection during export (Caption, InputMask and Format) and copying of

DBC-owned tables and companion files
- Expanded code conversion and menu and report support by normalizing legacy =< and => operators, handling alias/type collisions, commenting out EXTERNAL and DLL DECLARE statements, and adding extra VFP, menu and report UDC coverage

Bug Fixes
- Fixed exported VFP universe metadata to include FNAME() and FATTRIB() (#2015)
- Fixed SET SKIP conversion and improved continuing-line and comment handling during export
- Fixed output-folder handling, destination-folder cleanup, file-deletion messages, and missing ColorProperties.json deployment
- Fixed PropRules for RightToLeft, BorderStyle, DisabledColors, Icon, Picture, and dotted child-property overrides
- Fixed event-handler generation for multi-line LPARAMETERS and OptionGroup sub-control event names
- Fixed FormSet, form and menu export problems, menu MNX parsing and captions, generated project-reference paths, and DBC and DBF runtime path handling for free tables and OPEN DATABASE

Miscellaneous

New Features
- The templates for SDK projects now include .Net 11. Our preliminary tests show that our code compiles and runs fine with .Net 11.
- The rootnamespace property in the SDK projects is no longer "Company.Namespace1", but is derived from the name of the application
- The installer now adds the commandline option /queue:1 to the call to the ngen process for native code generation.
- We have added Help topics for the various variations of the FoxPro commands and for missing errors and warnings

Changes in XSharp 3

X# 3 has some breaking changes, so you should take the time to recompile and test all your existing projects and solutions.

Compiler

New Features
- The compiler no longer supports "Bring Your Own Runtime". If you want to compile in the VO or Vulcan dialect you will HAVE to include references to XSharp.Core and XSharp.RT.
  That does not mean that you can't link your program against the Vulcan runtime DLLs, but we no longer map our Xbase types to the types in these DLLs. So USUAL will not be mapped to Vulcan.__Usual. You can still use the types, functions and classes on these DLLs, but they will no longer be treated specially. If you want to call a function from the Vulcan Runtime yuou will have to use the syntax for static methods
- GLOBAL USING
- We have added file wide NAMESPACE support
- We have added support for the RECORD modifier for classes and structures
- We have added a new __XSHARP3__ define which is always true, so you can conditionally

compile differences between XSharp 3 and a previous version
- We have added code to check for the new [FoxArrayResultAttribute] that marks VFP functions which may create an array as a result (such as AMembers(), APrinters() and AUsed()). When the function is called with an undeclared variable then a new private variable is created for the result. (#1903)

Bug Fixes

- Suppressed a false ambiguity warning when a member has two overloads where one is parameterless and the other uses the Clipper calling convention; the compiler now silently chooses the parameter-less overload
- Fixed an issue with cref attributes that are read from include files
- Fixed an “Failed to emit module” error when the code contains an expression in the form of “UnnamedType.Property” or “UnnamedType.Method()” when the “allowdot” compiler option is enabled.
- The range expression rule in the compiler (a .. b) allowed both sides of the dots to be empty. This has been fixed. The compiler now supports “a .. b”, “a .. “ and “ .. b”
- Symbols and literal PSZs inside an interpolated string were not correctly processed. This has been fixed.
- Fixed a problem with text containing semicolon inside TEXT To (#1605)
- Fixed overload resolution when a Clipper calling convention function is called with a single NULL_OBJECT argument (#1782, #1016)
- Changed the UDC for the RECALL command, because FoxPro temporarily switches of the SetDeleted() setting during the command (#1838)
- Prevented an assertion error when compiling in debug mode for functions with the NeedsAccessToLocal attribute and a VOID return type (#1870)
- Fixed a "Type Expected" compiler error in the VO dialect when declaring a CLIPPER method with no arguments and no explicit return type (#1907)
- Suppress an error for missing XML for compiler generated code (#1920)
- Passing a literal NIL as argument for a type parameter now generates a compiler warning (#1946, #1967)
- Fixed an issue where certain errors were not shown causing a compilation to fail with "error XS7038: Failed to emit module" (#1951)
- Fixed an internal compiler error (#1966)
- Ensure INIT methods are called inside subclasses (#1988)

Build System

New Features

- The Build system now supports the new SDK style projects
- We have added a task to generate source code for GLOBAL USING values declared in the project files.
- The XAML files in the Rules subfolder of the MsBuild folder are no longer used and have been deleted. Also the references to these files in the .props and .targets files have been removed
- The builds system no longer automatically picks up the include folder from your Vulcan installation
- There are many changes in the .props and .targets files to support SDK style projects and .Net 10. One of these changes automatically sets dependency relations between files, such as between VOBinary files and the module that they belong to.
- The response file for the last build is now written to the intermediate output folder of the project with the name <OutputFileName>.rsp. A copy of this file is also written to the temp folder under

the name LastXSharpResponseFile.Rsp

Bug Fixes

- The Build task to compile native resources now checks the rc and include files for Byte Order Marks (BOM). When it finds such a BOM then it generates a warning
- Warning numbers for the Native Resource generation task now start with XS9xxx and no longer with MSB…
- For SDK projects File Foo.xh is now by default depending on Foo.prg when they both appear in the same folder
- Fixed an error reported by Stefan Hirsch, was caused by automatically adding an OSPlatformAttribute to the output assemblies
- Fixed a NuGet error about listing "win-x86" as runtime identifier for Old style projects with package references (#1801)
- Fixed an issue during build where an empty <PrebuildEvent> that contains a CRLF could cause a build to fail.

Runtime

The signature for several functions and methods in the XSharp Runtime has been changed.
We have used the new IN keyword for many that use parameters defined AS USUAL.
The new modifier tells the compiler to pass the variable by reference, which should make the code slightly faster.
It also tells the compiler that the parameter will not be changed by the code in the function / method
We have also moved several functions and types between Runtime DLLs, because they were not in the best possible location.
X# 3 comes with runtime DLLs for both .net 46 and .net 8. Not all VOSDK Dlls have been ported to .Net 8 because they contain code that does not work in an AnyCPU environment.
The NuGet packages for the runtime contain XML documentation for English and Chinese.

New Features

- We are now including .Net 8 versions of most of our runtime DLLs.
- The Runtime is now delivered in both DLL format as well as Nuget Packages. See the separate topic about the Runtime in this help file for a description of these packages and DLLs
- The record number in RDD operations is now a DWORD and no longer a LONG. The maximum record number in a DBF file is now 2^32 - 1
- Late bound code now gives precedence to the NEW method over a method defined in the parent class (#1286)
- We have added implicit conversion operators from USUAL to nullable value types(#1748, #1807)
- Added the Transform @N format function

Bug Fixes

- Fixed potential issues with Null_date returned from OpenADS for the Last Update date in a DBF
- Fixed an issue in the Macrocompiler for .Net 8
- Fixed an issue with Memo functions
- Fixed ADel() to correctly remove the element at the specified position
- Updated documentation for the Mb..() functions (#1124)
- Removed duplicate MAKELANGID() and MAKELCID() functions from the Win32APILibrary; they were already defined in another location (#1124)
- Added dummy stub implementations for Nation Functions in XSharp.VO (#1124)
- Fix a problem with the DBF cache when the same table is opened in 2 different workareas with a file name that only differs in case (#1828)

- Fix a problem with the DBF Header locking for FoxPro tables with autonumber fields (#1829)
- Fixed an issue in the Macrocompiler where fields and variables passed to a function with IN parameter would be updated. (#1933)
- Fixed an issue where assigning to a hidden field or property was incorrectly allowed. Added NoIVarGetSelf() and NoIVarPutSelf() methods (#1491)
- Fixed a macro compiler issue with certain method calls (#1860)
- Fixed a memo-related state issue (#1869)
- Fix Dbc.FindTable to support Alias and/or FileName (#1949)
- Improved the speed of the Directory Function (#1958)

XPP Dialect

- Added NLS_ defines and initial implementation for the SetLocale() function
- Fixed a late binding issue (#1836)
- Fixed an error in a UDC
- Added support for the Break() function (#1961)

New features the FoxPro dialect.

- The XSharp.VFP assembly now contains function definitions for all functions from VFP, even the ones that exist in XSharp.Core and XSharp.RT. These functions will delegate the work to the implementation in the other assemblies.
- Added an implementation of the AUsed() function (VFP)
- The implementation of several UI related functions, such as MessageBox(), have been moved from XSharp.VFP to XSharp.VFP.UI.
  These functions will now output to the console window when XSharp.VFP.UI cannot be found.
- Added [FoxArrayInputParameterAttribute] and an array result stack to support functions such as AMembers(), APrinters() and AUsed()
- Adjusted AMembers() and APrinters() to use the new array result mechanism
- Fixed Set.Compatible default return value: it was incorrectly returning string "OFF" instead of LOGIC FALSE
- Added Set.Compatible handler in the Set() function, supporting both string values ("ON", "OFF", "DB4") and LOGIC values
- The DBFVFP driver now allows index expressions that contain Trim(), Ltrim(), RTrim() etc. (#1837)
- Fixed FDate() to accept a filename parameter and return the actual last-write date of the file (#1856)
- Fixed FTime() to accept a filename parameter and return the actual last-write time of the file (#1856)
- Fixed FSize() VFP compatibility: when SET COMPATIBLE is ON and no work area is specified, it returns the size of the named file; when the filename has no extension, a .DBF extension is assumed (#1856)
- Added new file functions FName() (returns the filename from a path) and FAttrib() (returns file attributes) (#1856)
- Fixed COPY TO ARRAY (#1857)
- Added the [NeedsAccessToLocals] attribute to VFP Macro Wrapper functions to preserve correct macro evaluation behavior (#1858)
- Added SET POINT and SET SEPARATOR support (#1867, #1883)
- Fixed RELEASE ALL LIKE when a quoted pattern is used in the skeleton (#1870)
- Added support for the SET BELL TO command (the command is accepted and ignored, matching VFP behavior) (#1874)
- Implemented NORMALIZE() and DISPLAYPATH() functions (#1880)

- Fixed VARTYPE() throwing exceptions for undeclared variables (#1882)
- Fixed SET DEFAULT TO command paths and working directory handling (#1884)
- Implemented Lock(), RLock(), FLock() and UnLock() for SQLRDD (#1885)
- VFP GUI provider now loads dynamically via AssemblyHelper (#1888)
- Fixed RELEASE ALL LIKE when quoted patterns are passed in the skeleton (#1889)
- Added missing parameterless and IntPtr overloads for F* file functions (#1890)
- Fixed SET MEMOWIDTH, SYSMETRIC and COPY TO ARRAY (#1891)
- Refined SET('DEFAULT') and SET('DIRECTORY') (#1892)
- Implemented FCHSIZE() and VFP-compatible BINTOC() / CTOBIN() (#1893)
- Fixed SYSMETRIC() calls, SET MEMOWIDTH typo, and SET DEFAULT/DIRECTORY initialization issues at startup (#1894)
- Fixed MemoHelpers.MAX_WIDTH to be a dynamic property (#1895)
- Implemented GETDIR(), GETFILE() and GETPICT() dialog functions (#1896)
- Fixed MESSAGEBOX() timeout issue and implemented dynamic parameter handling (#1897)
- Implemented DIR/DIRECTORY command with output parity to VFP (#1900)
- Implemented SET DEVICE TO FILE redirection (#1901)
- Implemented ISPEN() (returns FALSE — no Tablet support) and IMESTATUS() (returns 0 — no IME support) (#1924)
- Implemented PROW() and PCOL() printer position functions (#1925)
- Implemented TXNLEVEL() which returns the current transaction nesting level (#1927)
- Implemented ISMEMOFETCHED(); for local DBF data, memo fields are always considered fetched (#1931)
- Fixed CREATE TABLE when the table name includes a path; the alias is now derived from the file name without the directory part (#1938)
- Implementation of GETFLDSTATE and SETFLDSTATE (#1952)
- Implements (partial) functions CURSORSETPROP() AND CURSORGETPROP() (#1954)
- Allow TRIM()/ALLTRIM()/RTRIM() in Index expressions (#1969)
- Fixed an issue with FONTMETRIC() (#1976)
- Fixed an issue with COPY TO CSV/SDF for date fields (#1977)
- Fixed an issue with Strtran(), ExecScript() and Set() (#1980)

Visual Studio Integration

We are now fully supporting Visual Studio 2026
We are no longer supporting Visual Studio 2017.
We are still supporting Visual Studio 2019 but not all of the new features will be implemented for this version of VS.

New Features

- We are now supporting SDK style projects. To do so many changes were made all over the Visual Studio support code
- The language service now respects the new commands, such as GLOBAL USING, NAMESPACE, RECORD etc.
- Added support for the VS Document Outline Window
- We have added new templates for SDK style projects
- The Project Property windows for SDK projects are now in WPF and support several new properties and have several new pages.
- Moved logging and most VS event handlers to a new Support DLL. Fixed an issue in the IDE Build Logger (#1802).

- We have added an option to edit a project file without unloading it. Saving the changes will trigger reloading of the project (#1817)
- Added Publish and Pack commands to the Build menu for SDK-style X# projects; a dialog allows configuring the target OS, self-contained deployment, and single-file output (#1939)

Bug Fixes

- The Output window in Visual Studio was not always showing progress. This has been fixed
- The assemblies that are automatically added for SDK projects now show more properties inside the property windows.
- Improved load speed for SDK projects..
- Fixed an issue where the first Build inside VS with a solution that has X# projects would seem to freeze VS. After pressing Cancel a second Build (or rebuild) would work.
- The “Toggle Comment” command now takes the comment status of the first line of a block of lines to decide if the block needs to be commented or uncommented. When a line inside a block is already commented then it may end up being double commented after a comment operation. And when a block containing a double commented line is uncommented, than that line will become single commented afterwards.
- When a project has multiple project references to the same external project, then problems could occur. This has been fixed. The second reference gets (silently) ignored.
- The document formatter now excludes lines that are part of a TEXT ... ENDTEXT
- Project References in SDK projects no longer store the name, guid etc.
- The Build/Cancel button was not visible when a X# project was selected and another project was being built. (#1799)
- Fixed an error “the type initializer for 'XSharp.Project.ErrorListManager' threw an exception” when loading projects in VS2026 (#1800)
- Fixed issue where the project system was incorrectly responding to the Package References project dependency. This caused the first Build in Visual Studio to freeze. (#1802)
- The include-files node could sometimes show files, even after the /nostddef compiler option was selected on the project properties page. We recommend a touch of all the files in a project to completely refresh (clear) the include list. (#1805)
- Sometimes VS would freeze when adding an assembly reference. This has been fixed (#1814)
- Fixed an issue with duplicate reference names in the Solution Explorer window (#1816)
- We no longer add Project Guid properties to SDK projects. Guids are also removed from project references inside SDK projects. (#1819)
- The IDE has been prepared for a menu option to convert a traditional project to the new SDK project file format. (#1823). The menu option is not working yet.
- Fixed a crash when Visual Studio was opened directly with a solution file (#1918)
- Replaced bitmap-backed command icons and project hierarchy icons with Visual Studio Image Monikers (#1921)
- Fixed a deadlock when opening a solution from the command line (#1922)
- Fixed incorrect conditional compilation block highlighting: the language service now uses the correct preprocessor defines from the active project configuration (#1923)
- Fixed an issue with Project References for SDK projects where these references were not shown in the "Add Reference" dialog (#1928)
- Fixed an issue with Project References for SDK projects where these references were not shown in the Project Dependencies page of the Solution Properties window ((#1929)
- Local functions were not always appearing correctly in the Navigation Bar and Document Outline window (#1932)
- Fixed the navigation bar not displaying all entities in files with nested classes, and not following

the cursor when it moves past the last declared entity (#1934)
- Fixed an issue where Goto Definition would fail when a project folder contains another project and the option "Show All Files" was enabled, showing that other project as "Non-member Item".
- Sometimes reading of type information from Assembly References fails. This has been fixed. (#1956)
- The intellisense database was not removing entries from assemblies that were no longer in use by any project.

Debugger

New Features

- The work area window now also shows index information and the table structure

Bug Fixes

- Fixed an issue where the debugger was not correctly displaying Symbol values inside Usuals (#1804)
- Fixed an issue in the debugger where USUAL values were not shown properly formatted (#1809)
- Fixed an issue in the debugger where breakpoint conditions and watch expressions were not working (#1810)

VOSDK

New Features

- Methods named FIELDGET and FIELDPUT (in uppercase) are now named FieldGet and FieldPut to be compatible with the case of other methods.

Typed VOSDK

New Features

- The menu and toolbar class now internally use the MenuStrip and ToolStrip classes from Windows Forms because the classes that were used before are no longer supported in newer versions of .Net.