

Silent Installation of Altova Desktop Developer Applications and Excel XBRL Add-Ins

Support for built-in licensing metering mechanism and LicenseServer registration

These instructions are based on Version 2025 of our software, though they support Version 2019 Release 3 and newer. For other versions please adjust the major version number as required. The following products are supported as of this writing.

- MissionKit
- XMLSpy
- MapForce
- StyleVision
- UModel
- SchemaAgent
- DatabaseSpy
- DiffDog
- Authentic Desktop
- EBA XBRL Add-In for Excel
- ESEF XBRL Add-In for Excel
- EIOPA (Solvency II) XBRL Add-In for Excel
- WIP XBRL Add-In for Excel

You should determine in advance whether you intend to use LicenseServer centrally to administer, monitor, and assign the software's licenses, or the standard built-in license metering mechanism.

Prerequisites

You will need Microsoft's `makecab` utility. If the command `makecab` does not show the following output on a console window, [download and install the Windows SDK from Microsoft](#).

```
> makecab
Cabinet Maker - Lossless Data Compression Tool[...]
```

Preparation

[Download the relevant program's installer from our website](#), then install and run the program at least once in order to create the license-activation files – these will have the file extension `.licbin` or `.licsvr`, depending on whether LicenseServer is being used. The license-activation files will be located in the folder `C:\ProgramData\Altova\PROGRAM_NAME` (note: `ProgramData`, not `Program Files`).

`PROGRAM_NAME` is the name of the actual program installed (e.g., `XMLSpy2025`). The resulting non-LicenseServer license files are named `PROGRAM_NAME.licbin` (e.g., `xmlspy.licbin`), and the LicenseServer-based license files are named `PROGRAM_NAME.licsvr` (e.g., `xmlspy.licsvr`).

Files

Into a new empty folder, copy the following files:

1. The product's installer executable (e.g., `MissionKitProf2025.exe` or `XMLSpyEnt2025_x64.exe`).
2. The license file (or files in case you are working with a MissionKit license) that you want to install with the product. For the MissionKit, you may provide license files for as many products as you wish. Do not keep any other `.licbin` or `.licsvr` files in that folder. Note the following points:
 - In case you observe the text `_new` as part of the `.licbin` file name, remove that from the copied file.
 - The script *will not* process the license file attached to your permanent license e-mail message (i.e., the one with the extension `altova_licenses` or `altova_license_server`).
3. The Visual Basic script. Ensure the file name has a `.vbs` extension. For the complete script, see the subsection *Visual Basic Script* below.

Execution

Open the command prompt or PowerShell console as an administrator and change to the prepared folder using the `cd` or `dir` command. Then execute the following command at the ">" prompt.

```
> cscript.exe <VisualBasicScript_NAME>.vbs
```

Follow the installation instructions at the end of the console output, which will look similar to the following:

```
To install you need following files in the same folder:
MissionKitEnt2025.exe
LicsvrFiles.cab
MissionKitEnterprise2025_LicFile.mst
On a command prompt change to that folder and call:
MissionKitEnt2025.exe TRANSFORMS=MissionKitEnterprise2025_LicFile.mst
For silent installation use the option '/q '
For logging use the option '/L*xv! <log_file_name> '
```

Incorporate the command and options into your deployment scheme as appropriate.

Caveats

Migration of previous license or LicenseServer registration

The installer will attempt to migrate any `.licbin` or `.licsvr` file found in the previous major version's `ProgramData` folder (e.g., `C:\ProgramData\Altova\XMLSpy2025`). This means it is possible to end up with both `.licbin` and `.licsvr` files in the new `ProgramData` folder (e.g., `C:\ProgramData\Altova\XMLSpy2025`).

If your deployment involves a change from a local license (`*.licbin`) to a LicenseServer registration (`*.licsvr`), incorporate a step manually to remove the unwanted file from the previous major version's `ProgramData` folder prior to the installation of the new version.

Visual Basic script

Provided below is the complete Visual Basic script (see *Step 3 in the subsection Preparation/Files above*). Copy the code listing to a file with a .vbs extension.

```
Rem -----
Rem v2024, 2023-11-10T14:06:
Rem 1. added support for Excel XBRL add-ins.
Rem 2. Corrected Fail message in Function getProductname to refer to
licFileBaseName rather than licFileName.
Rem 3. Incorporated numerous Debug messages.
Rem v2019r3:
Rem 1. Updates to work beginning with this release - names of directories and
features changed.
Rem -----
Const msiOpenDatabaseModeReadOnly = 0
Const msiOpenDatabaseModeTransact = 1
Const msidbFileAttributesCompressed = 16384
Const msiViewModifyInsert          = 1
Rem -----

Dim objShell, objFSO, strCurDir
Set objFSO = CreateObject("Scripting.FileSystemObject")
Set objShell = CreateObject("WScript.Shell")

objStartFolder = objShell.CurrentDirectory

REM find the installer executable
installerFileName = getFileWithExtension("EXE")
if installerFileName = "" then
    Fail "You need to run this script from the directory where the installer EXE
and the .licsvr files are prepared."
end if
```

```

Rem extract the MSI contained in the installer executable.
Set oExec = objShell.Exec(installerFileName & " /u /qn+")
Do While oExec.Status = 0
    WScript.Sleep 100
Loop

Rem locate the MSI file
msiFileName = getFileWithExtension("MSI")
if msiFileName = "" then
    Fail "Failed to extract MSI from installer executable."
end if

Rem parse out attributes from installer file name
Rem product name, edition and major version year
pos = InStr(installerFileName, "Ent")
if pos > 0 then
    editionName = "Enterprise"
    productName = Left(installerFileName, pos - 1)
else
    pos = InStr(installerFileName, "Prof")
    if pos > 0 then
        editionName = "Professional"
        productName = Left(installerFileName, pos - 1)
    else
        pos = InStr(installerFileName, "Cmu")
        if pos > 0 then
            editionName = "Community"
            productName = Left(installerFileName, pos - 1)
        else
            pos = InStr(installerFileName, "Basic")
            if pos > 0 then
                editionName = "Basic"
            end if
        end if
    end if
end if

```

```

        productName = Left(installerFileName, pos - 1)

    else

        pos = InStr(installerFileName, "SchemaAgent")

        if pos > 0 then

            editionName = ""

            productName = "SchemaAgent"

        else

            pos = InStr(installerFileName, "ESEFXBRLAddIn")

            if pos > 0 then

                editionName = ""

                productName = "ESEFXBRLAddIn"

            else

                pos = InStr(installerFileName, "WIPXBRLAddIn")

                if pos > 0 then

                    editionName = ""

                    productName = "WIPXBRLAddIn"

                end if

            end if

        end if

    end if

end if

Debug.WriteLine("installerFileName: " & installerFileName)

Debug.WriteLine("installerFileName: " & installerFileName)

Debug.WriteLine("editionName: " & editionName)

Debug.WriteLine("productName: " & productName)

if ( productName = "" ) then

    Fail "Internal error: failed to extract product name or edition from  
installer file name."

end if

pos = InStr(installerFileName, "20")

```

```

if pos = 0 then

    Fail "Internal error: failed to extract major version year from installer
file name."

end if

majorVersionYear = "20" + Mid(installerFileName, pos + 2, 2)

WScript.Echo "Creating license transformation file for " & productName & " " &
editionName & " " & majorVersionYear

Rem cab all lic files

createCabFile "LicsvrFiles.cab", "LICSVR", "LIC", "LICBIN"

Rem open the MSI database

On Error Resume Next

Dim installer : Set installer = Nothing

Set installer = Wscript.CreateObject("WindowsInstaller.Installer") : CheckError

Dim database, openMode

tempMSIFile = "tmp.msi"

objFSO.CopyFile msiFileName, tempMSIFile, true

Set database = installer.OpenDatabase(tempMSIFile, msiOpenDatabaseModeTransact) :
CheckError

Rem find out next free file sequence number

firstFreeSequenceNumber = msiFindNextFreeSequenceNumber()

actualSequenceNumber = firstFreeSequenceNumber

Rem add installation of every lic file to MSI database

For Each objFile in objFSO.GetFolder(objShell.CurrentDirectory).Files

    If UCase(objFSO.GetExtensionName(objFile.name)) = "LICSVR" or
UCase(objFSO.GetExtensionName(objFile.name)) = "LIC" or
UCase(objFSO.GetExtensionName(objFile.name)) = "LICBIN" Then

        Rem in case of MissionKit the product name and cannot be used

        appName = getProductname(productName,
objFSO.GetBaseName(objFile.name))

```

```

    Rem add license file to file table
    msiAppendToFileTable objFile.name, actualSequenceNumber
    actualSequenceNumber = actualSequenceNumber + 1

    Rem add component that will install this license file
    licenseDirectory = appName & "COMMONAPPPDATA"

    If Instr(productName, "WIP") > 0 Then licenseDirectory =
"WIPTemplateCOMMONAPPPDATA"

    msiAppendToComponentTable objFile.name, objFile.name, licenseDirectory

    parentFeatureName = appName & "_Binaries"

    If Instr(productName, "ESEF") > 0 Then parentFeatureName =
"ESEFAddInBinaries"

    If Instr(productName, "EBA") > 0 Then parentFeatureName =
"EBAAAddInBinaries"

    If Instr(productName, "SolvencyII") > 0 Then parentFeatureName =
"SolvencyIIAddInBinaries"

    If Instr(productName, "WIP") > 0 Then parentFeatureName =
"WipTemplateBinaries"

    Debug.WriteLine("parentFeatureName: " & parentFeatureName)

    Rem add feature that will install component
    msiAppendToFeatureTable objFile.name, parentFeatureName
End If
Next

Rem append LicsvrFiles.cab to media table
msiAppendToMediaTable "LicsvrFiles.cab", actualSequenceNumber - 1

Rem save database
database.Commit

Rem create MSI database transform as a difference between original and modified
database

```

```

WScript.Echo "Creating transformation..."

transformationFileName = productName & editionName & majorVersionYear &
"_LicFile.mst"

Dim databaseOriginal : Set databaseOriginal = installer.OpenDatabase (
msiFileName, msiOpenDatabaseModeReadOnly ) : CheckError

Dim different:different = database.GenerateTransform(databaseOriginal,
transformationFileName) : CheckError

database.CreateTransformSummaryInfo databaseOriginal, transformationFileName, 0, 0


Set database = Nothing
Set databaseOriginal = Nothing
Set installer = Nothing


Rem clean-up
rem objFSO.DeleteFile(tempMSIFile)
rem objFSO.DeleteFile(msiFileName)


WScript.Echo "Success!"
WScript.Echo ""
WScript.Echo "To install you need following files in the same folder:"
WScript.Echo "    " & installerFileName
WScript.Echo "    " & "LicsvrFiles.cab"
WScript.Echo "    " & transformationFileName
WScript.Echo "On a command prompt change to that folder and call:"
WScript.Echo "    " & installerFileName & " TRANSFORMS=" & transformationFileName
WScript.Echo "For silent installation use the option '/q'"
WScript.Echo "For logging use the option '/L*xv! <log_file_name>'"


Rem -----
Rem create a cab file that contains all files with specified extension.
Rem -----
Sub createCabFile(cabFileName, fileExtension1, fileExtension2, fileExtension3)
    Rem create the DDF file needed to cab eventually more than one file

```



```

set ddfFile = objFSO.CreateTextFile("LicsvrFiles.ddf", true)
ddfFile.WriteLine(".SET CabinetName1 = LicsvrFiles.cab")
ddfFile.WriteLine(".SET Compression = LZX")
ddfFile.WriteLine(".SET CompressionMemory = 21")
ddfFile.WriteLine(".SET Compress=ON")
ddfFile.WriteLine(".SET Cabinet=ON")
ddfFile.WriteLine(".Set MaxDiskSize=CDROM")
ddfFile.WriteLine(".SET DiskDirectoryTemplate = ")
ddfFile.WriteLine(".SET infFileName=LicsvrFiles.inf")
ddfFile.WriteLine(".SET rptFileName=LicsvrFiles.rpt")

For Each objFile in objFSO.GetFolder(objShell.CurrentDirectory).Files
    If UCase(objFSO.GetExtensionName(objFile.name)) = fileExtension1 or
UCase(objFSO.GetExtensionName(objFile.name)) = fileExtension2 or
UCase(objFSO.GetExtensionName(objFile.name)) = fileExtension3 Then

        Rem append file to DDF
        ddfFile.WriteLine("""" & objFile.name & """)

    End If
Next

ddfFile.Close

Rem create the CAB file based on information in the DDF file
Set oExec = objShell.Exec("makecab /F LicsvrFiles.ddf")
Do While oExec.Status = 0
    WScript.Sleep 100
Loop
if oExec.ExitCode <> 0 then
    WScript.Echo "Creation of LicsvrFiles.cab failed."
    WScript.Quit 4
end if

Rem delete all temporary files

```

```

        objFSO.DeleteFile("LicsvrFiles.ddf")

        objFSO.DeleteFile("LicsvrFiles.inf")

        objFSO.DeleteFile("LicsvrFiles.rpt")
End Sub

Rem -----
Rem helper function to get first file with specified file extension
Rem -----
Function getFileWithExtension(fileExtension)
    For Each objFile in objFSO.GetFolder(objShell.CurrentDirectory).Files
        If UCase(objFSO.GetExtensionName(objFile.name)) = fileExtension Then
            getFileWithExtension = objFile.Name
            Exit Function
        End If
    Next
End Function

Rem -----
Rem hard-code product name to get them in correct writing
Rem -----
Function getProductName(installerBaseName, licFileBaseName)
    Debug.WriteLine("installerBaseName: " & installerBaseName)
    Debug.WriteLine("licFileBaseName: " & licFileBaseName)
    if UCase(licFileBaseName) = "XMLSPY" then
        if UCASE(installerBaseName) = "AUTHENTIC" then
            getProductName = "Authentic"
        else
            getProductName = "XMLSpy"
        end if
    elseif UCase(licFileBaseName) = "MAPFORCE" then
        getProductName = "MapForce"
    elseif UCase(licFileBaseName) = "STYLEVISION" then

```

```

        getProductName = "StyleVision"
elseif UCase(licFileBaseName) = "UMODEL" then
        getProductName = "UModel"
elseif UCase(licFileBaseName) = "SCHEMAAGENT" then
        getProductName = "SchemaAgent"
elseif UCase(licFileBaseName) = "DATABASESPY" then
        getProductName = "DatabaseSpy"
elseif UCase(licFileBaseName) = "DIFFDOG" then
        getProductName = "DiffDog"
elseif UCase(licFileBaseName) = "" then
        getProductName = "XMLSpy"
elseif UCase(licFileBaseName) = "XMLSPY" then
        getProductName = "XMLSpy"
elseif UCase(licFileBaseName) = "EBAADDINFOREXCEL" then
        getProductName = "EBAAddInForExcel"
elseif UCase(licFileBaseName) = "ESEFADDINFOREXCEL" then
        getProductName = "ESEFAddInForExcel"
elseif UCase(licFileBaseName) = "SOLVENCYIIADDINFOREXCEL" then
        getProductName = "SolvencyIIAddInForExcel"
elseif UCase(licFileBaseName) = "WIPADDINFOREXCEL" then
        getProductName = "WIPAddInForExcel"
else
        Fail "Product '" & installerBaseName & "' or license file '" &
licFileBaseName & "' not supported by this script"
    end if
end Function

Rem -----
Rem find the highest sequence number used in the file table and return +1
Rem -----
Function msifFindNextFreeSequenceNumber()
    Dim view

```

```

Set view = database.OpenView("SELECT Sequence from File") : CheckError
view.Execute : CheckError

Dim record, maxRow, tempRow
maxId = 0
Do
    Set record = view.Fetch : CheckError
    if record is Nothing Then Exit Do
    if not record is Nothing Then
        tempId = record.IntegerData(1)
        if ( tempId > maxId ) Then
            maxId = tempId
        End If
    End if
Loop

view.Close
msiFindNextFreeSequenceNumber = maxId + 1
end Function

Rem -----
Rem append file entry to file table
Rem -----

Sub msiAppendToFileTable(fileName, sequenceNumber)
    WScript.Echo "Adding license file to MSI database..."
    Set view = database.OpenView( "SELECT File, Component_, FileName, FileSize,
Attributes, Sequence, Version from File" ) : CheckError
    view.Execute : CheckError

    ' Create the new Record
    Set record = Installer.CreateRecord (7)
    Dim lLicFileObject, updateMode

```

```

Set lLicFileObject = objFSO.GetFile ( fileName ) : CheckError

record.StringData (1) = fileName      ' file ID - name in CAB
record.StringData (2) = fileName      ' component name
record.StringData (3) = fileName      ' target file name
record.IntegerData(4) = lLicFileObject.Size
record.IntegerData(5) = msidbFileAttributesCompressed
record.IntegerData(6) = sequenceNumber
record.StringData (7) = "1.0"

view.Modify msiViewModifyInsert, record

view.Close

end sub

Rem -----
Rem add new component entry. component name is the same as the file id
Rem directoryId must be listed in the directory table
Rem -----
Sub msiAppendToComponentTable(fileId, componentId, directoryId)
    WScript.Echo "Adding component to MSI database..."
    ' Components
    Set view = database.OpenView( "SELECT Component, ComponentId, Directory_,
Attributes, KeyPath from Component" ) : CheckError
    view.Execute : CheckError

    ' Create the new Record
    Dim guidForNewComponent, TypeLib
    Set TypeLib = CreateObject("Scriptlet.TypeLib")
    guidForNewComponent = TypeLib.Guid

    Set record = Installer.CreateRecord (5)

```

```

        ' SELECT Component, ComponentId, Directory_, Attributes, KeyPath from
Component
        record.StringData (1) = componentId
        record.StringData (2) = left(guidForNewComponent, len(guidForNewComponent)-
2)
        record.StringData (3) = directoryId
        ' a permanent component so that the lic file doesn't get uninstalled with the
product
        record.IntegerData(4) = 16
        record.StringData (5) = fileId

        Rem Wscript.Echo "Inserting row: " & fileId & ", " &
left(guidForNewComponent, len(guidForNewComponent)-2) & ", " & sDirectoryParameter
& ", " & sLicShortFileName

        view.Modify msiViewModifyInsert, record
        view.Close

        Set TypeLib = Nothing
end sub

Rem -----
Rem add new component entry. component name is the same as the file id
Rem directoryId must be listed in the directory table
Rem -----
Sub msiAppendToFeatureTable(componentId, parentFeatureId)
    WScript.Echo "Adding feature to MSI database..."
    'FeatureComponents
    Set view = database.OpenView( "SELECT Feature_, Component_ from
FeatureComponents" ) : CheckError
    ' SELECT Feature_, Component_ from FeatureComponents
    view.Execute : CheckError
    ' Create the new Record
    Set record = Installer.CreateRecord (2)
    ' SELECT Feature_, Component_ from FeatureComponents
    record.StringData (1) = parentFeatureId

```

```

        record.StringData (2) = componentId

        Rem WScript.Echo "Insert feature " & parentFeatureId & " - " & componentId
        view.Modify msiViewModifyInsert, record
        view.Close
    end sub

    Rem -----
    Rem add new media table entry
    Rem -----

    sub msiAppendToMediaTable(cabFile, maxSequenceNumber)
        WScript.Echo "Adding media entry to MSI database..."

        Set view = database.OpenView( "SELECT DiskId, LastSequence, Cabinet from
Media" ) : CheckError

        view.Execute : CheckError

        ' Create the new Record
        Set record = Installer.CreateRecord (3)
        ' SELECT DiskId, LastSequence, Cabinet from Media
        record.IntegerData(1) = maxSequenceNumber
        record.IntegerData(2) = maxSequenceNumber
        record.StringData (3) = cabFile

        view.Modify msiViewModifyInsert, record
        view.Close
    end sub

    Rem -----
    Rem Check for error during last call including MSI and quit script in case of
    failure.
    Rem -----

    Sub CheckError
        Dim message, errRec

```

```
    If Err = 0 Then Exit Sub
    message = Err.Source & " " & Hex(Err) & ": " & Err.Description
    If Not installer Is Nothing Then
        Set errRec = installer.LastErrorRecord
        If Not errRec Is Nothing Then message = message & vbNewLine &
errRec.FormatText
    End If
    Fail message
End Sub

Rem -----
Rem Quit script
Rem -----
Sub Fail(message)
    Wscript.Echo message
    Wscript.Quit 2
End Sub
```