```
Option Declare
UseLSX "*javacon"

%REM
Agent LinkJavaLibToSSJS

Example JSS function (with static method but you can create a class instance to call a dynamic
method):

function getMyClassMethod() {
    // This function returns the result of the method com.mycompany.tools.MyClass.myMethod()
    importPackage(com.mycompany.tools);
    return MyClass.myStaticMethod();
}
%END REM

Private Const APP_ERR_CODE = 1001
Private Const NoCaseNoPitch = 5

Public Const wAPIModule = "nnotes.dll"  ' Windows/32
Declare Function MailAddMessageBodyComposite Lib wAPIModule (ByVal hMessage As Long, ByVal ItemName
As String, ByVal InputFileName As String) As Integer
Declare Function MailGetMessageBodyComposite Lib wAPIModule (ByVal hMessage As Long, ByVal ItemName
As String, ByVal OutputFileName As String, OutputFileSize As Long) As Integer
Declare Function OSLoadString Lib wAPIModule Alias "OSLoadString" (ByVal hModule As Long, ByVal
stringCode As Integer, ByVal retBuffer As String, ByVal bufferLength As Integer) As Integer
Declare Function GetTempPath Lib "kernel32" Alias "GetTempPathA" (ByVal nBufferLength As Long,
ByVal lpBuffer As String) As Long

Public Const LONGRECORDLENGTH = &h0000
Public Const SIG_CD_FILESEGMENT = 96
Public Const SIG_CD_FILEHEADER = 97
Private Const CD_MAX_SEGMENT_SIZE = 10240

Type LSIG
    Signature As Integer ' SIG_CD_ constant ORed with LONGRECORDLENGTH
    Length As Long ' (length is inclusive with this struct)
End Type

Type CDFILEHEADER
    Header As LSIG ' Signature and Length
    FileExtLen As Integer ' Length of file extension
    FileDataSize As Long ' Size (in bytes) of the file data
    SegCount As Long ' Number of CDFILESEGMENT records expected to follow
    Flags As Long ' Flags (currently unused)
    Reserved As Long ' Reserved for future use
    ' Variable length string follows (not null terminated). This string is the file extension for
the file.
    FileExt As String
    ' Header must consist of even number of bytes. If no (depending on the file extension)
    ' then it must be appended with the alignment filler (zero byte) and it must be considered in
the
    ' property Header.Length
End Type

Type CDFILESEGMENT
    Header As LSIG ' Signature and Length
    DataSize As Integer ' Actual Size of image bits in bytes, ignoring any filler
    SegSize As Integer ' Size of segment, is equal to or larger than DataSize if filler byte added
to maintain word boundary
    Flags As Long ' currently unused, but someday someone will be happy this is here
    Reserved As Long ' Reserved for future use
    ' File bits for this segment plus a possible alignment filler (in the last segment)
End Type

Sub Initialize
    Dim session As New NotesSession
    Dim db As NotesDatabase
    Dim nc As NotesNoteCollection
    Dim noteID As String
    Dim docJavaLib As NotesDocument

    Set db = session.CurrentDatabase

     ' Filter Java script libraries (not JavaScript libraries!)
    Set nc = db.CreateNoteCollection(False)
    nc.SelectScriptLibraries = True
```

```
    nc.SelectionFormula = {$Flags = "sj34Q"}
    Call nc.BuildCollection
    If nc.Count > 0 Then
        noteId = nc.GetFirstNoteID
        While noteId <> ""
            Set docJavaLib = db.GetDocumentByID(noteId)
            Call PutJavaLibToWebResource(docJavaLib)
            noteId = nc.GetNextNoteID(noteID)
        Wend

        MessageBox "Execution completed, all Java libraries processed"
    Else
        MessageBox "Execution completed, no Java libraries processed"
    End If
End Sub

Sub PutJavaLibToWebResource(docJavaLib As NotesDocument)
    Dim jarFilePath As String

    jarFilePath = ExtractJarFileFromJavaLib(docJavaLib)
    If jarFilePath <> "" Then
        Call CreateWebFileResource(jarFilePath, "WEB-INF/lib/")
        Kill jarFilePath
    End If
End Sub

Function ExtractJarFileFromJavaLib(docJavaLib As NotesDocument) As String
    Dim libName As String
    Dim att As NotesEmbeddedObject
    Dim extractedFilePath As String

    ' If no attached compiled file
    ExtractJarFileFromJavaLib = ""

    libName = docJavaLib.GetItemValue("$TITLE")(0)
    ' Extract attached compiled .jar file
    ' (notice that Java library note contains two attached files:
    ' - compiled: "%%object%%.jar"
    ' - source code: "%%source%%.jar"
    Set att = docJavaLib.GetAttachment("%%object%%.jar")
    If att  Is Nothing Then
        Exit Function
    End If

    ' Generate the temporary file name
    ' Remove the possible library extension e.g. ".java" or ".j"
    extractedFilePath = SeparateDir(GetOSTempDir) & ExtractFileNameRoot(libName) & ".jar"
    ' Remove existing file
    If FileExists(extractedFilePath) Then
        Kill extractedFilePath
    End If
    ' Extract
    Call att.ExtractFile(extractedFilePath)

    ExtractJarFileFromJavaLib = extractedFilePath
End Function

Sub CreateWebFileResource(ByVal binFilePath As String, ByVal resourcePath As String)
    Dim session As New NotesSession
    Dim db As NotesDatabase
    Dim nc As NotesNoteCollection
    Dim tmpCDFilePath As String
    Dim binFileSize As Long
    Dim resourceTitle As String
    Dim dxlContent As String
    Dim dxlImporter As NotesDXLImporter
    Dim stream As NotesStream
    Dim docDesignRsrc As NotesDocument
    Dim docTmp As NotesDocument
    Dim docDesignID As String
    Dim apiResult As Integer
    Dim errDescr As String
    Dim rtItem As NotesRichTextItem

    Set db = session.CurrentDatabase

    resourceTitle = resourcePath & ExtractFileName(binFilePath)
```

```
    tmpCDFilePath = ExtractFileNameRoot(binFilePath) & ".cd"
    tmpCDFilePath = GetTempFilePath("", tmpCDFilePath)

    ' Create CD file and embed it into the temporary document.
    ' For an unknown reason, could not embed the CD file directly into the file resource created
    ' through DXL - probably, the called C API method does not process design elements properly
    Set docTmp = db.CreateDocument
    Call docTmp.ReplaceItemValue("Form", "TMP")

    Call ConvertBinaryFileToCD(binFilePath, tmpCDFilePath, binFileSize)
    apiResult = MailAddMessageBodyComposite(docTmp.Handle, "$FileData", tmpCDFilePath)
    Kill tmpCDFilePath

    If apiResult <> 0 Then
        errDescr = GetAPIError(apiResult)
        Error APP_ERR_CODE, "Error importing CD file into document: " & errDescr
    End If

    ' Remove existing file resource to prevent DXL importing failure
    Set nc = db.CreateNoteCollection(False)
    Call nc.SelectAllDesignElements(True)
    nc.SelectionFormula = {$Flags = "~C4g" & @UpperCase($TITLE) = @UpperCase("} & resourceTitle &
{")}
    Call nc.BuildCollection()
    If nc.Count > 0 Then
        docDesignID = nc.GetFirstNoteID
        While docDesignID <> ""
            Set docDesignRsrc = db.GetDocumentByID(docDesignID)
            ' Get next ID before the current document will be removed
            docDesignID = nc.GetNextNoteId(docDesignID)

            Call docDesignRsrc.Remove(True)
        Wend
    End If

    ' Create an empty file resource, without an actual RT content
    dxlContent = {<?xml version='1.0' encoding='utf-8'?>
<note class='form'>
<item name='$Flags'><textlist><text>~C4g</text></textlist></item>
<item name='$FlagsExt'><textlist><text>w</text></textlist></item>
<item name='$TITLE'><textlist><text>} & resourceTitle & {</text></textlist></item>
<item name='$FileSize' sign='true'><numberlist><number>} & binFileSize &
{</number></numberlist></item>
<item name='$FileNames' sign='true'><textlist><text>} & resourceTitle & {</text></textlist></item>
</note>}

    Set stream = session.CreateStream
    Call stream.WriteText(dxlContent)

    Set dxlImporter = session.CreateDXLImporter(stream, db)
    dxlImporter.DesignImportOption = DXLIMPORTOPTION_REPLACE_ELSE_CREATE
    dxlImporter.InputValidationOption = VALIDATE_NEVER
    dxlImporter.ExitOnFirstFatalError = False
    Call dxlImporter.Process
    Call stream.Close
    If dxlImporter.ImportedNoteCount = 0 Then
        Error APP_ERR_CODE, "Could not create file resource"
    End If

    docDesignID = dxlImporter.GetFirstImportedNoteId
    Set docDesignRsrc = db.GetDocumentById(docDesignID)

    ' Copy RT item from the temporary document into the file resource
    Set rtItem = docTmp.GetFirstItem("$FileData")
    ' RT file became encrypted after creation by the C API method.
    ' It must be decrypted before copying into another document
    rtItem.IsEncrypted = False
    Call rtItem.CopyItemToDocument(docDesignRsrc, rtItem.Name)

    Call docDesignRsrc.Sign
    Call docDesignRsrc.Save(True, False)

    ' Do not save the temporary document
End Sub

Sub ConvertBinaryFileToCD(ByVal fileNameBin As String, ByVal fileNameCD As String, binaryFileSize
As Long)
```

```
    ' Convert any binary file to a file containing CD records ready to be imported into RT field of
a file resource

    Dim session As New NotesSession
    Dim streamIn As NotesStream, streamOut As NotesStream

    Set streamIn = session.CreateStream
    If Not streamIn.Open(fileNameBin, "Binary") Then
        Error APP_ERR_CODE, {Could not open source file "} & fileNameBin & {"}
    End If

    binaryFileSize = streamIn.Bytes

    Set streamOut = session.CreateStream
    If Not streamOut.Open(fileNameCD, "Binary") Then
        Error APP_ERR_CODE, {Could not create destination file "} & fileNameCD & {"}
    End If
    ' Clear the possible content if the file exists
    Call streamOut.Truncate

    Call WriteCDFileHeader(streamIn, streamOut, fileNameBin)
    While Not streamIn.IsEOS
        Call WriteCDFileSegment(streamIn, streamOut)
    Wend

    Call streamIn.Close
    Call streamOut.Close
End Sub

Sub WriteCDFileHeader(streamIn As NotesStream, streamOut As NotesStream, ByVal fileNameIn As
String)
    Dim filePrefix As Integer
    Dim fileHeader As CDFILEHEADER
    Dim fileExt As String
    Dim fileSize As Long
    Dim alignmentFiller As Byte
    Dim bufPos As Long
    Dim headerBuf() As Byte
    Dim isFillerRequired As Boolean

    fileSize = streamIn.Bytes
    fileExt = ExtractFileExt(fileNameIn)

    fileHeader.Header.Signature = SIG_CD_FILEHEADER

    fileHeader.Header.Length = 24 + Len(fileExt)
    If fileHeader.Header.Length Mod 2 > 0 Then
        fileHeader.Header.Length = fileHeader.Header.Length + 1
        alignmentFiller = &h00
        isFillerRequired = True
    Else
        isFillerRequired = False
    End If

    fileHeader.FileExtLen = Len(fileExt)
    fileHeader.FileDataSize = fileSize

    fileHeader.SegCount = fileSize \ CD_MAX_SEGMENT_SIZE
    If fileSize Mod CD_MAX_SEGMENT_SIZE > 0 Then
        fileHeader.SegCount = fileHeader.SegCount + 1
    End If

    fileHeader.Flags = &h0000
    fileHeader.Reserved = &h0000
    fileHeader.FileExt = fileExt

    ' For an unknown reason, the CD file exported during the testing contained
    ' the prefix &h0001 (DWORD) written in little-endian byte order (01 00).
    ' I did not find it in C API but added it for the compatibility
    ReDim headerBuf(0 To fileHeader.Header.Length - 1 + 2) As Byte
    bufPos = 0
    filePrefix = &h0001
    Call BufPutNumber(headerBuf, bufPos, filePrefix)

    Call BufPutNumber(headerBuf, bufPos, fileHeader.Header.Signature)
    Call BufPutNumber(headerBuf, bufPos, fileHeader.Header.Length)
    Call BufPutNumber(headerBuf, bufPos, fileHeader.FileExtLen)
```

```
    Call BufPutNumber(headerBuf, bufPos, fileHeader.FileDataSize)
    Call BufPutNumber(headerBuf, bufPos, fileHeader.SegCount)
    Call BufPutNumber(headerBuf, bufPos, fileHeader.Flags)
    Call BufPutNumber(headerBuf, bufPos, fileHeader.Reserved)
    Call BufPutString(headerBuf, bufPos, fileHeader.FileExt)

    If isFillerRequired Then
        Call BufPutNumber(headerBuf, bufPos, alignmentFiller)
    End If

    Call streamOut.Write(headerBuf)
    Erase headerBuf
End Sub

Sub WriteCDFileSegment(streamIn As NotesStream, streamOut As NotesStream)
    Dim fileSegmentHeader As CDFILESEGMENT
    Dim headerBuf(0 To 17) As Byte
    Dim bufPos As Long
    Dim fileData As Variant
    Dim fileDataLen As Integer
    Dim alignmentFiller(0 To 0) As Byte
    Dim isFillerRequired As Boolean

    fileData = streamIn.Read(CD_MAX_SEGMENT_SIZE)
    fileDataLen = UBound(fileData) + 1

    fileSegmentHeader.Header.Signature = SIG_CD_FILESEGMENT
    fileSegmentHeader.Header.Length = 18 + fileDataLen ' Total number of significant bytes in the
segment
    fileSegmentHeader.DataSize = fileDataLen ' Number of bytes in the binary file chunk

    ' Total size of binary data in the segment including filler
    fileSegmentHeader.SegSize = fileDataLen
    If fileDataLen Mod 2 > 0 Then
        fileSegmentHeader.SegSize = fileSegmentHeader.SegSize + 1 ' Total size of the segment with
a possible filler
        alignmentFiller(0) = &h00
        isFillerRequired = True
    Else
        isFillerRequired = False
    End If

    fileSegmentHeader.Flags = &h0000
    fileSegmentHeader.Reserved = &h0000

    bufPos = 0
    Call BufPutNumber(headerBuf, bufPos, fileSegmentHeader.Header.Signature)
    Call BufPutNumber(headerBuf, bufPos, fileSegmentHeader.Header.Length)

    Call BufPutNumber(headerBuf, bufPos, fileSegmentHeader.DataSize)
    Call BufPutNumber(headerBuf, bufPos, fileSegmentHeader.SegSize)
    Call BufPutNumber(headerBuf, bufPos, fileSegmentHeader.Flags)
    Call BufPutNumber(headerBuf, bufPos, fileSegmentHeader.Reserved)

    Call streamOut.Write(headerBuf)
    Erase headerBuf

    Call streamOut.Write(fileData)
    Erase fileData

    If isFillerRequired Then
        Call streamOut.Write(alignmentFiller)
    End If
End Sub

Sub BufPutString(byteBuf As Variant, bufPos As Long, vString As String)
    ' Put a string into the byte buffer starting with the current buffer position and increment the
buffer position after.
    ' Notices:
    ' - LMBCS characters are converted to ASCII codes;
    ' - the null terminator is not written
    Dim i As Integer
    Dim c As String

    If vString <> "" Then
        For i = 1 To Len(vString)
            c = Mid(vString, i, 1)
```

```
            byteBuf(bufPos) = Asc(c)
            bufPos = bufPos + 1
        Next
    End If
End Sub

Sub BufPutNumber(byteBuf As Variant, bufPos As Long, vNumber As Variant)
    ' Put a number (BYTE, WORD, DWORD) into the byte buffer in little-endian format (required for
CD RTF records)
    ' starting with the current buffer position and increment the buffer position after.
    ' Notices:
    ' - to simulate unsigned C language WORD and DWORD the LS Integer and Long are used
correspondingly;
    ' - parameter vNumber is specified without the ByVal keyword to keep its original size in bytes
    ' (otherwise it can be aligned to Long when copying to memory)
    ' -    numeric values of CD headers structures must be written to the CD file
    ' in the little-endian format (lowest byte is written first), e.g. &h01ABCDEF => EFCDAB01
    Dim i As Integer
    Dim lowestByte As Byte

    If Not IsEmpty(vNumber) Then ' Empty value can appear because of not existing zero filler in
the header
        For i = 1 To LenB(vNumber)
            lowestByte = vNumber And &hFF
            byteBuf(bufPos) = lowestByte
            bufPos = bufPos + 1
            ' Shift the number right
            ' (actually, the division properly considers the platform-specific byte order and does
not cause an overvlow)
            vNumber = vNumber \ &h100
        Next
    End If
End Sub

Function FileExists(ByVal fileName As String) As Boolean
    Dim fn As String

    On Error GoTo ExitHandle

    FileExists = False
    If fileName <> "" Then
        fn = Dir$(fileName)
        If fn <> "" Then
            FileExists = True
        End If
    End If

ExitHandle:
    Exit Function
End Function

Function ExtractFileName(ByVal filePath As String) As String
    Dim rst As String

    ' Cross-platform solution
    filePath = Replace(filePath, "\", "/", 1, 100, NoCaseNoPitch)

    If InStr(filePath, "/") > 0 Then
        rst = StrRightBack(filePath, "/", NoCaseNoPitch)
    ElseIf InStr(filePath, ":") > 0 Then
        rst = StrRightBack(filePath, ":", NoCaseNoPitch)
    Else
        rst = filePath
    End If
    ExtractFileName = rst
End Function

Function ExtractFileNameRoot(ByVal filePath As String) As String
    Dim rst As String

    rst = ExtractFileName(filePath)
    If InStr(rst, ".") > 0 Then
        rst = StrLeft(rst, ".")
    End If

    ExtractFileNameRoot = rst
End Function
```

```
Function SeparateDir(ByVal strDirName As String) As String
    Dim dirSeparator As String
    Dim jSession As JavaSession
    Dim jClass As JavaClass
    Dim jProperty As JavaProperty

    If IsWindowsPlatform() Then
        dirSeparator = "\"
    Else
        Set jSession = New JavaSession
        Set jClass = jSession.GetClass("java.io.File")
        Set jProperty = jClass.GetProperty("separator")
        dirSeparator = jProperty.GetValue()
    End If

    If Right(strDirName, 1) <> dirSeparator Then
        SeparateDir = strDirName & dirSeparator
    Else
        SeparateDir = strDirName
    End If
End Function

Function ExtractFileExt(ByVal fileName As String) As String
    ExtractFileExt = StrRightBack(fileName, ".", NoCaseNoPitch)
End Function


Function GetTempFilePath(ByVal tmpDir As String, ByVal filePath As String) As String
    ' Generate a temporary file path in the specified folder using the specified file path as a
pattern.
    ' Notice that this method generates an unique file name (it checks if a file with a generated
name already exists and if so - generates another name).
    ' Resulting file path will be generated as:
    '    TEMP_DIR\FILE_NAME-RANDOM_PART.FILE_EXT
    ' where
    '    TEMP_DIR = tmpDir
    '    FILE_NAME = file name without extention extracted from filePath
    '    RANDOM_PART = 8-character random string
    '    FILE_EXT = file extention extracted from filePath
    ' If tmpDir is empty then OS temporary directory will be used automatically:
    '    OS_TEMP_DIR\FILE_NAME-RANDOM_PART.FILE_EXT
    ' Parameter filePath can include a directory which will be ignored.
    ' If no file separators in the filePath (":", "/", "\", ".") then filePath will be interpreted
as a file extention:
    '    TEMP_DIR\RANDOM_PART.FILE_EXT
    ' If filePath is empty then only random part in the temporary directory will be returned:
    '    TEMP_DIR\RANDOM_PART
    ' E.g.:
    ' GetTempFilePath("C:\TMP", "D:\Documents\File.doc") = "C:\TMP\File-28702758.doc"
    ' GetTempFilePath("", "jpg") = "C:\Windows\Temp\02389582.jpg"
    ' GetTempFilePath("", "") = "C:\Windows\Temp\67230296"

    Dim cntr As Integer
    Dim fileExt As String
    Dim fileNameRoot As String
    Dim rndPart As String
    Dim rst As String

    If Trim(tmpDir) = "" Then
        tmpDir = GetOSTempDir()
    End If
    tmpDir = SeparateDir(tmpDir)

    ' Real file names can contain leading and trailing spaces which are unuseful in a temporary
file name
    fileNameRoot = Trim(ExtractFileNameRoot(filePath))
    fileExt = Trim(ExtractFileExt(filePath))

    ' Notice that Randomize with no numExpr seeds the random number generator with the return value
from Timer.
    ' It is more reliable than to use Randomize with a particular seed. But generation of a random
sequence is called
    ' again within a short time interval (less than Timer quantization) then the resulting sequence
will be the same.
    ' That is why an additional loop with testing for existence of a file is required
    Randomize
```

```
    Do
        rst = tmpDir

        rndPart = ""
        For cntr = 1 To 8
            rndPart = rndPart & Mid(LTrim(CStr(CInt(Rnd * 10))), 1, 1)
        Next
        rndPart = Trim(rndPart)

        If fileNameRoot <> "" Then
            rst = rst & fileNameRoot & "-"
        End If

        rst = rst & rndPart

        If fileExt <> "" Then
            rst = rst & "." & fileExt
        End If
    Loop While FileExists(rst)

    GetTempFilePath = rst
End Function

Function GetOSTempDir() As String
    ' Return the OS temporary directory
    Dim tempPathBuf As String*4096
    Dim errNo As Long
    Dim jSession As JavaSession
    Dim jClass As JavaClass
    Dim jMethod As JavaMethod

    If IsWindowsPlatform() Then
        errNo = GetTempPath(4096, tempPathBuf)
        GetOSTempDir = ClngStringToStr(tempPathBuf)
    Else
        Set jSession = New JavaSession
        Set jClass = jSession.GetClass("java/lang/System")
        Set jMethod = jClass.GetMethod("getProperty", "(Ljava/lang/String;)Ljava/lang/String;")
        GetOSTempDir = jMethod.Invoke( , "java.io.tmpdir")
    End If
End Function

Function ClngStringToStr(ByVal C_Str  As String) As String
    ' Convert a null-terminated (C language) string to LS string
    Dim posNull As Long
    Dim rst As String

    posNull = InStr(C_Str, Chr$(0))
    If posNull = 0 Then
        ' String is not a C language string - return whole string
        rst = C_Str
    Else
        rst = Left(C_Str, posNull -1)
    End If

    ClngStringToStr = rst
End Function

Public Function GetAPIError(errorCode As Integer) As String
    Const ERR_MASK = &h3fff
    Const PKG_MASK = &h3f00
    Const ERRNUM_MASK = &h00ff

    Dim errorString As String*256
    Dim returnErrorString As String
    Dim resultStringLength As Long
    Dim errorCodeTranslated As Integer

    returnErrorString = ""

    ' Mask off the top 2 bits of the errorCode that was returned; this is
    ' what the ERR macro in the API does
    errorCodeTranslated = (errorCode And ERR_MASK)

    ' Get the error code translation using the OSLoadString API function
    resultStringLength = OSLoadString(0, errorCodeTranslated, errorString, Len(errorString) - 1)
```

```
    ' Strip off the null-termination on the string before you return it
    If (InStr(errorString, Chr(0)) > 0) Then
        returnErrorString = Left$(errorString, InStr(errorString, Chr(0)) - 1)
    Else
        returnErrorString = errorString
    End If

    GetAPIError = returnErrorString
End Function

Function IsWindowsPlatform As Boolean
    Dim session As New NotesSession
    If InStr(1, session.Platform, "Windows", NoCaseNoPitch) > 0 Then
        IsWindowsPlatform = True
    Else
        IsWindowsPlatform = False
    End If
End Function
```