Notes /Domino 6 Script Classes

Lotus. software

Notes Session

Properties
notesDatabase:Array<notesSession.AddressBooks
commonUserName:=notesSession.CommonUserName
Date:=notesSession.CurrentDate
notesAgent:=notesSession.CurrentAgent
notesDatabase:=notesSession.CurrentDatabase
notesDocument:=notesSession.DocumentContext
nameOf=notesSession.EffectiveUserName
platform:=notesSession.GetEnvironmentString(NameList
{LIB} :=notesSession.IsOnServer()
notesInternational=notesSession.International
flag=notesSession.IsUnrestricted
code %=notesSession.LastExitStatus
{LIB} :=notesSession.LoadExtension(
{LIB} :=notesSession.NotesBuildVersion
{LIB} :=notesSession.NotesURL
version%=notesSession.NotesVersion
pig[Directory]%=notesSession.PigDirectory$Path
platform$=notesSession.Platform
notesDocument=notesSession.SavedData
name$=notesSession.ServerName
{LIB} for nerdsDatabase=notesSession.URLDatabase
nameOf=notesSession.UserName
{LIB} for nerdsName=notesSession.UserNameList
{LIB} for nerdsName=notesSession.UserNameObject

Methods
Set notesAdministrationProcess=notesSession.CreateAdministrationProcess(server$)
Set notesColorObject=notesSession.CreateColorObject()
Set notesDOMParser=notesSession.CreateDOMParser(input, output)
Set notesDXLExporter=notesSession.CreateDXLExporter(input$,output$)
Set notesDXLImporter=notesSession.CreateDXLImporter(input$,output$)
Set notesNewDateRange=notesSession.CreateDateRange()
Set notesDateTime=notesSession.CreateDateTime(dateTime$)
Set notesLog=notesSession.CreateLog(programName$)
Set notesName=notesSession.CreateName(username$,[unpack]$)
Set notesNewsletter=notesSession.CreateNewsletter(notesDocumentCollection)
{LIB} for nerdsRegistration=notesSession.CreateRegistration()
{LIB} for nerdsRichTextParagraphStyle=notesSession.CreateRichTextParagraphStyle()
Set notesRichTextStyle=notesSession.CreateRichTextStyle()
{LIB} for nerdsTimer=notesSession.CreateTimer(
Set notesSAXParser=notesSession.CreateSAXParser([input [, [output]])
Set notesTimer=notesSession.CreateTimer()
{LIB} for nerdsXSLTransformer=notesSession.CreateXSLTransformer([input$, [output$, [styleSheet$]]])
{LIB} for variant=notesSession.EvaluateVariant(&,&)
Set notesDateRange=notesSession.FreeTimeSearch(window_duration, name$, flag$)
Set notesDateTime=notesSession.GetLastExecutionDate([notesDateTime$])
Set notesDXLDirectory=notesSession.GetDXLDirectory(serverName$)
notesDocument = notesSession.GetUserPolicySettings(serverS, name$,[type %],[explicitPolicy$],[reserved$])
hashedPassword$=notesSession.HashPassword(password$)
value$=notesSession.GetEnvironmentString(name$,[system])
notesDocument=notesSession.GetUserSetupProfile(server$,[system])
{LIB} Call notesSession.InitializeUsingNotesUserName(name$[,password$])
flag=notesSession.VerifyPassword(password$, hashedPassword$)
{LIB} for nerdsVersion=notesSession.NotesVersion()
consoleReturn$=notesSession.SendConsoleCommand(serverName$,consoleCommand$)
Call notesSession.SetEnvironmentVar(name$,value$[,insystem])
Call notesSession.UpdateProcessedDocuments()

Notes UI Workspace

Properties
dateV=notesUIWorkspace.CurrentCalendarDateTime
Set notesUIDatabase=notesUIWorkspace.CurrentDatabase
Set notesUIDocument=notesUIWorkspace.CurrentDocument
Set notesView=notesUIWorkspace.CurrentView

Methods
Call notesUIWorkspace.AddDatabase(server$,filename$)
Call notesUIWorkspace.Check(Anyway)
Set notesUIDatabase=notesUIWorkspace.ComposeDocument(server$,file$,form$, window%=0,flag$=0)
Set notesUIDocument=notesUIWorkspace.ComposeNamedDocument(server$,file$,form$,[,autoOpenWindow$][,autoFields$][,autoUpdate$][,readOnly$][,title$],notesDocument,sizeToTable$,useOnCancel$,doCancel$,fullmenus)
Set notesUIDocument=notesUIWorkspace.EditDocument([editMode$],notesDocument[,notesDocumentReadOnly$],[,docLaunch=AutoNo$],returnNotesUIDocument$)
Call notesUIWorkspace.EditProfile(profileName$,userName$)
flag=notesUIWorkspace.EnableMenu(enable$)
{ } =notesUIWorkspace.FieldGet(fieldname$[,index])
Set notesUIDatabase = notesUIWorkspace.GetCurrentDatabase
Set notesUIDocument=notesUIWorkspace.GetFirstDocument()
Call notesUIWorkspace.OpenDatabase(server$,file$,view$,key$,newInstance,temp)
{ } =notesUIWorkspace.OpenFrameSet(frameset$)
Call notesUIWorkspace.OpenView(server$,file$,view$,key$,newInstance,temp)
Call notesUIWorkspace.PasteFromClipboard()
Call notesUIWorkspace.PostDataToServer(URL$,urlText$,key$,newInstance,temp)
Call notesUIWorkspace.PostDocumentSave()
Call notesUIWorkspace.ReloadWindow()
{ } =notesUIWorkspace.SaveFileDialog(multiple$,selectionLimit$,filterAll$,initialDirectory$[,initialFile$])
Call notesUIWorkspace.SetCurrentLocation(locationUrl$)
Call notesUIWorkspace.SetTargetFrame(frame$)
Call notesUIWorkspace.URLOpen(URLtoRead %,urlList %,charset%,webusername$,webpassword$,proxywebusername$,proxywebpassword$,knownfileviewer)
Call notesUIWorkspace.ViewRefreshAll(dontRefreshMe)
Call notesUIWorkspace.ViewRefreshFields()
Call notesUIWorkspace.ViewRefresh

Events
PostDropToArchive(Source As NotesUIDatabase, EntryName, Continue As Variant)
QueryDropToArchive(Source As NotesUIDatabase, EntryName, Continue As Variant)

Notes UI Document

Properties
flag=notesUIDocument.AutoReload
fieldName$=notesUIDocument.CurrentField
flag=notesUIDocument.DialogBoxCanceled
Set notesDocument=notesUIDocument.Document
R/W flag=notesUIDocument.EditMode
R/W flag=notesUIDocument.FieldHelp
R/W flag=notesUIDocument.HideActions
R/W flag=notesUIDocument.HorizScrollBar
flag=notesUIDocument.InPreviewPane
flag=notesUIDocument.IsNewDoc
Set notesUIDocument=notesUIDocument.NewDocLink
R/W flag=notesUIDocument.PreviewDocLink
R/W flag=notesUIDocument.PrintSaveState
nSleft=notesUIDocument.Window.Left
Methods
Call notesUIDocument.CategorizeEntry(category$)
Call notesUIDocument.Clear
Call notesUIDocument.Close
Call notesUIDocument.CollapseAllSections
Call notesUIDocument.Copy
Set handle%=notesUIDocument.CreateObject(name$,type$,filePath$[])
Call notesUIDocument.Cut
Call notesUIDocument.DeleteDocument
Call notesUIDocument.DeselectAll
Call notesUIDocument.ExpandAllSections
Call notesUIDocument.FieldAppendText(fieldName$,text$)
Call notesUIDocument.FieldClearText(fieldName$)
Call notesUIDocument.FieldGetText(fieldName$,[,keepVerbatim$])
text%=handle%.notesUIDocument.FieldGetText(fieldName$,keepVerbatim$)
Call notesUIDocument.FieldSetText(fieldName$,newValue$)
Call notesUIDocument.FindNextText(findingFromPopupHere$,replaceHere$,replaceAll$,ignoreCase$,wholeWord$,backwards$,wrapAround$,searchSelectionOnly$,searchFormFields$,regExpressions$)
{ } flag=notesUIDocument.FindOnPage(findingFromPopupHere$,replaceHere$,replaceAll$,ignoreCase$,wholeWord$,backwards$,wrapAround$,searchSelectionOnly$,searchFormFields$,regExpressions$,removeAllPeopleItems$,startDatabaseLoadDatabase$)
{ } Call notesUIDocument.FindString(text$,wholeWord$[,backwards$,][,useSelection$,][,startDatabaseLoadDatabase$])
{ } Call notesUIDocument.FindStringNextOccurrence(text$[,caseSensitive$,][,wholeWord$,][,backwards$,][,wrapAround$,][,ignoreDiacritics$])
Call notesUIDocument.Forward
Set handle%=notesUIDocument.GetObject(name$)
Set notesDocument=notesUIDocument.GetSelectedDocument()
Call notesUIDocument.GotoField(fieldName$)
Call notesUIDocument.Paste
Call notesUIDocument.Print(numCopies% [,fromPage %] [,toPage %] [,draft])
Call notesUIDocument.PrintWithDialog(allPages$,fromPage%,toPage%)
Call notesUIDocument.RefreshHideFormulas
Call notesUIDocument.Reload()
Call notesUIDocument.Save()
Call notesUIDocument.SaveNewVersion
Call notesUIDocument.SelectAll
Call notesUIDocument.Send()
{ } Call notesUIDocument.SpellCheck()
{ }
OnHideQuery(as Notes.UIDocument)
OnLoad(as Notes.UIDocument)
{ } OnSubmit(Source as Notes.UIDocument, Continue as Variant)
{ } OnUnload(Source as Notes.UIDocument, Continue as Variant)
PostModeChange(Source as Notes.UIDocument)
PostOpen(Source as Notes.UIDocument)
PostSave(Source as Notes.UIDocument)
QueryClose(Source as Notes.UIDocument, Continue as Variant)
QueryModeChange(Source as Notes.UIDocument, Continue as Variant)
QueryOpen(Source as Notes.UIDocument,Mode as Integer,Continue as Variant,Continue as Variant)
{ } QuerySave(Source as Notes.UIDocument,Continue as Variant)
{ } QuerySend(Source as Notes.UIDocument,Continue as Variant)

NotesUIDatabase

Properties
Set notesDatabase=notesUIDatabase.Database
Set documentCollection=notesUIDatabase.Documents
Methods
Call notesUIDatabase.Close
{ } Call notesUIDatabase.OpenNavigator(navigatorName$,fullWindow%)
Call notesUIDatabase.OpenView(viewName$,key$,newInstance$,[,replaceView])
{ }
PostDocumentDelete(Source as Notesuidatabase)
{ } PostDropToArchive(Source As Notesuidatabase,EntryName,Continue As Variant)
PostDropToArchive(Source As Notesuidatabase,EntryName,Continue As Variant)
PostOpen(Source As Notesuidatabase)
QueryDocumentClose(Source As Notesuidatabase,Continue As Variant)
QueryDocumentDelete(Source As Notesuidatabase,Continue As Variant)
QueryDropToArchive(Source As Notesuidatabase,EntryName,Continue As Variant)
QueryDocumentUndelete(Source As Notesuidatabase,Continue As Variant)
{ } QueryDropToQuery(Source As Notesuidatabase,EntryName,Continue As Variant)

Notes Database

Properties
Set notesACL=notesDatabase.ACL
string$array=notesDatabase.ACLActivityLog
nonAdminSize=notesDatabase.AllNonAdminDocuments
R/W category$=notesDatabase.Categories
date%=notesDatabase.Created
R/W flag=notesDatabase.CurrentAccessLevel
R/W flag=notesDatabase.DefaultDbTemplate
name$=notesDatabase.DesignTemplateName
flag$=notesDatabase.FileFormat
R/W frequency%=notesDatabase.FTIndexFrequency
Set notesDocument=notesDatabase.GetDocumentByUNID(universalID$,formatNotesView)
Set handle=notesDatabase.GetForm(name$)
{ } R/W flag=notesDatabase.FolderReferencesEnabled
nameOfNotesTemplate$=notesDatabase.GetNotesTemplate()
{LIB} nerdsDatabase.HttpURL
flag=notesDatabase.IsAnyReplicaFailing
flag=notesDatabase.IsDefaultDbSearch
flag=notesDatabase.IsOpen
flag=notesDatabase.IsPrivateAddressBook
flag=notesDatabase.IsPublicAddressBook
R/W flag=notesDatabase.LastModified
flag=notesDatabase.LicenseLogging
flag=notesDatabase.LocalReplication
flag=notesDatabase.LocalSecurityProtection
flag=notesDatabase.LogActions
flag=notesDatabase.LogAgentRun
flag=notesDatabase.LogReplicationEvents
flag=notesDatabase.LogSecurityEvents
flag=notesDatabase.LogToNotesFile
flag=notesDatabase.LogToSQL
flag=notesDatabase.LogToSyslog
flag=notesDatabase.LogToTrapDiscard
flag=notesDatabase.LogToTrapNotify
flag=notesDatabase.MarkForDelete
Set notesDatabase=notesDatabase.NewCopy(server$,title$,createFullAccessACL)
Set notesDatabase=notesDatabase.NewReplica(server$,title$,createFullAccessACL)
Set notesDocument=notesDatabase.OpenAgent(agentName$)
Set notesDocument=notesDatabase.OpenDocumentByID(noteID$)
Set notesDocument=notesDatabase.OpenDocumentByReplicaID(replicaID$)
Set notesDocument=notesDatabase.OpenDocumentByURL(URL$,[,reload][,urlHas][,charset$],webusername$]
[,webpassword$][,proxywebusername$][,proxywebpassword$][,returnmimeheader$])
Set dataFrame=notesDatabase.GetFrameset(name$)
{ } Save notesOutline=notesDatabase.GetNotesOutline(outlineName$)
Set notesDocumentCollection=notesDatabase.GetProfileDocCollection(profileName$)
Set handle%=notesDatabase.GetURLHeader(URL$,headerName$,[,webpassword$])
{ } Proxywebusername$[,Proxywebpassword$])
Set notesView=notesDatabase.GetView(viewName$)
Call notesDatabase.GrantAccess(name$,level %)
Call notesDatabase.Open()
Call notesDatabase.OpenIfModified()
Call notesDatabase.OpenMailDb()
Call notesDatabase.OpenURLDb(URL$,[,timeout%])
Call notesDatabase.OpenWeb()
Call notesDatabase.PasteFromClipboard()
Call notesDatabase.RemoveACL()
level %=notesDatabase.QueryAccess(name$)
Call notesDatabase.Remove
flag=notesDatabase.ReplicaInvolves(name$)
Call notesDatabase.ResetLastModified()
Call notesDatabase.RevokeACL()
Call notesDatabase.SetUpdatedSince(name$ %,flag)
Call notesDatabase.Sign([documentType %][,existingSig$=yES,][,nameSelf$,nameSelfcNotED])
Set notesDocumentCollection=notesDatabase.SearchAllUnreadUserViews(name$,cutoffDate$,deleteMatches$)
Set notesDocumentCollection=notesDatabase.SearchFreeTimeSlots(dateTimeRange$,duration %,[,optionalItems$])
Set notesDocumentCollection = notesDatabase.UnprocessedDocuments([%,[,interruptions %][,interruptions %][,search$]])
Set notesDocumentCollection = notesDatabase.UnprocessedFTSearch(dateRange$,duration %,[,interruptions %][,interruptions %][,search$])
Set notesDocumentCollection = notesDatabase.UnprocessedSearch(formula$,[,timeDateTime,maxDocs %])
Call notesDatabase.UpdateFTIndex(createFlag)

NotesNoteCollection

Properties
count%=notesNoteCollection.Count
flag=notesNoteCollection.DeleteAllInDbTime
Set notesDatabase=notesNoteCollection.Parent
R/W flag=notesNoteCollection.SelectACL
R/W flag=notesNoteCollection.SelectActions
R/W flag=notesNoteCollection.SelectAgents
R/W flag=notesNoteCollection.SelectDatabaseScript
R/W flag=notesNoteCollection.SelectDataConnections
R/W flag=notesNoteCollection.SelectDbProperties
R/W flag=notesNoteCollection.SelectFolders
R/W flag=notesNoteCollection.SelectFramesets
R/W flag=notesNoteCollection.SelectHelpAbout
R/W flag=notesNoteCollection.SelectHelpUsing
R/W flag=notesNoteCollection.SelectHistory
R/W flag=notesNoteCollection.SelectIcon
R/W flag=notesNoteCollection.SelectImageResources
R/W flag=notesNoteCollection.SelectJavaResources
R/W flag=notesNoteCollection.SelectMiscCodeElements
R/W flag=notesNoteCollection.SelectMiscFormatElements
R/W flag=notesNoteCollection.SelectMiscTemplateElements
R/W flag=notesNoteCollection.SelectNavigators
R/W flag=notesNoteCollection.SelectOutlines
R/W flag=notesNoteCollection.SelectPages
R/W flag=notesNoteCollection.SelectProfiles
R/W flag=notesNoteCollection.SelectScriptLibraries
R/W flag=notesNoteCollection.SelectSharedFields
R/W flag=notesNoteCollection.SelectSharedResources
R/W flag=notesNoteCollection.SelectSubforms
R/W flag=notesNoteCollection.SelectViews
R/W lastBuildTime = notesNoteCollection.SinceTime

Methods
Call notesNoteCollection.AddLikeNote(type$[,title])
Call notesNoteCollection.BuildCollection()
Call notesNoteCollection.ClearCollection
Call notesNoteCollection.DeleteAll(force$)
Call notesNoteCollection.GetFirstNoteID(retNoteID$)
Call notesNoteCollection.IntersectSetwise(operation$[,otherCollection])
Call notesNoteCollection.SelectAllActionNotes(selectorValue)
Call notesNoteCollection.SelectAllAgentNotes(selectorValue)
Call notesNoteCollection.SelectAllDataConnectionNotes(selectorValue)
Call notesNoteCollection.SelectAllDesignElements(selectorValue)
Call notesNoteCollection.SelectAllFormulaNotes(selectorValue)
Call notesNoteCollection.SelectAllHelpAboutDocuments(selectorValue)
Call notesNoteCollection.SelectAllNotes(selectorValue)

Notes Form

Properties
stringArray=notesForm.Aliases
stringArray=notesForm.Fields
flag=notesForm.FormUsesFrameSets
flag=notesForm.IsAutoFrame
IncHiddens=notesForm.LockHiddens
string=notesForm.Name
name$=notesForm.Name$[0]
{ LIB} Set notesDatabase=notesForm.Parent
R/W flag=notesForm.ProtectReaders
R/W flag=notesForm.StoreForm
R/W formula$=notesForm.Readers

Methods
{ } type %=notesForm.GetFieldType(name)
flag=notesForm.Lock[name[,previousvalue]]
Call notesForm.Remove
Call notesForm.Unlock()

Notes View

Properties
aliases=notesView.Aliases (R/W)
Set notesDatabase=notesView.Database
flag=notesView.HasChildren
Call notesView.IsAncestorOf(notesView)
Call notesView.IsDescendantOf(notesView)
Set notesDocument=notesView.LastDocument
Set notesDocument=notesView.NextDocument(notesDocument)
Set notesDocument=notesView.NextSibling(notesDocument)
Set notesDocument=notesView.PrevDocument(notesDocument)
Set notesDocument=notesView.PrevSibling(notesDocument)
Set notesDocument=notesView.ParentDocument
Set notesDocumentCollection=notesView.AllEntries
Call notesView.Clear
Call notesView.Create
{ } Set notesViewColumn=notesView.CopyColumn(sourceColumn[,destinationIndex$])
{ } Set notesViewColumn=notesView.CreateColumn([position$,][,columnName$][,formula$])
Call notesView.CreateColumns(fromDocument)
Call notesView.CreateViewNavigator(navigatorName$,fullWindow$)
Call notesView.CreateViewNavigatorFromCollection(documentCollection,navigatorName$,fullWindow$)
Call notesView.CreateViewNavigatorFromDocument(navigatorName$,startDocument,fullWindow$)
Call notesView.CreateViewNavigatorFromDocumentKey(navigatorName$,keyDocument$,fullWindow$)
Call notesView.CreateViewNavigatorFromView(navigatorName$,sourceView,fullWindow$)
Call notesView.FTSearch(query$,maxDocs %)
Call notesView.GetAllDocumentsByKey(keyArray,exactMatch %)
Call notesView.GetChild(document)
Call notesView.GetDocumentByKey(keyArray,exactMatch %)
Call notesView.GetFirstDocument()
Call notesView.GetLastDocument()
Call notesView.GetNextDocument(document)
Call notesView.GetNthDocument(index%)
Call notesView.GetPrevDocument(document)
Call notesView.GetPrevSibling(document)
Call notesView.Refresh()
Call notesView.Remove
Call notesView.RemoveColumn(columnNumber%|index$)
Call notesView.Update()

Notes Document Collection

Properties
count%=notesDocumentCollection.Count
flag=notesDocumentCollection.IsSorted
Set notesDatabase=notesDocumentCollection.Parent
query$=notesDocumentCollection.Query
Methods
{ } Call notesDocumentCollection.Add(documentOrUniversalID,[,beforeDocument])
{ } Call notesDocumentCollection.DeleteAll(markDocumentsUnread$)
Call notesDocumentCollection.FTSearch(query$,maxDocs %)
{ } Set newDocument=notesDocumentCollection.GetDocument(document)
Set notesDocument=notesDocumentCollection.GetFirstDocument()
Set notesDocument=notesDocumentCollection.GetLastDocument()
Set notesDocument=notesDocumentCollection.GetNextDocument(notesDocument)
Set notesDocument=notesDocumentCollection.GetNthDocument(index%)
Set notesDocument=notesDocumentCollection.GetPrevDocument(notesDocument)
Call notesDocumentCollection.PutAllInFolder(folderName$,createIndex$)
Call notesDocumentCollection.RemoveAll(force)
Call notesDocumentCollection.RemoveFromFolder(folderName$)
Call notesDocumentCollection.StampAll(itemname$,value)
Call notesDocumentCollection.UpdateAll()

Notes Navigator

Properties
flag=notesNavigator.CacheSize
Set notesDatabase=notesNavigator.Database
R/W flag=notesNavigator.MacroFormula
Set notesView=notesNavigator.ParentView
Methods
Set notesObject=notesNavigator.GetChildAtXY(notesView,x%,y%)
Set notesView=notesNavigator.GetChildView(notesView)
Set notesEntry=notesNavigator.GetClosestEntry(x%,y%)
Set notesEntry=notesNavigator.GetFirstEntry()
Set notesEntry=notesNavigator.GetLastEntry()
Set notesEntry=notesNavigator.GetNextEntry(notesEntry)
Set notesEntry=notesNavigator.GetNextSibling(notesEntry)
Set notesEntry=notesNavigator.GetPrevEntry(notesEntry)
Set notesEntry=notesNavigator.GetPrevSibling(notesEntry)
{LIB} Call notesViewNavigator.GotoEntry(obj%|document,)
{LIB} Call notesViewNavigator.GotoFirst()
{LIB} Call notesViewNavigator.GotoFirstDocument()
{LIB} Call notesViewNavigator.GotoLast()
{LIB} Call notesViewNavigator.GotoLastDocument()
{LIB} Call notesViewNavigator.GotoNext(notesViewEntry)
{LIB} Call notesViewNavigator.GotoNextDocument(notesViewEntry)
{LIB} Call notesViewNavigator.GotoNextSibling(notesViewEntry)
{LIB} Call notesViewNavigator.GotoPos(fractionPosition)
{LIB} Call notesViewNavigator.GotoPrev(notesViewEntry)
{LIB} Call notesViewNavigator.GotoPrevCategory(notesViewEntry)
{LIB} Call notesViewNavigator.GotoPrevDocument(notesViewEntry)
{LIB} Call notesViewNavigator.GotoPrevSibling(notesViewEntry)

Notes UI View

Properties
dateV=notesUIView.CalendarDateTime
{ } dateV=notesUIView.CalendarDateTimeEnd
{ } flag=notesUIView.ChooserIsActive
{ } nameOrNumber$=notesUIView.CurrentField
nomenDocument Collection=notesUIView.Documents
{ } view%=notesUIView.FTSearchScore
nomenView=notesUIView.View
{ } view%=notesUIView.ViewFormula
view%=notesUIView.ViewName

Methods
{ } Call notesUIView.Close
Call notesUIView.DeselectAll
{ } Call notesUIView.PrintFrameCount(, %[,fromPage %][,toPage %][,draft][,pageSeparator %][,keepOverwritesIE],dateHeaderIE[,dateHeaderIE])
Events
{ } Sub OnDragDrop(Source As NotesUIView,ItemCount%,type As Integer,CutPropName As Variant,ColumnValue As Variant,Continue As Variant)
PostDragDrop(Source As NotesUIView)
{ } PostEntryResize(Source As NotesView)
PostOpen(Source As NotesView)
PostPaste(Source As NotesView)
QueryAddToFolder(Source As NotesView,Target As Variant,Continue As Variant)
QueryClose(Source As NotesView,Continue As Variant)
QueryDragDrop(Source As NotesView,Continue As Variant)
QueryEntryResize(Source As NotesView,Continue As Variant)
QueryOpen(Source As NotesView,Continue As Variant)
QueryOpenDocument(Source As NotesView,Continue As Variant)
QueryPaste(Source As NotesView,Continue As Variant)
QueryRecalc(Source As NotesView,Continue As Variant)
RegionDoubleClick(Source As Notesview)

Notes View Column

Properties
alignment=notesViewColumn.Alignment (R/W)
{ } Set notesDocumentCollection=notesViewColumn.AllEntries
{LIB} for nerdsViewColumn.AutoUpdate
{ } color %=notesViewColumn.BackgroundColor (R/W)
{ } Count %=notesViewColumn.Count
{LIB} for nerdsViewColumn.DateTimeFormat
{LIB} for nerdsViewColumn.IsHidden
{ } Set notesViewColumn=notesViewColumn.NextColumn
{ } Set notesViewColumn=notesViewColumn.PrevColumn
{ } position%=notesViewColumn.Position
{LIB} for nerdsViewColumn.ResizeToFitContents
{ } width%=notesViewColumn.Width (R/W)

Methods
Call notesViewColumn.AddFormula(formula$)
Call notesViewColumn.Remove
Call notesViewColumn.SetFormula(formula$)

Notes Timer

Properties
R/W string=notesTimer.Comment
R/W flag=notesTimer.Enabled
R/W interval %=notesTimer.Interval
A
Alarm(Source as NotesTimer)

Notes Date Time

Properties
4.6 date$=notesDateTime.DateOnly
flag=notesDateTime.IsDST
flag=notesDateTime.IsValidDate
4.6 time$=notesDateTime.LocalTime
gmt%=notesDateTime.LSAGMTime
R/W lsZone%=notesDateTime.LSAZone
{LIB} for nerdsDateTime.TimeZone.Parent
4.6 time$=notesDateTime.TimeOnly
zone %=notesDateTime.TimeZone
zone$=notesDateTime.TZName

Methods
Call notesDateTime.Adjust(Year)% [,preserveLocalTime])
Call notesDateTime.AdjustDay(numberOfDays) %[,preserveLocalTime])
Call notesDateTime.AdjustMinute(n %[,preserveLocalTime])
Call notesDateTime.AdjustMonth(n %[,preserveLocalTime])
Call notesDateTime.AdjustSecond(n %[,preserveLocalTime])
Call notesDateTime.AdjustYear(n %[,preserveLocalTime])
Call notesDateTime.ConvertToZone(zone,dst)
Call notesDateTime.SetGMTTime(t)
Call notesDateTime.SetAsyTime
Call notesDateTime.SetNow
diff#=notesDateTime.TimeDifference(notesDateTime)
{ } diff#=notesDateTime.TimeDifferenceDouble(notesDateTime)

Notes International

Properties
string=notesInternational.AMString
string=notesInternational.CurrencyDigits
string=notesInternational.CurrencySymbol
flag=notesInternational.DST
string=notesInternational.DecimalSep
string=notesInternational.DateSep$Separator
flag=notesInternational.IsCurrencyLdSp
flag=notesInternational.IsCurrencyZero
flag=notesInternational.IsDateDMY
flag=notesInternational.IsDateMDY
flag=notesInternational.IsDateYMD
flag=notesInternational.IsDST
flag=notesInternational.IsTime24Hour
{LIB} for nerdsDomain=notesInternational.Parent
string=notesInternational.PMString
string=notesInternational.StringSep$Separator
string=notesInternational.TimeSep
zone %=notesInternational.TimeZone
string=notesInternational.Today
string=notesInternational.Tomorrow
string=notesInternational.Yesterday

Notes Log

Properties
R/W flag=notesLog.LogActions
R/W flag=notesLog.LogErrors
R/W flag=notesLog.LogMailRoutingEvents
errors %=notesLog.NumErrors
R/W flag=notesLog.OverwriteFile
{LIB} for nerdsServer=notesLog.Parent
R/W programName$=notesLog.ProgramName
Call notesLog.Close
Call notesLog.LogAction(description$)
Call notesLog.LogError(code %,description$)
Call notesLog.LogEvent(eventsgrp$,queuenum$,type %,severity %)
Call notesLog.OpenAgentLog
Call notesLog.OpenFileLog(path$)
Call notesLog.OpenMailLog(recipients$,subject$)
Call notesLog.OpenNotesLog(servers$,dbPath$)

Notes Name

Properties
R/W flag=notesName.Abbreviated
{ } string$=notesName.Add(A$)
{ } string$=notesName.Add(ADCommon$)
{ } string$=notesName.Add(ADCommon2
{ } string$=notesName.Add(ADCommon3
{ } string$=notesName.Add(ADCommPart
{ } string$=notesName.Add(Surname
string$=notesName.ADMD
string$=notesName.Canonical
string$=notesName.Common
string$=notesName.Country
string$=notesName.Generation
string$=notesName.Given
string$=notesName.Initials
flag=notesName.IsHierarchical
string$=notesName.Keyword
{LIB} for nerdsName.Language
flag=notesName.Organization
string$=notesName.Org[Unit1
string$=notesName.Org[Unit2
string$=notesName.Org[Unit3
string$=notesName.Org[Unit4
{LIB} for nerdsDomain=notesName.Parent
string$=notesName.PRMD
string$=notesName.Surname

Methods
Call notesName.Abbreviate
Call notesName.AddCertifierToAddressBook(dbHdl$,certpw$[,location$,[,comment$]])
Call notesRegistration.AddNewOrgCertifier(server$,orgCertPw$,orgName$,[,locality$,[,comment$]])
Call notesRegistration.AddNewOrgUnitCertifier(server$,orgCertPw$,orgUnit$,orgName$,[,locality$,[,comment$]])
Call notesRegistration.AddServerToAddressBook(dbHdl$,serverName$,[,serverPw$,[,networkName$,[,administrator$
[,altName$,[,location$,[,comment$]]]]])
Call notesRegistration.AddUserProfile(username$,profileName$)
Call notesRegistration.AddUserToAddressBook(dbHdl$,fullname$,lastname$[,userpw$[,firstname$[, middle$
[,mailserver$,[,maildbpath$,[,forlaterdelete$,[,location$[,comment$]]]]]]])
Call notesRegistration.AddUserToNewServer(server$,idfile$,domain$,servpw$,orgName$,[,location$,[,comment$
[,network$,[,administrator$,[,title$]]]]])
Call notesRegistration.RegisterNewCertifier(idfile$,certpw$,orgName$,[,comment$])
Call notesRegistration.RegisterNewOrgUnit(idfile$,certpw$,orgUnit$,orgName$,[,comment$])
Call notesRegistration.RegisterNewServer(server$,idfile$,domain$,servpw$,orgName$,[,location$,[,comment$
[,network$,[,administrator$,[,title$]]]]])
Call notesRegistration.RegisterNewUser(certifierIdFile$,certifierPw$,firstName$,middleInitial$,lastName$,
[,mailOwnerAccess$,[,mailSystem$,[,mailTemplate$,[,mailAlternateName$,[,mailAddress$,[,storeIDinAddressBook$
[,location$,[,comment$,[,mailInDomain$,[,networkAccount$,[,policy$,[,profile$,[,localAdminp$]]]]]]]]]]])
Call notesRegistration.RenameUser(certifierIdFile$,certifierPw$,userName$,newFirstName$,newMiddleInitial$,newLastName$,newAltName$)
Call notesRegistration.SetServerInfo(server$,administrator$,newPw$,[,networkName$,[,domain$,[,title$]]])
Call notesRegistration.SetUserProfile(username$,profileName$)
Call notesRegistration.SwitchToID(idfile$,certpw$)

Notes Agent

Properties
comment$=notesAgent.Comment
commonUserName=notesAgent.CommonOwner
{ } flag=notesAgent.HasRunSinceModified
{LIB} nerdsAgent.IsEnabled
R/W flag=notesAgent.IsActivated
flag=notesAgent.IsNotesAgent
flag=notesAgent.IsPublicAccess
name$=notesAgent.Name
{ } name$=notesAgent.Name$[0]
agentName$=notesAgent.Name
{LIB} nerdsAgent.NotesURL
name$ = notesAgent.OnHold(pUid$)
owner$=notesAgent.Owner
parameter$=notesAgent.ParameterDocID
Set notesDocument=notesAgent.Parent
R/W server$=notesAgent.Server
R/W server$=notesAgent.ServerName
{ } flagIntegerConstant=notesAgent.Target
{ } flagIntegerConstant=notesAgent.Trigger

Method
Call notesAgent.Remove
status=notesAgent.Run([runInUIContext])
{ } Call notesAgent.RunOnServer([,numDocs])
flag=notesAgent.LastPermissionIf(name$)
4.61 status=notesAgent.RunOnServer([runInUI])
4.6 Call notesAgent.Save
Call notesAgent.UnStick()

Notes Embedded Objekt

Properties
className$=notesEmbeddedObject.Class
nSleft=notesEmbeddedObject.FileSize
4.62 R/W flag=notesEmbeddedObject.FillInNewFields
4.62 R/W flag=notesEmbeddedObject.FitToWindow
name$=notesEmbeddedObject.Name
Set handle%=notesEmbeddedObject.Object
Set notesRichTextItem=notesEmbeddedObject.Parent
4.62 R/W flag=notesEmbeddedObject.RunReadOnly
nsuc=$=notesEmbeddedObject.Source
type %=notesEmbeddedObject.Type
verb$=notesEmbeddedObject.Verbs
Methods
handle%=notesEmbeddedObject.Activate(show)
variant=notesEmbeddedObject.DoVerb(verb$)
Call notesEmbeddedObject.ExtractFile(path$)
Call notesEmbeddedObject.Remove

Notes ACL

Properties
R/W serverName$=notesACL.AdministrationServer
{ } 4.6 R/W level %=notesACL.InternetLevel
R/W flag = notesACL.IsAdminOnly
R/W flag = notesACL.IsAdminReaderAuthor
R/W flag = notesACL.IsEnforceConsistency
R/W flag = notesACL.IsUniformAccess
notesACL.Parent
R/W flag=notesACL.Roles
R/W flag=notesACL.UniformAccess

Methods
Call notesACL.AddRole(name$)
Set notesACLEntry=notesACL.CreateACLEntry(name$,level %)
Call notesACL.DeleteRole(name$)
Set notesACLEntry=notesACL.GetEntry(name$)
Set notesACLEntry=notesACL.GetFirstEntry
Set notesACLEntry=notesACL.GetNextEntry(notesACLEntry)
Call notesACL.RenameRole(oldName$,newName$)
Call notesACL.ReviseRole(roleName$)

Notes ACL Entry

Properties
R/W flag=notesACLEntry.CanCreateDocuments
{ } R/W flag = notesACLEntry.CanCreatePersonalAgent
R/W flag=notesACLEntry.CanCreatePersonalFolder
R/W flag=notesACLEntry.CanCreateSharedFolder
R/W flag = notesACLEntry.CanDeleteDocuments
R/W flag = notesACLEntry.CanReplicateOrCopyDocuments
flag=notesACLEntry.IsAdminReaderAuthor
{ } R/W flag=notesACLEntry.IsAdminServer
{ } R/W flag=notesACLEntry.IsGroup
{ } R/W flag=notesACLEntry.IsPerson
R/W flag=notesACLEntry.IsPublicReader
R/W flag=notesACLEntry.IsPublicWriter
R/W flag=notesACLEntry.IsServer
R/W flag=notesACLEntry.IsUnspecified
R/W name$=notesACLEntry.Name
{LIB} Set nerdsName=notesACLEntry.NameObject

Methods
Call notesACLEntry.Disable(folder$)
Call notesACLEntry.Enable(folder$)
Call notesACLEntry.EnforceConsistency()
Call notesACLEntry.Remove

Notes Replication

Properties
R/W flag=notesReplication.Abstract
timedate=notesReplication.CutoffDate
R/W flag=notesReplication.CutoffDelete
R/W flag=notesReplication.DontSendLocalSecurityUpdates
R/W flag=notesReplication.Ignore$Deletes
R/W flag=notesReplication.IgnoreDestDeletes
R/W flag=notesReplication.Priority

Methods
Call notesReplication.ClearHistory()
Set notesReplicationEntry=notesReplication.GetEntry(source$,destination$[,createFlag])
Call notesReplication.Reset()
Call notesReplication.Save()

Notes Newsletter

Properties
R/W flag=notesNewsletter.DoScore
R/W flag=notesNewsletter.DoSubject
R/W name$=notesNewsletter.SubjectItemName
Methods
Set notesDocument=notesNewsletter.FormatDocument(notesDatabase, %)
Set notesDocument=notesNewsletter.FormatMsgWithDoclinks(notesDatabase)

Notes Outline

R/W alias$=notesOutline.Alias
R/W comment$=notesOutline.Comment
R/W name$=notesOutline.Name
Set notesOutlineEntry=notesOutline.ParentDatabase

Methods
Call notesOutline.AddEntry(newEntryLevel%,newEntryAlias$,[,AsChild])
{ LIB} Set notesOutlineEntry=notesOutline.CreateEntry(name$,[,addBefore][,asChild])
{ LIB} Set notesOutlineEntry=notesOutline.CreateEntryFrom(previousEntry$,[,addAfter][,asChild])
{ LIB} Set notesOutlineEntry=notesOutline.CreateEntryFrom(previousEntry$,[,addAfter][,asChild])
Set notesOutlineEntry=notesOutline.GetChildEntry
Set notesOutlineEntry=notesOutline.GetFirstEntry
Set notesOutlineEntry=notesOutline.GetLastEntry
Set notesOutlineEntry=notesOutline.GetNextEntry(notesOutlineEntry)
Set notesO