Pica

Handleiding Scripting in WinIBW

Leiden, december 2002
DN 059b/0212

Inhoudopgave

1. Introductie Scripting

1.1. Inleiding

In dit hoofdstuk worden de basisbeginselen van scripting in WinIBW behandeld. U kunt door middel van scripts veel terugkerende handelingen of complexe databasebewerkingen automatisch laten uitvoeren. Het is mogelijk om deze scripts als functies in menu's of werkbalken op te nemen, zodat ze volledig geïntegreerd zijn met andere WinIBW functies.

1.2. WinIBW gereed maken voor scripting

Voordat u met scripting aan de gang kan, moet u, als het nog niet aanwezig is, een scripting menu plaatsen in de WinIBW menubalk. Dit kunt u als volgt doen:

- Kies in het menu Opties | Aanpassen… .
- Kies in de tab commando's de categorie Script.
- Kies in de categorie Script het commando Script.
- Sleep het commando Script in de menubalk.
- Vink het keuzerondje Tekst aan.
- Klik op OK.

Nu is er een menuoptie Script, vanwaar u verschillende scripting mogelijkheden kan starten. In deze handleiding wordt er van uitgegaan dat het scripting menu in de WinIBW menubalk heeft gezet.
Het scripting menu bestaat uit de volgende onderdelen:

Menuoptie:	**Functie / Omschrijving:**
Uitvoeren:	De scriptcode wordt opnieuw ingelezen.
Pauze:	Tijdens het uitvoeren van het script wordt het script tijdelijk gestopt.
Stoppen:	Het uitvoeren van het script stoppen.
Script Opnemen:	De 'Learning Edition' activeren (zie 1.3.1).
Functie Toevoegen:	Een script toevoegen (zie 1.3.2)
Bewerken:	Een script bewerken

1.3. Een script maken

U kunt op twee verschillende manieren een script maken. De meest eenvoudige manier is de 'learning edition', maar u kunt ook zelf de scripts maken.

1.3.1. Scripts maken via de 'Learning Edition'

De 'Learning Edition' (LE) is de script variant van het opnemen van een macro (zie handleiding WinIBW, hoofdstuk 6.2.1). Het principe van LE is dat een aantal handelingen die u uitvoert als het ware wordt opgenomen, waarna dit steeds weer kan worden afgespeeld.

Als u nog nooit scripts heeft gemaakt, is de LE ideaal om kennis te maken met scripting. Scripts die met behulp van de LE worden gemaakt, worden door LE zelf gedocumenteerd, zodat u precies kan zien op welke manier WinIBW uw handelingen in scripts omzet. Hierover later meer.

In onderstaand voorbeeld maakt u een script waarmee u een zoekactie in een database doet. Het voorbeeld gaat er van uit dat u in een bestand (bijvoorbeeld GGC) bent ingelogd. U gaat een script maken waarmee een zoekactie wordt gedaan.

- Kies de optie Script opnemen in het menu script (deze wordt hierdoor aangevinkt).
- Doe een zoekactie, bijvoorbeeld ‘z ttl pica’, gevolgd door enter.
- Kies op optie Script opnemen opnieuw (deze wordt hierdoor uitgevinkt).
- Geef een naam voor, bijvoorbeeld ‘Zoeken’
- Klik op ‘OK’

Nu wordt het venster script getoond. Daar staat de code van het script in. Als u bovenstaand voorbeeld heeft gevolgd, staat er (o.a.) het volgende:

```
Sub Zoeken()
   ‘ Send the command “z ttl pica” to the system and display the data in
   ‘ the same window
   Application.ActiveWindow.Command “z ttl pica”, False
End Sub
```

De eerste regel van een code is “Sub <naam van het script>()”, de laatste regel bestaat uit “End Sub”. Dit is bij elk script hetzelfde, het geeft het begin en het eind van het script aan. Tussen deze regels staat de inhoud van het script.
Omdat u dit script met LE heeft gemaakt, staat voor elke regel script een regel met uitleg. De regel is uitgecommetarieerd met behulp van het ‘-teken. Er blijft één regel over, met de code die ervoor zorgt dat het commando “z ttl pica” naar de server wordt opgestuurd. In hoofdstuk 2 wordt uitgebreid ingegaan op de codes die in de scripts gebruikt kunnen worden.
Omdat u op dit moment het script niet gaat aanpassen, kunt u dit venster sluiten.

Vervolgens wordt het dialoogvenster Aanpassen getoond, hiermee kunt u een sneltoets aan het zelf gemaakte script kan toewijzen. Dit kunt u als volgt doen:

- Druk op de toetscombinatie die u wilt toekennen aan de betreffende functie. Indien de toetscombinatie nog niet is toegewezen, geeft het programma dit aan onder het vak Nieuwe sneltoets. Indien de toetscombinatie al is toegewezen aan een andere functie geeft het programma aan welke functie met de toetscombinatie is geassocieerd.
- Klik op de knop Toewijzen. De gekozen sneltoets wordt nu geassocieerd met de betreffende functie. U kunt overigens meerdere sneltoetsen toekennen aan een bepaalde functie.
- Klik op Sluiten om het dialoogvenster Aanpassen te sluiten.

U heeft nu uw eerste script gemaakt! Het script kunt uw oproepen door het intoetsen van de sneltoets die u zojuist heeft gemaakt.

1.3.2. Zelf scripts maken

U kunt ook zonder de LE scripts maken. Dit gaat als volgt:

- Kies de optie Functie toevoegen in het menu Script
- Geef de naam van de functie, bijvoorbeeld ‘Tonen'
- Klik op ‘OK”

Vervolgens krijgt u het scripteditor venster , waarin u de code gewoon kunt intikken. U ziet dat het script opnieuw begint met “Sub <naam van het script>()” en “End Sub”

Als voorbeeld kunt u onderstaand script maken:

```
Sub Tonen()
   ‘Place your function code here
   Application.ActiveWindow.Command “t s1”, false
End Sub
```

Als u “Application.” heeft getikt, zult u merken dat er een popup-keuzelijst verschijnt met mogelijke termen die na “Application.” kunnen worden geplaatst. Als u de eerste letters van een woord tikt, ziet u dat de selectiebalk in het window vanzelf naar de juiste term gaat. Een spatie is vervolgens voldoende om de term te plaatsen. U kunt de term ook selecteren door deze te dubbelklikken. Zie voor meer informatie Appendix A van de handleiding.

1.4. Een script op menu of werkbalk plaatsen

Als u een script in een menu of werkbalk wil plaatsen, gaat dit op dezelfde wijze als het plaatsen van functies in een menu of werkbalk. In onderstaand voorbeeld gaat u het script dat u in 1.3.1 met LE heeft gemaakt als pictogram in de algemene WinIBW werkbalk plaatsen. U kunt dit als volgt doen:

- Kies de optie Aanpassen… in het menu Opties. Het venster Aanpassen verschijnt.
- Selecteer indien nodig het tabblad Commando's.
- Selecteer in de keuzelijst Categorieën Functies.
- De door u gemaakte scripts verschijnen, samen met een aantal scripts die al door Pica zijn gemaakt, in het keuzevak Commando's.
- Sleep de functie zoeken vanuit het vak Commando's naar zijn plaats op de werkbalk
- Een invoegteken verschijnt op de plaats waar de functie zal worden neergezet.
- Het dialoogvenster Knopvlakuiterlijk verschijnt.
- Selecteer een pictogram voor de functie zoeken (zie eventueel paragraaf 5.2.2 in de WinIBW handleiding).
- Klik op Sluiten om het dialoogvenster Aanpassen te sluiten.

2. Bibliografische bewerkingen

2.1. Inleiding

In dit hoofdstuk wordt uitgelegd hoe met behulp van scripting bibliografische bewerkingen kunnen worden gedaan. Uitgangspunt hierbij is dat GGC-records gemuteerd worden, maar de bewerkingen zijn ook geschikt voor het lokaal catalogiseren.
Als u bekend bent met vorige WinIBW-versies, zult u hier alle mogelijkheden vinden waarvoor in het verleden de upload-functionaliteit voor werd gebruikt.

De meeste onderdelen worden beschreven aan de hand van 3 onderdelen:

- Syntax: De wijze waarop een bepaalde bewerking in het script wordt opgenomen.
- Uitleg: De delen van de syntax worden beschreven.
- Voorbeeld: De uitleg wordt verduidelijkt aan de hand van een voorbeeld.

Naarmate dit handleiding vordert zullen de bewerkingen complexer worden. In principe worden alleen de WinIBW-zaken behandeld, bij VB-script zaken wordt niet lang stil gestaan. Als u niet bekend bent met VB-scripting, is het verstandig om hierover documentatie te lezen. Zie bijvoorbeeld http://msdn.microsoft.com/scripting/ voor meer informatie.

2.2. Basisbeginselen

In deze paragraaf worden een aantal basisbeginselen van scripting uitgelegd. Met deze basisbeginselen gaat u een script maken dat een aantal handelingen voor u uitvoert. Hoe u het invoervenster voor een script kunt openen, staat in hoofdstuk 1.3.2 van deze handleiding.

2.2.1. Het begin en einde van een script

Een script heeft een begin en een eind.

syntax:
Sub <naam van het script>()
End Sub

uitleg:
Als een script wordt gemaakt, worden deze twee regels altijd automatisch gedefinieerd. De eerste regel geeft aan dat het script begint, de tweede geeft het einde van het script aan.

voorbeeld:
```
Sub Deelder()
End Sub
```

Bovenstaand script heet Deelder. Er staat nog geen code in, deze gaan we in de volgende paragraaf maken.

2.2.2. Commando's uitvoeren

Als u met WinIBW aan het werken bent, doet u dat vaak aan de hand van opgegeven Pica3- commando's in de commandoregel. U kunt al deze commando's in een script opnemen.

syntax:
Application.ActiveWindow.Command "<commando>", Boolean

uitleg:
Application.ActiveWindow.Command "<commando>" : stuurt het commando naar de server toe.
Boolean: bepaal of het commando in een nieuw window moet worden uitgevoerd (True) of in het huidige window moet worden uitgevoerd (False).

voorbeeld:
```
Sub Deelder()
   Application.ActiveWindow.Command "z aut deelder, jules", False
End Sub
```

Bovenstaande script heeft als gevolg dat de zoekactie "z aut deelder, jules" wordt in het huidige window uitgevoerd.

2.2.3. Regelmatig terugkerende handelingen uitvoeren

U kunt veel regelmatig terugkerende handelingen (bijvoorbeeld het bewerken van titels, het geschiedenisscherm tonen etc.) in scripts opnemen

syntax:
Application.ActiveWindow.SimulateIBWKey "<functie>"

uitleg:
Application.ActiveWindow.SimulateIBWKey "<functie>" : stuurt een functie naar de server toe.
De volgende functies kunt u gebruiken:

F1	Vooruit bladeren naar het volgende scherm
F2	Terug bladeren naar het vorige scherm
F3	Toont het geschiedenisscherm
F4	Toont het index-overzicht
F5	Verwijderen van schermgegevens
F6	Invoeren van een nieuwe titel
F7	Bewerken van de huidige titel
FE	Escape
FR	Enter

voorbeeld:
```
Sub Deelder()
   Application.ActiveWindow.Command "z aut deelder, jules", False
   Application.ActiveWindow.Command "t 16",
   Application.ActiveWindow.SimulateIBWKey "F7"
End Sub
```

In bovenstaand voorbeeld wordt er eerst een zoekactie gedaan, vervolgens wordt treffer 16 van set 1 getoond, daarna wordt deze titel in bewerkingstand gezet.

2.2.4. Commentaar in scripts toevoegen

U kunt commentaar toevoegen aan scripts. Dit kan erg handig zijn, het kan de inhoud van de scripts verhelderen.

syntax:
‘ <commentaar>

uitleg:
Tekst die na het ‘ teken volgt, wordt niet uitgevoerd door het script.

voorbeeld:
```
Sub Deelder()
   ‘ In dit script wordt na een zoekactie een record in bewerken gezet.
   Application.ActiveWindow.Command "z aut deelder, jules", False
   Application.ActiveWindow.Command "t s1 16",
   Application.ActiveWindow.SimulateIBWKey "F7"
End Sub
```

In bovenstaand voorbeeld wordt in een commentaarregel vermeld wat het script doet. De commentaarregel verschijnt in een andere kleur (groen) en wordt voorafgegaan door ‘ (enkele quote).

2.3. Tekst invoeren en bewerken

U kunt een aantal tekstbewerkinghandelingen in scripts verwerken. Dit voert u uiteraard altijd in de GGC uit (of in de OWC als u lokaal catalogiseert).

2.3.1. Tekst invoeren

Met een script kunt u eenvoudig teksten in een invoerscherm invoeren.

syntax:
Application.ActiveWindow.Title.InsertText “<tekst>” & vbCr

uitleg:
Application.ActiveWindow.Title.InsertText "<tekst>": voert de tekst in het invoerscherm.
Application.ActiveWindow.Title.InsertText vbCr: stuurt een harde return naar het invoerscherm.

Teksten en codes kunnen door elkaar heen gebruikt worden, zolang ze maar onderscheiden worden met het &-teken. Eigenlijk is dit een concatenatie van strings. Ze kunnen ook apart van elkaar worden gebruikt.

voorbeeld:
```
Sub Blokje()
   ‘ Dit script voert een “blokje” in in de GGC
   Application.ActiveWindow.Command “inv”
   Application.ActiveWindow.Title.InsertText “0500 Aay”
   Application.ActiveWindow.Title.InsertText vbCr
   Application.ActiveWindow.Title.InsertText “1100 2000” & vbCr
   Application.ActiveWindow.Title.InsertText “1500 /1ned” & vbCr
   Application.ActiveWindow.Title.InsertText “1700 /1nl” & vbCr
   Application.ActiveWindow.Title.InsertText “4000” & vbCr & “4030” & vbCr & “4062”
End Sub
```

Bovenstaand voorbeeld voert de voornaamste kmc’s in het invoerscherm in. Er worden tevens een aantal default gegevens ingevuld.

2.3.2. Tekst zoeken en selecteren

U kunt via een script bepaalde teksten zoeken en selecteren. Daarvoor moet u altijd het record dat u wilt doorzoeken in de bewerkingsstand zetten.

syntax:
Application.ActiveWindow.Title.Find "<tekst>", Boolean1, Boolean2

uitleg:
Application.ActiveWindow.Title.InsertText "<tekst>": zoekt de tekst op in het invoerscherm.
Boolean1: bepaald of er hoofdlettergevoelig moet worden gezocht (true) of niet (false).
Boolean2: bepaald of er alleen in de huidige regel moet worden gezocht (true) of in de gehele tekst van het invoerscherm (false).

voorbeeld:
```
Sub Zoeken()
   ‘Dit script vindt de tekst “Den Haag” in ppn 189628715.
   Application.ActiveWindow.Command “z ppn 189628715”
   Application.ActiveWindow.SimulateIBWKey “F7”
   Application.ActiveWindow.Title.Find "Den Haag", False, False
End Sub
```

In bovenstaand voorbeeld wordt er in een bepaalde titel naar de tekst Den Haag gezocht. Er wordt in het hele record gezocht, waarbij hoofd- en kleine letters niet uitmaken. De gevonden tekst wordt automatisch geselecteerd.

2.3.3. Tekst zoeken en vervangen in een record

U kunt een script ook tekst zoeken, en deze tekst vervolgens vervangen. U kunt dit op twee manieren doen.
Bij de eerste methode zoekt u één keer naar een bepaalde tekst die u vervolgens vervangt.
Een bepaalde tekst kan echter meerdere keren in een record voorkomen. Bij de tweede methode vervangt u alle tekst met één handeling (de alles vervangen functie). Uiteraard moet het record in bewerking zijn. In feite komt het zoeken en vervangen in een record neer op muteren.

2.3.3.1. Een tekst zoeken en vervangen

syntax:
Application.ActiveWindow.Title.Replace “<tekst>”, Boolean

uitleg:
Application.ActiveWindow.Title.Replace "<tekst>": vervangt de geselecteerde tekst met de hier aangegeven tekst.
Boolean: bepaald of er hoofdlettergevoelig moet worden gezocht (true) of niet (false).

voorbeeld:
```
Sub Vervang()
   ‘Dit script vervangt een bepaalde tekst in ppn 181981130.
   Application.ActiveWindow.Command “z ppn 181981130”
   Application.ActiveWindow.SimulateIBWKey “F7”
   Application.ActiveWindow.Find “8731 D 22”, False
   Application.ActiveWindow.Title.Replace “8731 E 22” , False
   Application.ActiveWindow.SimulateIBWKey “FR”
End Sub
```

In bovenstaand voorbeeld wordt er eerst gezocht naar 8731 D 22 , en wordt deze tekst, mits gevonden en dus geselekteerd, vervangen door 8731 E 22.

2.3.3.2. Een tekst zoeken en vervangen (alles vervangen)

syntax:
Application.ActiveWindow.Title.ReplaceAll “<tekst1>”, “<tekst2>”, Boolean

uitleg:
Application.ActiveWindow.Title.ReplaceAll "<tekst1>": zoekt de tekst op in het invoerscherm en selecteert de tekst.
<tekst2>: vervangt de geselecteerde tekst met de hier aangegeven tekst
Boolean: bepaald of er hoofdlettergevoelig moet worden gezocht (true) of niet (false)

voorbeeld:
```
Sub Vervang2()
   ‘Dit script vervangt een bepaalde tekst eventueel meerdere keren in ppn 181981130.
   Application.ActiveWindow.Command “z ppn 181981130”
   Application.ActiveWindow.SimulateIBWKey “F7”
   Application.ActiveWindow.Title.ReplaceAll "8731 D 22", “8731 E 22” , False
   Application.ActiveWindow.SimulateIBWKey “FR”
End Sub
```

In bovenstaand voorbeeld wordt de tekst 8731 D 22 vervangen door 8731 E 22. Als de tekst meerdere keren in het record voorkomt, worden deze allemaal vervangen.

2.3.4. Tekst verwijderen

U kunt met behulp van een script bepaalde teksten in een record verwijderen.

syntax:
Application.ActiveWindow.Title.DeleteSelection

uitleg:
Application.ActiveWindow.Title.DeleteSelection: verwijderd een geselecteerde tekst.
Om een geselecteerde tekst te verwijderen, moet u deze uiteraard eerst selecteren.

voorbeeld:
```
Sub Verwijd()
   ‘Dit script verwijderd een bepaalde tekst in ppn 181981130 .
   Application.ActiveWindow.Command “z ppn 181981130”
   Application.ActiveWindow.SimulateIBWKey “F7”
   Application.ActiveWindow.Title.Find "D 22", False, False
   Application.ActiveWindow.Title.DeleteSelection
   Application.ActiveWindow.SimulateIBWKey “FR”
End Sub
```

In bovenstaand script wordt een record in bewerking genomen. Er wordt een tekst gezocht (en dus geselecteerd). Vervolgens wordt deze tekst verwijderd en het record wordt opgeslagen.

2.4. Specifiekere bewerkingen

Naast het algemeen bewerken van gegevens, zijn er ook een aantal specifieke handelingen die u in scripts kunt verwerken. In de paragraaf passeren de voornaamste mogelijkheden de revue. De handelingen zijn vaak combinaties van de bewerkingen die in paragraaf 2.2 zijn behandeld.

2.4.1. Een regel (kmc) toevoegen

U kunt een kmc toevoegen aan een al bestaand record. In een script kun u dit het makkelijkst doen door aan het eind van het record het kmc toe te voegen.

syntax:
Application.ActiveWindow.Title.EndOfBuffer
Application.ActiveWindow.Title.InsertText “<tekst> “

uitleg:
Application.ActiveWindow.Title.EndOfBuffer: zet de cursor aan het einde van het record.
Application.ActiveWindow.Title.InsertText “<tekst> “: voegt tekst toe.

voorbeeld:
```
Sub KMCToe()
   ‘Dit script voegt kmc 4205 in ppn 102181144 toe.
   Application.ActiveWindow.Command “z ppn 102181144”
   Application.ActiveWindow.SimulateIBWKey “F7”
   Application.ActiveWindow.Title.EndOfBuffer
   Application.ActiveWindow.Title.InsertText “4206 Bevat de verhalen”
   Application.ActiveWindow.SimulateIBWKey “FR”
End Sub
```

In bovenstaand script wordt een record in bewerking genomen. Eerst wordt de cursor aan het einde van het record gezet, vervolgens wordt er tekst toegevoegd, en het record wordt opgeslagen.

2.4.2. Een regel (kmc) “conditioneel” toevoegen

U kunt met behulp van scripting bepalen of een bepaald record aan criteria voldoet om deze te bewerken of niet. Als u WinIBW al langer gebruikt, herkent u hier waarschijnlijk de .CADD token uit de uploadfunctionaliteit.

Er is niet één syntax om een kmc afhankelijk van een bepaald criterium wel of niet in te voeren (er zijn meerdere handelingen voor nodig). Daarom volgt nu een voorbeeld, met een uitleg. In het voorbeeld wordt elke regel voor het gemak genummerd, als u dit script wilt invoeren moet u dit uiteraard zonder regelnummers doen.

voorbeeld:
```
1  Sub KMCCon()
2     Application.ActiveWindow.Command “z ppn 181981130”
3     Application.ActiveWindow.SimulateIBWKey “F7”
4     Application.ActiveWindow.Title.Find vbCr & "3261", False, False
5
6     If Application.ActiveWindow.Title.GetSelection() = “3261” Then
7        Application.ActiveWindow.Title.EndOfBuffer
8        Application.ActiveWindow.Title.InsertText “3262 @Ottoman Empire”
9     Else
10       Application.ActiveWindow.Title.EndOfBuffer
11       Application.ActiveWindow.Title.InsertText “3261 @Ottoman Empire”
12    End If
13
14    Application.ActiveWindow.SimulateIBWKey “FR”
15 End Sub
```

Bovenstaand script voert, afhankelijk van de aanwezigheid van een bepaald kmc, handelingen uit. De handelingen bestaan uit het toevoegen van een kmc met inhoud.
Regel 1 tot en met 4 zoeken een bepaalde ppn op (regel 2), zet deze in bewerking (3) en zoekt of kmc 3261 aanwezig is (4). Als kmc 3261 aanwezig is, wordt de tekst “3261” geselecteerd.
Regel 6 tot en met 12 bevat het toevoegen van het kmc. Eerst (6) wordt gekeken of de tekst “3261” is geselecteerd (GetSelection). Er zijn twee mogelijkheden: de tekst is geselecteerd, of niet.
Als de tekst geselecteerd is, wordt doorgegaan met regel 7 (de cursor wordt aan het einde van het record gezet) en 8 (de tekst “3262 @Ottoman Empire” wordt toegevoegd).
Als de tekst niet geselecteerd is, wordt doorgegegaan met regel 10 (de cursor wordt aan

het einde van het record gezet) en regel 11 (de tekst “3261 @Ottoman Empire” wordt toegevoegd).
Regel 14 slaat het record op.

2.4.3. Beter naar kmc’s zoeken

In het voorbeeld in paragraaf 2.4.2 wordt naar een kmc gezocht met behulp van een zoekactie naar de combinatie harde return en het nummer van het kmc. Er is een betere methode om er achter te komen of een bepaald kmc aanwezig is: het FindTag-commando.
Het FindTag-commando geeft de waarde van een kmc terug. Om wat te kunnen doen met het resultaat, moet de waarde worden opgevangen in bijvoorbeeld een variabel. Dit komt later in de paragraaf bij de voorbeelden nog aan de orde.

syntax:

Application.ActiveWindow.Title.FindTag(“<kmc>”, Volgnummer, Boolean, Boolean)

uitleg:

Application.ActiveWindow.Title.FindTag(
<kmc>: de kmc(‘s) die gezocht wordt. Het ? kan worden gebruikt om te maskeren, bijvoorbeeld 40??
Volgnummer: Het zoekargument dat bij kmc is ingevuld kan meerdere keren in een titel voorkomen. Het volgnummer bepaald op welke kmc de selectie wordt gedaan. Houd er rekening mee dat de teller bij nul begint!
Boolean: deze boolean geeft aan of de gezochte kmc zelf ook wordt teruggegeven.
Boolean: deze boolean geeft aan of de inhoud van de gezochte kmc geselecteerd wordt.
)

voorbeeld:

```
1  Sub Zoek()
2     ‘Dit script zoekt naar het derde kmc dat begint met 40 in ppn 181981130 en
3     ‘geeft de inhoud inclusief het kmc in een venster weer. De cursor wordt niet
4     ‘naar de regel waar het kmc wordt gevonden verplaatst.
5     Application.ActiveWindow.Command “z ppn 181981130”
6     Application.ActiveWindow.SimulateIBWKey “F7”
7     Resultaat = Application.ActiveWindow.Title.FindTag(“40??”, 2, True, False)
8     MsgBox Resultaat
9
10 End Sub
```

In bovenstaand script wordt in regel 5 en 6 ppn 181981130 opgezocht en in muteren gezet. In regel 7 wordt het resultaat van het FindTag commando naar de variabel Resultaat geschreven. In regel 8 wordt de inhoud van de variabel in een venster getoond.

Ter verduidelijking nog enkele resultaten van het FindTag commando. Uitgangspunt is ppn 181981130 welke uit de volgende kmc’s bestaat:

```
0500 Aav
1100 1999
1121 j
1500 /1eng
1700 /1nl
2000 9004113533*geb.
2020 B9908152
2040 9927409
3000 Edhem@Eldem!088815595!Edhem Eldem
4000 @French trade in Istanbul in the eighteenth century / by Edhem Eldem
4030 Leiden [etc.] : Brill
4060 XIV, 321 p
```

```
4061 ill
4062 25 cm
4170 The @Ottoman Empire and its heritage, ISSN 1380-6076 ; vol. 19
4180 #190#!113743300!The @Ottoman empire and its heritage, ISSN 1380-6076
; vol. 19
4204 Lit. opg.: p. 297-321
4700 TR.
4700 YY
5201 !07560857X!handel
5202 !075690144!Istanbul
5300 1005
5301 !077599594!15.70 geschiedenis van Europa
5311 17XX
5401 !078509572!Fransen
5402 !07846126X!Buitenlandse handel
5421 !078550505!5.205 Istanbul
4701 md/dok
7001 16-06-99 : gda
7100 4129588 !d! @ f
8009 md/24
8322 f. 160.- (Serieprijs f. 95.-)
7800 30671292X
7002 05-02-01 : gwB
7100 MED 461 @ u
8100 00035897//EvW
8200 13372245
7800 327336307
```

FindTag:	**Resultaat:**
FindTag(“4030”, 0, False, False)	Leiden [etc.] : Brill
FindTag(“4030”, 0, True, False)	4030 Leiden [etc.] : Brill
FindTag(“7100”, 0, True, False)	7100 4129588 !d! @ f
FindTag(“7100”, 1, False, False)	MED 461 @ u
FindTag(“5?01”, 1, True, False)	5301 !077599594!15.70 geschiedenis van Europa
FindTag(“5???”, 4, False, False)	17XX

2.4.4. Een regel (kmc) verwijderen

U kunt in een script een kmc met inhoud verwijderen. Dit doet u door het kmc te zoeken en als het kmc aanwezig is wordt de regel verwijderd.

syntax:
Application.ActiveWindow.Title.DeleteLine

uitleg:
Application.ActiveWindow.Title.DeleteLine: verwijderd de regel waar de cursor op dit moment staat.

voorbeeld:
```
1  Sub KMCVer()
2     ‘Dit script verwijderd kmc 4206 in ppn 181981130.
3     Application.ActiveWindow.Command “z ppn 181981130”
4     Application.ActiveWindow.SimulateIBWKey “F7”
5     Resultaat = Application.ActiveWindow.Title.FindTag(“4206”, 0, False, True)
6
7     If Resultaat = “” Then
8        Else
9        Application.ActiveWindow.Title.DeleteLine
10    End If
11
12    Application.ActiveWindow.SimulateIBWKey “FR”
13
14 End Sub
```

In bovenstaand script wordt (regel 3 en 4) een record gezocht en in bewerking genomen. Er wordt naar de inhoud van kmc 4206 gezocht, het resultaat wordt naar een variabel met de naam Resultaat geschreven(5). Als kmc 4206 wordt gevonden, heeft de variabel Resultaat een inhoud, als kmc 4206 niet wordt gevonden, heeft de variabel geen inhoud. Vervolgens wordt in regel 7 tot en met 10 aan de hand van de inhoud van de variabel Resultaat bekeken of de regel moet worden verwijderd of niet. Het record wordt opgeslagen.

2.5. Bestanden benaderen

In sommige situaties is het handig om gegevens naar bestanden weg te schrijven of in te lezen.

2.5.1. Gegevens naar een bestand wegschrijven

U kunt met een script gegevens naar bestanden wegschrijven. Dit gaat in drie stappen:

- Bestand creëren en openen:

  syntax:
  Set fso = CreateObject("Scripting.FileSystemObject")
  Set Bestand = fso.CreateTextFile("c:\logfile.txt")

- Gegevens naar het bestand schrijven:

  syntax:
  Bestand.WriteLine(<gegevens>)

- Bestand sluiten

  syntax:
  Bestand.Close

Op deze stappen wordt qua uitleg niet verder ingegaan, het is volledig VB scripting. Het mag duidelijk zijn dat fso.CreateTextFile(“<bestandsnaam>”) de bestandsnaam is waar de gegevens naar toe wordt geschreven.

voorbeeld:

```
1  Sub Uit()
2     Set fso = CreateObject("Scripting.FileSystemObject")
3     Set Bestand = fso.CreateTextFile("c:\logfile.txt", True)
4
5     Application.ActiveWindow.Command "z sel van"
6     Application.ActiveWindow.Command "t 1"
7
8     Aantal = Application.ActiveWindow.Variable("P3GSZ")
9
10    For Teller = 1 To Aantal
11       Bestand.WriteLine(Application.ActiveWindow.Variable("P3GPP"))
12       Application.ActiveWindow.SimulateIBWKey "F1"
13    Next
14
15    Bestand.Close
16 End Sub
```

Bovenstaand voorbeeld schrijft alle in een set aanwezige ppn’s naar een file. Regel 2 en 3 maken een bestand (c:\logfile.txt) aan, regel 5 en 6 voeren twee commando’s uit, waardoor er een set wordt gecreëerd.

Elk scherm van WinIBW heeft een aantal gegevens (ofwel variabelen), bijvoorbeeld het aantal treffers in een set, of het PPN bij het tonen van een volledige presentatie. In regel 8 wordt de variabel “P3GSZ” (dit is het aantal treffers van de huidige set) naar het scriptvariabel Aantal weggeschreven. Dit aantal treffers wordt in regel 10 tot en met 14 gebruikt om een aantal handelingen een bepaald aantal keren uit te voeren. Deze handelingen zijn het wegschrijven van het PPN naar de tekstfile (11), en het volgende record selecteren(12). Het PPN wordt opgehaald aan de hand van de variabel “P3GPP” (dit is het ppn van een record dat volledig wordt gepresenteerd). Regel 15 sluit het bestand.

2.5.2. Gegevens uit een bestand ophalen

Uiteraard kunt u ook gegevens uit bestanden gebruiken in scripts. Hiervoor zijn drie stappen nodig:

- Bestand openen:

  syntax:
  Set fso = CreateObject("Scripting.FileSystemObject")
  Set Bestand = fso.OpenTextFile("c:\logfile.txt")

- Gegevens uit een bestand ophalen:

  syntax:
  <variabel> = Bestand.ReadLine

- Bestand sluiten

  syntax:
  Bestand.Close

Op deze stappen wordt qua uitleg niet verder ingegaan, het is volledig VB scripting. Het mag duidelijk zijn dat fso.OpenTextFile(“<bestandsnaam>”) de bestandsnaam is waar de gegevens uit worden ingelezen.

voorbeeld:

```
1  Sub In()
2     Set fso = CreateObject("Scripting.FilesystemObject")
3     Set Bestand = fso.OpenTextFile("c:\logfile.txt")
4
5     Dim PPN(20)
6
7     For Teller = 1 To 20
8        PPN(Teller) = Bestand.ReadLine
9     Next
10
11    Bestand.Close
12
13    Application.ActiveWindow.Command "z ppn " & PPN(10)
14 End Sub
```

Bovenstaand voorbeeld leest de file die in 2.5.1 is gemaakt. Er wordt vanuit gegaan dat er meer dan 20 PPN’s zijn weggeschreven naar de file. In regel 2 en 3 wordt de file geopend. Regel 5 maakt de scriptvariabel PPN als tabel, waar de ppn’s naar worden weggeschreven. Dit gebeurd in regel 7, 8 en 9. Tot slot wordt het bestand gesloten en één van de ppn’s die zijn ingelezen gebruikt in een commando.

2.6. Pauze-mogelijkheden

U kunt tijdens het uitvoeren van een script het script tijdelijk laten stoppen. Er zijn een aantal varianten, waarvan er vier verschillende als voorbeeld zijn opgenomen.

2.6.1. Pauze (zonder aktiviteiten)

De meest simpele vorm om een pauze in een script is een venster te laten verschijnen met een mededeling.

Voorbeeld:

```
1  Sub Pauze1()
2
3     Application.ActiveWindow.Command “rec t ; z aut baantjer”
4     Application.ActiveWindow.Command “t s1 1”
5     Msgbox “Druk op enter of klik op ok om verder te gaan”
6     Application.ActiveWindow.Command “t s1 2”
7
8  End Sub()
```

Bovenstaand voorbeeld presenteert na het tonen van de eerste titel uit set 1 een venster met de tekst “Druk op enter op klik op ok om verder te gaan”. Wanneer op enter is gedrukt of op ok is geklikt wordt het script hervat.

2.6.2. Pauze (met aktiviteiten)

Er kan ook een pauze in een script worden ingebouwd die de gebruiker de mogelijkheid geeft zelf aktiviteiten te doen die niet in het script staan vermeld.

Voorbeeld:

```
1  Sub Pauze1()
2
3     Application.ActiveWindow.Command “rec t ; z aut baantjer”
4     Application.ActiveWindow.Command “t s1 1”
5     Application.Pause
6     Application.ActiveWindow.Command “t s1 2”
7
8  End Sub()
```

Bovenstaand voorbeeld stopt na het tonen van treffer 1 uit set 1. De gebruiker krijgt de mogelijkheid om nu zelf aktiviteiten te doen die niet in het script staan. Om het script te hervatten moet de gebruiker de functie “script hervatten” in het menu script kiezen.

2.6.3. Pauze (beslissing ja/nee)

U kunt een venster laten verschijnen met een vraag die met ja of nee beantwoord kan worden. Het antwoord dat gegeven wordt kunt u vervolgens weer voor het vervolg van het script gebruiken.

Voorbeeld:

```
1  Sub Pauze2()
2
3     Application.ActiveWindow.Command "rec t ; z aut baantjer"
4     Application.ActiveWindow.Command "t s1 1"
5     antwoord = (msgbox ("Wil u de tweede treffer uit de set zien?", VbYesNo))
6
7     If antwoord = VBYes Then
8        Application.ActiveWindow.Command "t s1 2"
9        Else
10       MsgBox "Jammer!"
```

```
11    End If
12
13 End Sub
```

In dit voorbeeld wordt eerst een zoekvraag uitgevoerd en de eerste treffer getoond (regel 3 en 4). Vervolgens (regel 5) wordt een venster getoond met een vraag die met ja of nee beantwoord kan worden. Het antwoord wordt naar een variabel geschreven. Aan de hand van de waarde van de variabel (“vbyes” als ja is geklikt, “vbno” als nee is geklikt) wordt het script vervolgd.

2.6.4. Pauze (invoer van gegevens)

U kunt een venster laten verschijnen met ruimte om gegevens in te geven. De gegevens kunnen naar een variabel worden weggeschreven, waarna deze bijvoorbeeld als commando kan worden gegeven.

Voorbeeld:

```
1  Sub Pauze3()
2
3     Application.ActiveWindow.Command "rec t ; z aut baantjer"
4     Application.ActiveWindow.Command "t s1 1"
5     antwoord = (inputbox ("Welke treffer wilt u als volgende zien?"))
6     Application.ActiveWindow.Command "t s1 " & antwoord
7
8   End Sub
```

In dit voorbeeld wordt eerst een zoekvraag uitgevoerd en de eerste treffer wordt getoond (regel 3 en 4). Vervolgens (regel 5) verschijnt er een venster waar de gebruiker een gegeven kan invoeren. Het gegeven wordt naar een variabel weggeschreven. In regel 6 wordt de variabel gebruik in een nieuw commando.

2.7. Foutafhandeling

U kunt met behulp van een script grote hoeveelheden titels bewerken. Het is dan erg handig om resultaten (de ppn’s van de records die gemuteerd zijn, eventuele foutmeldingen die de validatie tijdens het opslaan van het record genereerd) weg te schrijven naar een logfile.

voorbeeld:

```
1  Sub Log1()
2     CreateObject("Scripting.FileSystemObject")
3     Set Bestand = fso.CreateTextFile("c:\log1.txt", True)
4
5     Application.ActiveWindow.Command "z sel 01-07-2000 tot 01-11-2000"
6     Application.ActiveWindow.Command "t 1"
7
8     Aantal = Application.ActiveWindow.Variable("P3GSZ")
9
10    For Teller = 1 To Aantal
11       Application.ActiveWindow.SimulateIBWKey "F7"
12       Application.ActiveWindow.Title.ReplaceAll "DE 112", "DE 117", False
13       Application.ActiveWindow.SimulateIBWKey "FR"
14
15       If Application.ActiveWindow.Status = "ERROR" Then
16          Bestand.Writeline("PPN " & Application.ActiveWindow. _
17          Variable("P3GPP") & “had een foutstatus")
18          Application.ActiveWindow.SimulateIBWKey "FE"
19       Else
20          Bestand.Writeline("PPN " & Application.ActiveWindow. _
21          Variable("P3GPP") & “had geen foutstatus")
22       End If
23       Application.ActiveWindow.SimulateIBWKey "F1"
```

```
24    Next
25
26    Bestand.Close
27 End Sub
```

Bovenstaand voorbeeld maakt in regel 2 en 3 een bestand aan. Regel 5 en 6 voert Pica3-commando's uit. Vervolgens worden op regel 10 tot en met 24 alle records van de set bewerkt. De tekst DE 112 wordt vervangen door DE 117 (12). In regel 13 wordt de bewerkte titel opgeslagen. Telkens als een titel wordt opgeslagen, wordt de status bekeken. Afhankelijk van de status wordt een bericht naar het bestand toegeschreven. Als de status "ERROR" is, kan de titel niet worden opgeslagen en moet er <escape> worden gegeven om de titel te verlaten (regel 18). In alle gevallen wordt op regel 23 naar het volgende record gegaan.

Naast de status kan ook de exacte foutmeldingen naar een file worden weggeschreven.

voorbeeld:

```
1  Sub Log2()
2     Set fso = CreateObject("Scripting.FileSystemObject")
3     Set Bestand = fso.CreateTextFile("c:\log2.txt", True)
4
5     Application.ActiveWindow.Command "z sel 01-07-2000 tot 01-11-2000"
6     Application.ActiveWindow.Command "t 1"
7
8     Aantal = Application.ActiveWindow.Variable("P3GSZ")
9
10    For Teller = 1 To Aantal
11       Application.ActiveWindow.SimulateIBWKey "F7"
12       Application.ActiveWindow.Title.ReplaceAll "DE 112", "DE 117", False
13       Application.ActiveWindow.SimulateIBWKey "FR"
14
15       If Application.ActiveWindow.Status = "ERROR" Then
16          Application.ActiveWindow.SimulateIBWKey "F9"
17          For Each msg In Application.ActiveWindow.Messages
18             Bestand.Writeline("PPN " & Application.ActiveWindow. _
19             Variable("P3GPP") & " " & msg.Text)
20          Next
21          Application.ActiveWindow.SimulateIBWKey "FE"
22       Else
23          For Each msg In Application.ActiveWindow.Messages
24             Bestand.Writeline("PPN " & Application.ActiveWindow. _
25             Variable("P3GPP") & " " & msg.Text)
26          Next
27       End If
28       Application.ActiveWindow.SimulateIBWKey "F1"
29    Next
30
31    Bestand.Close
32
33 End Sub
```

Bovenstaand script is bijna gelijk aan het eerste script in deze paragraaf. Er worden alleen andere gegevens naar de logfile geschreven. Dit gebeurd in regel 16 tot en met 19 (als er een foutmelding is) of regel 23 tot en met 25 (als er geen foutmelding is). In regel 16 wordt het record gevalideerd. Na validatie wordt er een foutmelding getoond in het berichtenvenster. Regel 17 haalt de inhoud van het berichtenvenster op, regel 18 en 19 schrijven deze, samen met het PPN en een stukje tekst, naar de logfile toe. Regel 23 tot en met 25 doet hetzelfde, maar er wordt niet gevalideerd, de status is niet "ERROR".

Appendix A. WinIBW extensions to VB Script

A.1. Introduction

WinIBW 2.000 supports scripting. In principle support of scripting is implemented generically. However, the current implementation focuses on VisualBasic script for several reasons.

WinIBW will automatically start a script as specified by user settings. The script is hooked up into WinIBW at WinIBW startup time and will start listen to events. You can stop, pause, resume or edit the script by choosing the appropriate entries from the “Script” menu.

If you choose “Edit” from the scrip menu, WinIBW will display its script editor, which allows you editing of the script. The script editor of WinIBW will assist you in editing of scripts by listing the available objects and their corresponding methods and properties as well as the syntax for invoking methods and accessing properties.

You can access a context menu in the script editor by right clicking.
At the left side of the script editor, you will see a selection margin. If you left click on this margin, you can set a breakpoint in the current script. If a script debugger is installed on your system, WinIBW will break your script into the debugger at the specified point (refer to http://msdn.microsoft.com/scripting/ and click on Script Debugger for more information). This will allow you to inspect objects and variables and step through your code or to change the contents of variables.
For standard VBScript language reference and tutorial please refer to http://msdn.microsoft.com/scripting/ and click on “VBScript” and “Documentation”.

Apart from standard VBScript features WinIBW currently supports the following global objects in its scripting environment:

- Application
- Form

In the script editor you can list the global objects with the key combination CTRL+Space or by selecting “Global Objects” from the context menu. To get help on invoking a method or accessing a property, you can choose “Quick Info” on the context menu or press CTRL+SHIFT+Space.

The Application and Form objects expose several methods and properties and they can fire events. The script editor will list the available methods or properties add the appropriate place. In case of methods, a tool-tip window will popup to display the parameters required for the method call (this is also the case for properties if they take parameters).

A.2. Application object

A.2.1. Properties

- **ActiveWindow** as ActiveWindow object (read only)
  Returns a reference to the currently active window. Refer to chapter A.5 for a detailed description of the ActiveWindow object.

- **bOverwrite** as boolean
  Indicates whether WinIBW is in INSERT (False) or OVERWRITE (True) mode.

- **Dict** as Dictionary object (read only)
  Returns a reference to a WinIBW specific instance of a Visual Basic Script dictionary object (refer to the VB script documentation).

- **Language** as string (read only)
  Returns the currently active language, return values can be
  "FR" French
  "DU" German
  "NE" Dutch
  "EN" English

- **nSizeX** as long
  specifies the width of the main application window

- **nTimerInterval** as long
  specifies the interval of the timer event in milliseconds; if set to 0 no event will be fired.

- **Windows** as WindowsCollection
  A collection of WindowObject that resembles the currently open windows. Refer to chapter A.3 for a description of the WindowsCollection properties.

A.2.2. Methods

- Sub **Activate**()
  Makes WinIBW the active window.

- Function **ActivateWindow**(*lWindowID* as long) as boolean
  Activates the window with the WindowID *lWindowID*. Returns True on success and False on failure (i.e. the window with the given ID does not exist).

- Function **CloseWindow**(*lWindowID* as long) as boolean
  Closes the window with the WindowID lWindowID. Returns True on success and False on failure (i.e. the window with the given ID does not exist).

- Function **CloseWindowsExcept**(*windowList* as WindowsCollection) as boolean
  Closes all open windows except for those specified in *windowList*. *windowList* is of type WindowsCollection and must be retrieved via the Windows property of the application object

- Function **Connect**(*strHost* as string, *strPort* as string) as boolean
  Makes a connection to the CBS and LBS. *strHost* is the machine name or IP Address; *strPort* is the port to connect to. *strPort* may be a range of ports separated by a hyphen.
  Example:
  ```
  Application.Connect("chico.pica.nl", "1024-1028")
  ```

- Function **GetFolderPath**(*nFolder* as integer) as string
  Returns a string with the path of the folder for *nFolder*. *nFolder* is a short representing a CSIDL as specified in the shell documentation and is the same as in the Shell.Namespace() function. Refer to http://msdn.microsoft.com/library/default.asp?URL=/library/psdk/shellcc/Shell/Objects/Shell/Shell.htm for a complete overview of the possible values of *nFolder*.
  Example:
  ```
  Sub FindFormFolder()
    Dim FormFolder
    FormFolder = Application.GetFolderPath(35)
    FormFolder = FormFolder & "\Pica\WinIBW20\forms\"
  End Sub
  ```

- Function **GetMachineProfileInt**(*strSection* as string, *strEntry* as string, *lDefault* as long) as long
  Retrieves the value of *strEntry* as long from the Windows registry from the key *strSection* in HKEY_LOCAL_MACHINE\Software\Pica\WinIBW20. If the value is not present, *lDefault* is returned.

- Function **GetMachineProfileString**(*strSection* as string, *strEntry* as string, *strDefault* as string) as string
  Retrieves the value of *strEntry* as string from the Windows registry from the key *strSection* in HKEY_LOCAL_MACHINE\Software\Pica\WinIBW20 . If the value is not present, *strDefault* is returned.

- Function **GetProfileInt**(*strSection* as string, *strEntry* as string, *lDefault* as long) as long
  Retrieves the value of *strEntry* as long from the Windows registry from the key *strSection* in HKEY_CURRENT_USER\Software\Pica\WinIBW20. If the value is not present, *lDefault* is returned.

- Function **GetProfileString**(*strSection* as string, *strEntry* as string, *strDefault* as string) as string
  Retrieves the value of *strEntry* as string from the Windows registry from the key *strSection* in HKEY_CURRENT_USER\Software\Pica\WinIBW20. If the value is not present, *strDefault* is returned.

- Function **HtmlToPica**(*strIn* as string) as string
  Converts the supplied string from HTML to Pica character set using the users current format.

- Function **HtmlToLatin**(*strIn* as string) as string
  Converts the supplied string from HTML character set to ISO-Latin1 using the users current format.

- Function **LatinToHtml**(*strIn* as string) as string
  Converts the supplied string from ISO-Latin1 character set to HTML character set.

- Function **LatinToPica**(*strIn* as string) as string
  Converts the supplied string form ISO-Latin1 to Pica character set

- Sub **Pause**()
  Pauses script execution until the script is resumed by invoking "Resume" on the script menu.

- Function **PicaToHtml**(*strIn* as string) as string
  Converts the supplied string form Pica to HTML character set using the users current format.

- Function **PicaToLatin**(*strIn* as string) as string
  Converts the supplied string from Pica character set to ISO-Latin1 character set

- Function **TableToPica**(*strIn* as string) as string
  Converts the supplied string from the special table character set to Pica character set

- Function **WriteMachineProfileInt**(*strSection* as string, *strEntry* as string, *lValue* as long) as boolean
  Stores the long *lValue* as content of the item *strEntry* in the Windows registry under the key *strSection* in HKEY_LOCAL_MACHINE\Software\Pica\WinIBW20 .

- Function **WriteMachineProfileString**(*strSection* as string, *strEntry* as string, *strValue* as string) as boolean
  Stores the string *strValue* as content of the item *strEntry* in the Windows registry under the key *strSection* in HKEY_LOCAL_MACHINE\Software\Pica\WinIBW20 .

- Function **WriteProfileInt**(*strSection* as string, *strEntry* as string, *lValue* as long) as boolean
  Stores the long *lValue* as content of the item *strEntry* in the Windows registry under the key *strSection* in HKEY_CURRENT_USER\Software\Pica\WinIBW20.

- Function **WriteProfileString**(*strSection* as string, *strEntry* as string, *strValue* as string) as boolean
  Stores the string *strValue* as content of the item *strEntry* in the Windows registry under the key *strSection* in HKEY_CURRENT_USER\Software\Pica\WinIBW20.

A.2.3. Events

- **Timer**
  Fired every nTimerInterval milliseconds. If property nTimerInterval is 0, the event is never fired.

A.3. WindowsCollection collection

A.3.1. Properties

- **Count** as long (read only)
  Specifies the number of items contained in this collection.

- **Item**(*lIndex* as long) as WindowObject (read only)
  Returns the WindowObject at index *lIndex* contained in this collection.

A.4. WindowObject object

A.4.1. Properties

- **Text** as string
  The Window text of this window object. The meaning of this property is dependent on the type of the window object.

- **Visible** as boolean
  Specifies if the WindowObject is visible. You can hide a window object by setting this property to False.

- **WindowID** as long (read only)
  Specifies the ID used to access this WindowObject.

A.4.2. Methods

- Sub **Close**()
  Closes this WindowObject.

- Sub **Minimize**()
  Displays this WindowObject minimized.

- Sub **Maximize**()
  Displays this WindowObject maximized.

- Sub **Restore**()
  Displays this WindowObject in its restored state.

A.5. ActiveWindow object

A.5.1. Properties

- **Caption** as string
  Specifies the caption (title) of the active window. You can get and set this property. By assigning a value to this property, the title of the active window can be changed.

- **ClipBoard** as string
  Specifies the content of the ClipBoard. You can get and set this property. By assigning a value to this property, the assigned value is placed on the ClipBoard. Although this is a property of the active window, the ClipBoard is a Windows global resource.

- **CodedData** as boolean
  Specifies if CodedData is switched on or not for the active window. You can toggle CodedData by assigning True or False to this property.

- **CommandLine** as string (write only)
  By assigning a value to this property, this value is pasted into the command line.

- **DocType** as string
  Returns or sets the document type of the current title (content of 002@) if known, an empty string otherwise.

- **Messages** as collection (read only)
  Contains a collection of Message objects. Refer to the VB script documentation for an in detail description of collections.
  Note: In contrast to some other properties of ActiveWindow, the messages are a real property of the window, i.e. each window will keep its messages property even if there are multiple connections. Refer to chapter A.7 for a description of the properties of the Messages collection.

- **NoviceMode** as boolean
  Specifies if NoviceMode is switched on or not for the active window. You can toggle NoviceMode by assigning True or False to this property

- **Status** as string (read only)
  Contains the content of the standard Pica3 variable “/V”.
  Note: In contrast to some other properties of ActiveWindow, the status is stored as a real property of the window, i.e. each window will keep its status property even if there are multiple connections.

- **Title** as TitleObject (read only)
  Returns the TitleObject of the active window if present, or nothing if not. Refer to TitleObject for a description of the methods and properties of this object.

- **TitleCopyfile** as string
  Specifies the complete path to the currently active TitleCopy file. Although this is a property of the active Window, this setting is WinIBW global. You can set the TitleCopy file by assigning the complete path of a file to this property.

- **Variable**(*strName* as string) as string
  Allows access to the internal WinIBW variables. You can retrieve and set these values. Pica3 variables are named according to the following scheme:
  P3GXX for global variables
  P3LXX for local variables
  P3VXX for fields

For example, the PPN in a full presentation is contained in the variable “P3GPP”, the currently active set in “P3GSE” and the number of titles in a set in “P3GSZ”. Besides from this, there are special variables like “scr”, “buf” etc.

- **WindowID** as long (read only)
  Specifies the WindowID of the active Window.

A.5.2. Methods

- Sub **AppendMessage**(*strMessage* as string, *lStyle* as long*)*
  Appends *strMessage* to the messageBar with *lStyle*.

- Sub **BeginOfPage**()
  Scrolls to the top of the page.

- Sub **CloseWindow**()
  Closes the active window

- Function **Command**(*strCommand* as string [, *bInNewWindow* as boolean = False]) as boolean
  Executes the command as specified in *strCommand* in the active window. If *bInNewWindow* is set to True, the command will be executed in a newly open window. *bInNewWindow* is optional and defaults to False.

- Function **CopyTitle**() as string
  Executes the title copy function in the same way as “TitleCopy” on the Edit menu. If there is no title available in the active window, the function does nothing

- Sub **EndOfPage**()
  Scrolls to the end of the page.

- Sub **LineDown**([*lCount* as number = 1])
  Scrolls *lCount* lines down. *lCount* is optional and defaults to 1.

- Sub **LineUp**([*lCount* as number = 1])
  Scrolls *lCount* lines up. *lCount* is optional and defaults to 1.

- Sub **NewWindow**()
  Opens a new window in WinIBW.

- Sub **PageDown**([*lCount* as number = 1])
  Scrolls *lCount* pages down. *lCount* is optional and defaults to 1.

- Sub **PageUp**([*lCount* as number = 1])
  Scrolls *lCount* pages up. *lCount* is optional and defaults to 1.

- Sub **PasteTitle**()
  Executes the paste title function in the same way as “PasteTitle” on the Edit menu. If there is no title edit control available on the active window, the function does nothing.

- Sub **PressButton**(*varButton* as variant)
  Takes either a string or a number as parameter. Invoking with a number will invoke the button on the button bar starting with 0, i.e. PressButton(0) will simulate a click on the first button in the button bar. Invoking with a string will invoke the button with the label specified by string, i.e. PressButton(“Invoer”) will invoke the button with the label “Invoer”.

- Sub **ShowMessage**(*strMessage* as string, *lStyle* as long)
  Shows *strMessage* in the MessageBar with *lStyle* as style. Existing messages are overwritten.

- Function **SimulateIBWKey**(*strKey* as string) as boolean
  Simulates pressing a WinIBW key, where *strKey* is the internal presentation of the key. Commonly used keys are:

"FE"	Escape
"FR"	Enter
"F1" to "F12"	Usually related to the buttons on the Pica button bar.

A.6. TitleObject object

A.6.1. Properties

- **lSelEnd** as long
  Character position of the last character of the selected text.

- **lSelStart** as long
  Character position of the first character of the selected text.

A.6.2. Methods

- Sub **CharLeft**([*lCharsLeft* as long = 1] [, *bSelecting* as boolean = False])
  Moves the cursor *lCharsLeft* characters to the left. If *bSelecting* is set to True, the text is selected. *lCharsLeft* is optional and defaults to 1. *bSelecting* is optional and defaults to False.

- Sub **CharRight**([*lCharsRight* as long = 1] [, *bSelecting* as boolean = False])
  Moves the cursor *lCharsRight* characters to the right. If *bSelecting* is set to True, the text is selected. *lCharsRight* is optional and defaults to 1. *bSelecting* is optional and defaults to False.

- Function **CopyToFile**(*strFileName* as string) as boolean
  Equivalent to CopyToFile on the Edit menu.

- Sub **DeleteLine**([*lCount* as long = 1])
  Deletes *lCount* lines in the current title at the current cursor position. *lCount* is optional and defaults to 1.

- Sub **DeleteSelection**()
  Deletes the currently selected text from this title.

- Sub **DeleteToEndOfLine**()
  Deletes to the end of the current line.

- Sub **DeleteWord**([*lCount* as long = 1])
  Deletes *lCount* words at the cursor position. *lCount* is optional and defaults to 1.

- Sub **EndOfBuffer**([*bSelecting* as boolean = False])
  Moves the cursor to the end of the buffer. If *bSelecting* is True, the text is selected. *bSelecting* is optional and defaults to False.

- Sub **EndOfField**([*bSelecting* as boolean = False])
  Moves the cursor to the end of the current field, which is the current tag. If *bSelecting* is set to True, the text is selected. *bSelecting* is optional and defaults to False.

- Function **Find**(*strString* as string [, *bCaseSensitive* as boolean = False] [, *bLineOnly* as boolean = False]) as boolean
  Searches for *strString* in the current title and marks the string if found. If *bCaseSensitive* is set to True, the search is case sensitive. If *bLineOnly* is set to True, the search is performed in the current line only. *bCaseSensitive* is optional and defaults to False. *bLineOnly* is optional and defaults to False.

- Function **FindTag**(*strTag* as string [, *nOccurrence* as number = 0] [, *bIncludeTag* as boolean = True] [,*bMoveToPosition* as boolean = False]) as string
  Returns the content of tag *strTag* with occurrence *nOccurrence*. If *bIncludeTag* is

set to True, the tag is included in the returned string. If *bMoveToPosition* is set to True, the found text is selected. *strTag* may be specified truncated by a question mark.
nOccurrence is optional and defaults to 0. *bIncludeTag* is optional and defaults to True. *bMoveToPosition* is optional and defaults to False.
The following example will return the content of the first tag starting with 47, including the tag and selecting the text found:

```
FindTag("47?", 0, True, True)
```

- Function **GetCurrentField**() as string
  Returns the current field, i.e. the current tag plus contents at the current cursor position.

- Function **GetCurrentLine**() as long
  Returns the index of the current line.

- Function **GetSelection**() as string
  Returns the currently selected text.

- Function **GetTag**() as string
  Returns the tag of the line where the cursor is positioned.

- Function **GetTagAndSelection**() as string
  Returns the selected text prefixed by the tag of where the selection occurs.

- Sub **InsertText**(*strText* as string)
  Inserts *strText* at the cursor position. If there is a selection, the text is replaced.

- Sub **JoinLines**()
  Joins the current line with the following.

- Sub **LineDown**([*lCount* as long = 1] [, *bSelecting* as boolean = False])
  Moves the cursor *lCount* lines down. If *bSelecting* is set to True, the text is selected. *lCount* is optional and defaults to 1. *bSelecting* is optional and defaults to False.

- Sub **LineUp**([*lCount* as long = 1] [, *bSelecting* as boolean = False])
  Moves the cursor *lCount* lines up. If *bSelecting* is set to True, the text is selected. *lCount* is optional and defaults to 1. *bSelecting* is optional and defaults to False.

- Sub **PageDown**([*lCount* as long = 1] [, *bSelecting* as boolean = False])
  Moves the cursor *lCount* pages down. If *bSelecting* is set to True, the text is selected. *lCount* is optional and defaults to 1. *bSelecting* is optional and defaults to False.

- Sub **PageUp**([*lCount* as long = 1] [, *bSelecting* as boolean = False])
  Moves the cursor *lCount* pages up. If *bSelecting* is set to True, the text is selected. *lCount* is optional and defaults to 1. *bSelecting* is optional and defaults to False.

- Sub **Replace**(*strReplace* as string)
  Replaces the selected text with *strReplace*.

- Sub **ReplaceAll**(*strSearch* as string, *strReplace* as string, *bCaseSensitive* as boolean)
  Replaces all occurrences of *strSearch* by *strReplace*. If *bCaseSensitive* is set to True, the search is performed case sensitive.

- Sub **SetSelection**(*lStart* as long, *lEnd* as long [, *bScrollToPosition* as boolean = False])

Sets the current selection to begin at the character position specified by *lStart* and to end at the character position specified by *lEnd*. If *bScrollToPosition* is set to True, the selection is scrolled into view. *bScrollToPosition* is optional and defaults to False.

- Sub **StartOfBuffer**([*bSelecting* as boolean = False])
  Moves the cursor to the start of the buffer. If *bSelecting* is set to True, the text is selected. *bSelecting* is optional and defaults to False.

- Sub **StartOfField**([*bSelecting* as boolean = False])
  Moves the cursor to the start of the field, i.e. the current tag. If *bSelecting* is set to True, the text is selected. *bSelecting* is optional and defaults to False.

- Sub **WordLeft**([*lCount* as long = 1] [, *bSelecting* as boolean = False])
  Moves the cursor *lCount* words to the left. If *bSelecting* is set to True, the text is selected. *lCount* is optional and defaults to 1. *bSelecting* is optional and defaults to False.

- Sub **WordRight**([*lCount* as long = 1] [, *bSelecting* as boolean = False])
  Moves the cursor *lCount* words to the right. If *bSelecting* is set to True, the text is selected. *lCount* is optional and defaults to 1. *bSelecting* is optional and defaults to False.

A.7. Messages collection

A.7.1. Properties

- **Count** as long (read only)
  Specifies the number of items contained in this collection.

- **Item**(*lIndex* as long) as Message (read only)
  Returns the Message object at index *lIndex* contained in this collection. Refer to chapter A.8 for a description of the properties of the Message object.

A.7.2. Example

The following example shows how to use both properties of the Messages collection of the Application.ActiveWindow:

```
Sub ShowFirstMessage()
   If Application.ActiveWindow.Messages.Count > 0 Then
      MsgBox Application.ActiveWindow.Messages.Item(1).Text
   End If
End Sub
```

A.8. Message object

A.8.1. Properties

- **Text** as string
  Contains the text of the message.

- **Type** as integer
  Contains the type of the message.

A.8.2. Example

The following example shows how to access all messages in the Application.ActiveWindow.Messages collection:

```
Sub ShowMessages()
   Dim msg
   For Each msg In Application.ActiveWindow.Messages
      MsgBox msg.Text
   Next
End Sub
```

A.9. Form Object

WinIBW supports the concept of forms. This is very similar to the form concept of VisualBasic or VisualBasic for Applications. Forms are basically windows, in which you can place controls or ActiveX controls. A form contains its own script and can react on events from or call functions in WinIBW. For example, the current table function in WinIBW is implemented as form. You can thus change its behavior or change it in a way you would like it to work.
To edit forms, you can use the Pica Form editor (separate executable). It will assist you in a similar way to the WinIBW script editor in editing and creating scripts of your form.
WinIBW can either show you forms as model or modeless dialog or create a 'normal' MDI child window for them. To allow you displaying of forms, WinIBW implements the global form object.

A.9.1. Properties

- **FormCount** as long (read only)
  Specifies the number of currently active forms in WinIBW.

A.9.2. Methods

- Function **CreateForm**(*strFileName* as string) as object
  Creates the form specified by *strFileName* and displays it in a MDI child window. You can invoke methods and access properties of the form via the object returned from this function

- Function **CreateFormDlg**(*strFileName* as string) as object
  Creates the form specified by *strFileName* and displays it as a modeless dialog. You can invoke methods and access properties of the form via the object returned from this function

- Function **GetForm**(*Key* as variant) as object
  Returns the object of the form specified by *strFileName*. *Key* refers to the filename of the form.

- Function **GetOrCreateForm**(*strFileName* as string) as object
  Returns the object for the form referred to by *strFileName*. If the form is already created, you simply get the object of it. If the form is not created, it will be, and you get the form object

- Function **IsFormCreated**(*strFileName* as string) as boolean
  Returns True if the form referred to by *strFileName* is created in WinIBW, False otherwise.

Appendix B. Standaard script

B.1. Inleiding

De standaard scripts die bij WinIBW 2.000 meegeleverd worden, bevatten een aantal routines, klassen en objecten die ook in uw eigen scripts aangeroepen kunnen worden. In deze appendix worden deze componenten nader beschreven.

Het gebruik van onderdelen uit de standaard scripts in uw eigen scripts veronderstelt een zekere basiskennis in het (objectgeoriënteerde) programmeren van Visual Basic scripts. Zie voor een uitgebreide inleiding in VB-script en een volledige referentie de Microsoft website http://msdn.microsoft.com/scripting/.

N.B. Het wordt ten sterkste afgeraden de standaard scripts zelf aan te passen, aangezien deze bij elke installatie van WinIBW worden overgeschreven met de meegeleverde versie, en op dat moment uw wijzigingen verloren zullen gaan.

In het standaard script van WinIBW worden de volgende klassen gedefinieerd:

- PicaSiteMap — deze klasse omvat diverse site-specifieke instellingen en commando's.
- PicaTranslate — deze klasse verzorgt het vertalen van teksten.
- PicaLink — deze klasse bevat de functionaliteit voor automatisch linken.
- PicaCreate — deze klasse bevat de functionaliteit voor het invoeren van records (Abes specifiek).
- PicaKillAndReplace — deze klasse bevat functionaliteit voor het verwijderen van records, en het corrigeren van links.
- PicaSetHelp — deze klasse implementeert een iterator die een bepaalde functie uitvoert voor iedere titel in een set.

Ieder van deze klassen bevat een aantal functies en/of procedures die ook buiten de standaard scripts gebruikt kunnen worden. Deze functies en procedures worden in de volgende paragrafen beschreven.

Er zijn een viertal voorgedefinieerde objecten beschikbaar, elk van één van de hiervoor genoemde klassen:

- PicaSite — geinitialiseerd object van de klasse PicaSite.
- PicaTranslator — geinitialiseerd object van de klasse PicaTranslator.
- PicaLinker — niet-geinitialiseerd object ('nothing') van de klasse PicaLinker.
- PicaKiller — niet-geinitialiseerd object ('nothing') van de klasse PicaKiller.

Tenslotte zijn er een aantal scripts (in de vorm van VB subroutines) beschikbaar, die via de WinIBW interface door de gebruiker kunnen worden aangeroepen. De belangrijkste daarvan zijn:

- PicaSearchLink — routine om links te zoeken (niet exact).
- PicaSearchLinkExact — routine om exacte links te zoeken.
- PicaPasteLink — routine om gevonden ppn's in de titel te plakken.
- PicaCopyRecord — routine om een record te kopiëren.
- PicaSetSite — routine om de instelling voor een bepaalde site te initialiseren.

Er is tevens een aantal routines aanwezig die (nog) niet van toepassing zijn op de Nederlandse situatie. Deze zullen hier verder niet besproken worden.

B.2. Class PicaSiteMap

Deze klasse omvat diverse site-specifieke instellingen en commando's.

Alle specifieke instellingen worden gezet in `Class_Initialize()`, en bewaard in *private members* van de klasse.
De site wordt geconfigureerd tijdens de installatie van WinIBW, en wordt opgehaald uit de registerinstellingen in HKEY_LOCAL_MACHINE tijdens het uitvoeren van WinIBW.

Het globale object **`PicaSite`** wordt tijdens het starten van WinIBW gecreëerd, en kan gebruikt worden in gebruikersscripts.

De globale functie **`PicaSetSite`**`(strSite)` is beschikbaar om een andere site in te stellen tijdens het uitvoeren van WinIBW. Dit is noodzakelijk wanneer een gebruiker aanlogt bij een andere site, en vervolgens de standaard scripts wil gebruiken.

De klasse **PicaSiteMap** bevat de volgende *public methods*:

- sub **Command**(*strCommand* as string, *bNewWindow* as boolean)
- sub **CopyRecord**()
- sub **InsertLink**(*strDocType* as string, *strPPN* as string)
- sub **RequestKill**(*strPPN* as string)
- sub **InsertRequiredTags**()
- function **HasNoviceMode**() as boolean
- function **GetFormat**() as string
- function **BuildLinkCommand**(*bExact* as boolean, *strDocType* as string, *strSearchTerm* as string) as string
- function **GetEditDocType**() as string
- function **IsAuthority**(*strDocType* as string) as boolean

B.2.1. Method Command

Subroutine om site-specifieke commando's uit te voeren.

Definitie

sub **Command**(*strCommand* as string, *bNewWindow* as boolean)

strCommand	Commando dat uitgevoerd moet worden.
bNewWindow	Boolean die aangeeft of het commando in een nieuw venster moet worden uitgevoerd.

Opmerkingen
Momenteel worden de volgende commando's ondersteund:

- cmdShowFull — Toon een lange presentatie.
- cmdInsertTitle — Creëer een nieuwe titel.
- cmdInsertAuthority — Creëer een nieuw thesaurus record.
- cmdSwitchFromExternal — Wissel vanuit een extern systeem
- cmdEditRecord — Muteer een record.

Voorbeeld
Dit voorbeeld is een fragment uit de *method* CopyRecord(), beschreven in de volgende paragraaf, waarin het commando `cmdShowFull` wordt uitgevoerd, gevolgd door een catalogiseercommando:

```
Dim strDocType

Command "cmdShowFull", False
```

```
Application.ActiveWindow.CopyTitle
strDocType = Application.ActiveWindow.DocType
If IsAuthority(strDocType) Then
   Command "cmdInsertAuthority", False
Else
   Command "cmdInsertTitle", False
End If
Application.ActiveWindow.PasteTitle
```

B.2.2. Method CopyRecord

Subroutine om een record te kopiëren.

Definitie

sub **CopyRecord**()

Opmerkingen
Kopieert een geselecteerde titel vanuit het presentatiescherm naar een catalogiseerscherm.

Voorbeeld
Dit voorbeeld toont hoe de globale subroutine PicaCopyRecord() is geimplementeerd d.m.h. aanroepen van CopyRecord() voor het globale object PicaSite:

```
Sub PicaCopyRecord()
   PicaSite.CopyRecord
End Sub
```

B.2.3. Method InsertLink

Subroutine om een link PPN in te voegen.

Definitie

sub **InsertLink**(*strDocType* as string, *strPPN* as string)

strDocType	Document type (materiaal code) van het record.
strPPN	PPN waarnaar verwezen wordt.

Voorbeeld
Deze methode wordt bijvoorbeeld aangeroepen in de PicaLinker methode PasteLink():

```
Dim strDocType
Dim strPPN

strDocType = PicaSite.GetEditDocType()
strPPN = Application.ActiveWindow.Variable("P3GPP")
Application.ActivateWindow LinkWindow
'delegate to site object to insert link
PicaSite.InsertLink strDocType, strPPN
```

B.2.4. Method RequestKill

Subroutine om een titel op de kill-nominatie te plaatsen.

Definitie

sub **RequestKill**(*strPPN* as string)

strPPN	PPN van de voorkeurstitel waarnaar verwezen wordt in kenmerk 1699..

Voorbeeld

```
Dim strCorrectPPN

' retrieve the value of strCorrectPPN..
' and insert tag 1699 into the current title:
PicaSite.RequestKill(strCorrectPPN)
```

B.2.5. Method InsertRequiredTags

Subroutine die ervoor zorgt dat noodzakelijke kenmerken aanwezig zijn in de titel.

Definitie

sub **InsertRequiredTags**()

Opmerkingen
Deze routine voegt kenmerk “4700 .” in om een titel gevalideerd te kunnen krijgen.

Voorbeeld
In dit voorbeeld worden de noodzakelijke kenmerken toegevoegd, vlak voordat het record naar de validatieserver wordt gestuurd:

```
PicaSite.InsertRequiredTags
Application.ActiveWindow.SimulateIBWKey "FR"
```

B.2.6. Method HasNoviceMode

Functie die aangeeft of Novice Mode / Coded Data wordt ondersteund voor deze *site*.

Definitie

function **HasNoviceMode**() as boolean

Opmerkingen
Novice Mode / Coded Data wordt momenteel alleen voor Abes in Frankrijk ondersteund, en nog niet voor de overige sites (Nederland en Duitsland).

Voorbeeld

```
If PicaSite.HasNoviceMode() then
   MsgBox “This must be Abes!”
Application.ActiveWindow.CodedData = True
End If
```

B.2.7. Method GetFormat

Functie die het huidige formaat voor lange presentaties (en catalogiseren) teruggeeft, zoals ingesteld bij de installatie van WinIBW.

Definitie

function **GetFormat**() as string

Opmerkingen

Deze functie geeft een site-specifieke standaard waarde terug (in Nederland: ‘D’), en niet de eventueel door de gebruiker ingestelde waarde.

Voorbeeld
```
If UCase(PicaSite.GetFormat()) <> "UNM" Then
   MsgBox PicaTranslator("strNotImplFormat")
   Exit Sub
End If
```

B.2.8. Method BuildLinkCommand

Functie om het link commando (LNK) te construeren.

Definitie

function **BuildLinkCommand**(*bExact* as boolean, *strDocType* as string, *strSearchTerm* as string) as string

bExact	Boolean die aangeeft of er exact gezocht moet worden.
strDocType	Document type (materiaal code) van het record.
strSearchTerm	Zoek term voor het LNK commando.

Opmerkingen
Deze functie geeft het opgemaakte commando terug.

Voorbeeld
Dit voorbeeld is een fragment uit de implementatie van de PicaLink methode SearchLink():

```
strCommand = PicaSite.BuildLinkCommand(bExact, strDocType, _
     Application.ActiveWindow.Title.GetTagAndSelection())
Application.ActiveWindow.Command strCommand, True
```

B.2.9. Method GetEditDocType

Functie om het document type (materiaal code) te achterhalen van een nieuwe titel.

Definitie

function **GetEditDocType**() as string

Opmerkingen
Deze functie geeft het documenttype (materiaal code) terug zoals gevonden in het muteerscherm, of een lege string in geval het documenttype niet vastgesteld kon worden. Deze functie gaat ervan uit dat het catalogiseerformaat overeenkomt met het default formaat dat voor de site is gedefinieerd (en dat opgehaald kan worden met GetFormat()).

Voorbeeld
```
Dim strDocType

strDocType = PicaSite.GetEditDocType()
If IsAuthority(strDocType) Then
   Command "cmdInsertAuthority", False
Else
   Command "cmdInsertTitle", False
End If
```

B.2.10. Method IsAuthority

Functie om te bepalen of een titel een thesaurus record is, aan de hand van het opgegeven document type (materiaal code).

Definitie

function **IsAuthority**(*strDocType* as string) as boolean

strDocType	Document type (materiaal code) van het record.

Opmerkingen
Deze functie geeft TRUE terug wanneer het document type van een thesaurus record is.

Voorbeeld
Zie vorige paragraaf.

B.3. Class PicaTranslate

Deze klasse verzorgt het vertalen van voorgedefinieerde teksten.

Alle specifieke teksten worden vastgesteld in Class_Initialize als *private members* van de klasse.

De gebruikte taal wordt geconfigureerd tijdens installatie van WinIBW, en wordt opgehaald uit de registerinstellingen in HKEY_LOCAL_MACHINE tijdens het uitvoeren van WinIBW.

Het globale object **`PicaTranslator`** wordt tijdens het starten van WinIBW gecreëerd, en kan gebruikt worden in gebruikersscripts.

De klasse **PicaTranslate** bevat de volgende *public methods*:

- default function **Translate** (*strMessage* as string) as string
- function **TranslateFmt** (*strMessage* as string, *dicReplace* as dictionary) as string

Ondersteunde taalcodes zijn:

- NE Nederlands
- DU Duits
- FR Frans
- EN Engels

Met deze klasse kunnen een aantal voorgedefinieerde teksten worden getoond aan de gebruiker, in de taal die op dat moment gebruikt wordt. Het is niet mogelijk teksten aan de interne *dictionaries* toe te voegen, waardoor het nut van deze klasse enigszins beperkt blijft.

B.3.1. Method Translate

Functie om voorgedefinieerde teksten te vertalen.

Definitie

default function **Translate** (*strMessage* as string) as string

strMessage	Identificatie van een tekst die vertaalt moet worden (zie hieronder voor de mogelijkheden).

Opmerkingen
De volgende teksten worden in vier talen geinitialiseerd, en kunnen worden aangeroepen m.b.v. `Translate()`:

- "strSearchLinkImpossible" "Functie F9 kan nu niet gebruikt worden!"
- "strPasteImpossible" "Functie F11 kan nu niet gebruikt worden!"
- "strCorrectPPNRequired" "U dient eerst het correcte PPN te selecteren!"
- "strWrongPPNRequired" "U dient eerst het incorrecte PPN te selecteren!"
- "strMustBeEditing" "U dient in een bewerkingsscherm te staan om deze functie te kunnen gebruiken!"
- "strMustHaveSelection" "U dient de tekst waarnaar de link moet verwijzen, te selecteren!"
- "strMustHaveSearched" "U dient eerst de functie 'Selectie om te relateren' te gebruiken!"
- "strMustHaveFound" "Functie F11 kan nu niet gebruikt worden!"
- "strKillTitle" "Verwijderverzoek"
- "strReplaceTitle" "Corrigeren van links"
- "strLinkTitle" "Selectie om te relateren"

- "strNoDocType" "Het is niet mogelijk om de materiaalsoort van dit record te bepalen!"
- "strNotImplFormat" "Niet geschikt voor dit formaat!"
- "strGetPPNFailed" "Het is niet mogelijk om een PPN voor dit record te selecteren!"
- "strPPNCorrectSelected" "Correct PPN geselecteerd : "
- "strPPNIncorrectSelected" "Incorrect PPN geselecteerd : "
- "strMustBeOnSet" "U dient in een korte presentatie te staan om deze functie te gebruiken!"
- "strTitlesProcessed" " verwerkte titels"
- "strAuthoritiesIgnored" " overgeslagen ingangen"
- "strReplacePrompt" "Deze functie vervangt de link naar PPN '%OLDPPN%' door een link naar PPN '%NEWPPN%' in alle titels van set %SET%" met %SETSIZE% records."
  "De ingangen binnen de set blijven ongewijzigd."
  "Kies 'OK' om door te gaan of 'Annuleren' om te stoppen."

Voorbeeld
Merk op hoe in dit voorbeeld gebruik is gemaakt van het feit dat Translate() de ***default*** methode is van de klasse PicaTranslate:

```
If strDocType = "" Then
MsgBox PicaTranslator("strNoDocType"), vbOKOnly, _
   PicaTranslator("strLinkTitle")
   Exit Sub
End If
```

B.3.2. Method TranslateFmt

Functie om voorgedefinieerde, geparametriseerde teksten te vertalen.

Definitie

function **TranslateFmt** (*strMessage* as string, *dicReplace* as dictionary) as string

strMessage	Identificatie van een tekst die vertaalt moet worden (zie vorige paragraag voor de mogelijkheden), met parameters van de vorm %NAAM%.
dicReplace	Dictionary met vervangende teksten voor de parameters %NAAM%.

Voorbeeld

```
Dim dicMessage
Set dicMessage = CreateObject("Scripting.Dictionary")
dicMessage.Add "%OLDPPN%", strKillPPN
dicMessage.Add "%NEWPPN%", strCorrectPPN
dicMessage.Add "%SET%", strSet
dicMessage.Add "%SETSIZE%", strCount
strPrompt = PicaTranslator.TranslateFmt("strReplacePrompt", dicMessage)
```

B.4. Class PicaLink

Deze klasse bevat de functionaliteit voor automatische thesaureerfunctie.

Het globale object **`PicaLinker`** wordt tijdens het starten van WinIBW gecreëerd (maar niet geinitialiseerd), en kan gebruikt worden in gebruikersscripts.

De klasse **PicaLink** bevat de volgende *public methods*:

- sub **SearchLink** (*bExact* as boolean)
- sub **PasteLink** ()
- function **IsValid** () as boolean

B.4.1. Method SearchLink

Subroutine om links te zoeken.

Definitie

sub **SearchLink** (*bExact* as boolean)

bExact	Boolean die aangeeft of er exact moet worden gezocht.

Opmerkingen
De functie IsValid() kan worden gebruikt om te kijken of deze functie succesvol is geweest.

Voorbeeld
Dit voorbeeld toont hoe deze methode wordt gebruikt in de implementatie van de globale subroutine PicaSearchLink();

```
Sub PicaSearchLink()
   Set PicaLinker = nothing
   Set PicaLinker = new PicaLink
   If Not PicaLinker Is Nothing Then
      PicaLinker.SearchLink FALSE
   End If
End Sub
```

B.4.2. Method PasteLink

Subroutine om het PPN van het selecteerde record te plakken in de titel.

Definitie

sub **PasteLink** ()

Opmerkingen

Voorbeeld
Dit fragment toont hoe deze methode wordt gebruikt in de implementatie van de globale subroutine PicaPasteLink();

....

```
   If Not PicaLinker.IsValid() Then
      MsgBox PicaTranslator("strMustHaveFound"), vbOKOnly, _
                           PicaTranslator("strLinkTitle")
      Set PicaLinker = nothing ' hasta la vista, baby
      Exit Sub
```

```
   End If
   PicaLinker.PasteLink
   Set PicaLinker = nothing
End Sub
```

B.4.3. Method IsValid

Geeft aan of het zoeken succesvol is geweest.

Definitie

function **IsValid** () as boolean

Opmerkingen
Na het plakken geeft deze functie altijd FALSE terug.

Voorbeeld
Zie vorige paragraph.

B.5. Class PicaKillAndReplace

Deze klasse bevat functionaliteit voor het verwijderen van records, en het corrigeren van links.

Het globale object **`PicaKiller`** wordt tijdens het starten van WinIBW gecreëerd (maar niet geinitialiseerd), en kan gebruikt worden in gebruikersscripts.

De klasse **PicaKillAndReplace** bevat de volgende *public methods*:

- sub **SelectCorrectPPN** ()
- sub **SelectIncorrectPPN** ()
- sub **RequestKill** ()
- sub **ReplacePPNs** ()

PicaKillAndReplace maakt gebruik van de klasse PicaSiteMap om site-specifieke instellingen te verbergen.

B.5.1. Method SelectCorrectPPN

Subroutine om het correcte PPN te selecteren.

Definitie

sub **SelectCorrectPPN** ()

Opmerkingen
Het PPN wordt uit de Pica3 context variabele “P3GPP” gehaald, en opgeslagen in een *private member* van deze klasse.

Voorbeeld

```
Sub PicaSelectCorrectPPN()
   If PicaKiller Is Nothing Then
      Set PicaKiller = New PicaKillAndReplace
   End If
   PicaKiller.SelectCorrectPPN
End Sub
```

B.5.2. Method SelectIncorrectPPN

Subroutine om het incorrecte PPN te selecteren.

Definitie

sub **SelectIncorrectPPN** ()

Opmerkingen
Het PPN wordt uit de Pica3 context variabele “P3GPP” gehaald, en opgeslagen in een *private member* van deze klasse.

Voorbeeld

```
Sub PicaSelectCorrectPPN()
   If PicaKiller Is Nothing Then
      Set PicaKiller = New PicaKillAndReplace
   End If
   PicaKiller.SelectIncorrectPPN
End Sub
```

B.5.3. Method RequestKill

Subroutine om een titel op de kill-nominatie te plaatsen.

Definitie

sub **RequestKill** ()

Opmerkingen
Deze routine roept de routine RequestKill() van de klasse PicaSiteMap aan (via het globale object PicaSite) om het eigenlijke werk te doen. Het ingevoegde PPN moet eerder door SelectCorrectPPN() zijn opgehaald.

Voorbeeld
```
Sub PicaRequestKill()
   PicaKiller.RequestKill
   Set PicaKiller = Nothing
End Sub
```

B.5.4. Method ReplacePPNs

Subroutine om PPN links te corrigeren voor een gehele set.

Definitie

sub **ReplacePPNs** ()

Opmerkingen

Voorbeeld
```
Sub PicaCorrectLinks()
   PicaKiller.ReplacePPNs
   Set PicaKiller = Nothing
End Sub
```

B.6. Class PicaSetHelp

Deze klasse implementeert een iterator die een gespecificeerde functie uitvoert voor iedere titel in een set.

De uit te voeren functie dient de volgende vorm te hebben:

function **YourFunction**(*strSet* as string, *nTitle* as integer) as boolean

Wanneer de uit te voeren functie FALSE teruggeeft, stopt de iteratie.

De klasse **PicaSetHelp** bevat de volgende *public method*:

- default sub **Process** ()

B.6.1. Method Process

Subroutine die een gespecificeerde functie uitvoert voor iedere titel in een set.

Definitie

default sub **Process** (*strFuncName* as string)

strFuncName	De naam van de aan te roepen functie.

Opmerkingen
De uit te voeren functie dient de volgende vorm te hebben:

function **YourFunction**(*strSet* as string, *nTitle* as integer) as boolean

strSet	Het setnummer (als string!).
nTitle	Het volgnummer van de titel binnen de set.

Wanneer de uit te voeren functie FALSE teruggeeft, stopt de iteratie.

Voorbeeld

```
Function DoSomething(strSet, nTitle)
   …
   If some_condition Then
      DoSomething = True
   Else
      DoSomething = False
   End If
End Function

Dim Iterator
Set Iterator = New PicaSetHelp

Iterator.Process(“DoSomething”)
```

B.7. Globale subroutines

Gebaseerd op de hiervoor beschreven klassen en bijbehorende methoden, zijn er een aantal globale subroutines gedefinieerd, die tevens beschikbaar zijn in het interface van WinIBW zelf.

Ze kunnen ook gebruikt worden in eigen VB scripts.

B.7.1. Sub PicaSearchLink

Routine om links te zoeken (niet exact).

Definitie

sub **PicaSearchLink**()

Opmerkingen
Deze routine gebruikt het globale object PicaLinker, om het eigenlijke werk te doen, en om tijdelijke gegevens te kunnen overdragen aan de routine PicaPasteLink().

B.7.2. Sub PicaSearchLinkExact

Routine om exacte links te zoeken.

Definitie

sub **PicaSearchLinkExact**()

Opmerkingen
Deze routine gebruikt het globale object PicaLinker, om het eigenlijke werk te doen, en om tijdelijke gegevens te kunnen overdragen aan de routine PicaPasteLink().

B.7.3. Sub PicaPasteLink

Routine om gevonden ppn's in de titel te plakken.

Definitie

sub **PicaPasteLink**()

Opmerkingen
Deze routine gebruikt hetzelfde globale object PicaLinker dat ook door PicaSearchLink() en PicaSearchLinkExact() gebruikt is, om het eigenlijke werk te doen, en om tijdelijke gegevens te kunnen overnemen van de zoek-routine.

B.7.4. Sub PicaCopyRecord

Routine om een record te kopiëren.

Definitie

sub **PicaCopyRecord**()

Opmerkingen

Deze routine gebruikt het globale object PicaSite, om het eigenlijke werk te doen, m.b.v. de *method* CopyRecord().

B.7.5. Sub PicaSetSite

Routine om de instelling voor een bepaalde site te initialiseren.

Definitie

sub **PicaSetSite**()