C4B COM FOR BUSINESS

DASHBOARD SCRIPT LIBRARY

FÜR XPHONE CONNECT

VERSION 1.9

STAND 12.02.2025

Inhaltsverzeichnis

ÜBERBLICK

Das XPhone Dashboard bietet eine einfache Möglichkeit zur Ausführung von Script-Code im Kontext des Microsoft Windows Scripting Hosts (WSH). Die bevorzugte Programmiersprache dafür ist Visual Basic Script (VBS).

Lesen Sie dazu eine Einführung in der XPhone Connect Dokumentation: https://help.c4b.com/xphone-connect9/doc/de/admin/config/dash/dsh-scrpt.html

Die hybriden Dashboard Scripts bestehen aus einem BATCH Prolog und dem eigentlichen WSF Code (Visual Basic). Der BATCH Prolog muss normalerweise nie angepasst werden. Die gesamte Script-Funktionalität findet im unteren WSF Block statt.

Hybrides Batch Script

```
<!-- : Begin BATCH Part -----------------------------------
@echo off
cscript //nologo "%~f0?.wsf" %1
Rem cscript //nologo //D //x "%~f0?.wsf" ^""%1"^" 'debug version exit /b

----- Begin WSF Part--------------------------------------->
<job><script language="VBScript">
    MsgBox "Hello Dashboard Script!"
</script></job>
```

In den meisten Scripts gibt es immer wiederkehrende Aufgaben zu erledigen:

- Auswertung von Parametern, die an das Script übergeben wurden
- Zugriff auf Windows-Umgebungsvariable
- Interaktion mit dem Windows Betriebssystem ("Shell")
- Auswertung des aktuellen Anrufer-Status (ein-/ausgehend, intern/extern, ringing/connected, etc.)
- Ablaufverfolgung des Scripts als Logfile oder im Eventlog (Tracing, Logging)
- Fehlerbehandlung

Um diese Aufgaben zu vereinfachen und zu vereinheitlichen, stellt C4B eine Visual Basic Script Library "c4bScriptLibrary.vbx" zur Verfügung, die auf einfache Weise in eigene Dashboard Scripts integriert werden kann.

Hinweis: Aus Sicherheitsgründen wird anstelle der Endung "*.vbs" die unverfängliche Endung "*.vbx" verwendet.

Zentraler Einstiegspunkt für die Verwendung der Script-Library ist die globale Variable "xps", die automatisch zur Verfügung gestellt wird.

Dashboard Script Library

```
'=====================================================================
' C4B Script Library (für VBS Dashboard Scripts)
' Version 1.8
'=====================================================================
'Globale Script-Variable "xps" für den Zugriff auf die Dashboard-Library
Dim xps
Set xps = New XPhoneScript
'----------------- CLASS XPhoneScript (General Script Class) ------------
Class XPhoneScript
```

```
  '... Implementierung der Dashboard-Library - hier weggelassen.
End Class
```

Befindet sich die Library im selben Verzeichnis wie die Script-Datei, wird sie über den Include-Mechanismus des Windows Scripting Hosts mit Hilfe des "src"-Attributs in das eigene Script eingebunden. Ab dann steht die globale Variable "xps" zur Verfügung:

Batch Script mit Script Library

```
<job><script language="VBScript" src="c4bScriptLibrary.vbx">
  xps.DumpContext("Contact")
</script></job>
```

Auf diesem Objekt "xps" können nun alle öffentlichen Methoden der Klasse aufgerufen werden. Im obigen Beispiel ist das die Methode "DumpContext" mit einem Parameter, der angibt, welche Teilmenge der Dashboard-spezifischen Umgebungs-Variablen als Message-Box angezeigt werden sollen. Im Beispiel oben sind das alle Attribute eines Kontakts aus einer Kontaktsuche im XPhone Connect Directory.

In allen folgenden Beschreibungen bezeichnet xps immer ein instanziiertes XPhone-Scripting-Objekt:

Set xps = new XPhoneScript

WINDOWS UMGEBUNG

Die Property Shell ist eine Instanz des Objekts "WScript.Shell" und bietet direkten Zugriff auf dessen Methoden und Eigenschaften. Die Dokumentation dazu findet man im Internet, z.B. hier: https://www.informit.com/articles/article.aspx?p=1187429&seqNum=5

XPS.SHELL

Beispiel: Notepad öffnen

```
' Ausführen des Programms "Notepad.Exe"
xps.Shell.Run "Notepad.Exe"
```

JSON PARSER

XPS.JSONPARSER(JSON)

Diese Methode akzeptiert einen JSON-String als Parameter und liefert ein VBScipt-Objekt zurück, mit dem man bequem auf die JSON-Struktur zugreifen kann.

Beispiel: json String als Objekt erzeugen

```
' json kommt z.B. von einem REST API Call
json = "{""id"":""3rdpartyID"",""type"":""anyType""}"

Set parsed = xps.JsonParser(json)
myID = parsed("id")
xps.Log.DEBUG "Verwende ID = " & myID
```

UMGEBUNGSVARIABLEN

Innerhalb eines Dashboard-Scripts stehen eine Reihe von Umgebungsvariablen zur Verfügung, die vom XPhone Connect Client vor dem Aufruf des Scripts in dessen "Process"-Kontext hinterlegt werden. Welche Variablen das im Einzelnen sind, wird in der XPhone Dokumentation beschrieben (https://help.c4b.com/xphoneconnect-9/doc/de/admin/config/dash/dsh-appndx.html).

Der Zugriff auf diese Variablen wird durch Methoden der Dashboard Script-Library vereinfacht.

XPS.GETCONTEXT(NAME)

Diese Methode greift auf das "Process"-Environment des ausgeführten Scripts zu. Statt xps.GetContext(name) könnte man äquivalent auch schreiben: xps.Shell.Environment("Process")(name).

Beispiel

```
'Zugriff auf die E-Mail-Adresse des aktuellen XPhone Users und Ausgabe als Message-Box
UserName = xps.GetContext("XP.User.EMAIL1")
MsgBox UserName
```

XPS.SETCONTEXT(NAME, VALUE)

Mit dieser Methode können bequem Environment-Variaben im "Process" Kontext gesetzt werden. Zum Beispiel um mit festen Test-Variablen während der Entwicklung des Scripts zu arbeiten.

XPS.GETUSERCONTEXT(NAME)

Diese Methode greift auf das "User"-Environment des ausgeführten Scripts zu. Statt xps.GetUserContext(name) könnte man äquivalent auch schreiben: xps.Shell.Environment("User")(name).

XPS.SETUSERCONTEXT(NAME, VALUE)

Mit dieser Methode können bequem Environment-Variaben im "User" Kontext gesetzt werden. Zum Beispiel um Daten zwischen verschiedenen Dashboard-Scripts auszutauschen.

CALL-STATUS

Mit Hilfe der State Property wird der Zustand des aktuellen Anrufs ermittelt. Alle Methoden liefern entweder "True" oder "False" zurück. Kombinationen sind möglich.

Beispiel

```
' Behandlung eingehender Calls, getrennt nach "signalisierend" und "verbunden"
If ( xps.State.IsIncoming ) Then
  If ( xps.State.IsRinging ) Then
    'Eingehender, signalisierter Call
  ElseIf ( xps.State.IsConnected ) Then
    'Eingehender, verbundener Call
  End If
End If
```

XPS.STATE.ISINCOMING

True: eingehender Call

XPS.STATE.ISOUTGOING

True: ausgehender Call

XPS.STATE.ISRINGING

True: Call wird signalisiert (ist aber noch nicht verbunden)

XPS.STATE.ISCONNECTED
True: Call ist verbunden

XPS.STATE.ISDROPPED
True: Call ist nicht verbunden und nicht signalisiert.

XPS.STATE.ISINTERNAL
True: es handelt sich um einen internen Call

XPS.STATE.ISEXTERNAL
True: es handelt sich um einen externen Call

XPS.STATE.ISSOFTPHONE
True: es handelt sich um einen Softphone-Call

XPS.STATE.ISCONFERENCE
True: es handelt sich um einen Konferenz-Call

XPS.STATE.ISCONFERENCEDIALIN
True: es handelt sich um einen eingehenden Konferenz-Call

XPS.STATE.ISANYDEVICE
True: es handelt sich um einen AnyDevice-Call

PROTOKOLLIERUNG

Die Log Property ermöglicht eine einfache Protokollierung in einer Text-Datei für das gerade ausgeführte Script. Die Logeinträge landen in einer Text-Datei namens XPhoneConnect_WSH_<DATUM>.txt, wobei <DATUM> durch das tagesaktuelle Datum ersetzt wird. Der Speicherort liegt im Benutzerprofil des angemeldeten

Benutzers:

C:\Users\<USER>\AppData\Local\C4B\Log\XPhoneConnect_WSH_JJJ-MM-TT.txt

Die Log-Einträge werden nur geschrieben, wenn das clientseitige Logging für den Benutzer aktiviert wurde:

Jeder Logeintrag entspricht einer Zeile mit diesem Aufbau:

Datum Uhrzeit Kategorie Script-Name Freitext

Beispiel:

2023-10-12 10:21:22.397 INF MyScript.bat Ab hier steht beliebiger Text...

XPS.LOG.ERROR(MSG)
Mit dieser Methode werden Log-Einträge der Kategorie ERR geschrieben.

XPS.LOG.WARNING(MSG)
Mit dieser Methode werden Log-Einträge der Kategorie WRN geschrieben.

XPS.LOG.SUCCESS(MSG)
Mit dieser Methode werden Log-Einträge der Kategorie SUC geschrieben.

XPS.LOG.INFO(MSG)
Mit dieser Methode werden Log-Einträge der Kategorie INF geschrieben.

XPS.LOG.DEBUG(MSG)
Zeigt eine Message-Box mit dem Inhalt "msg", wenn "DebugMode" auf "True" gesetzt ist.

HELPER METHODEN

XPS.READFROMREGISTRY(KEY, DEFAULT)
Diese Methode liest einen Wert der Windows-Registry zum angegebenen *key* aus. Wird der *key* nicht gefunden, wird der *default* Wert zurückgegeben.

Beispiel

```
' Ist das Logging für den XPhone Connect Client aktiviert?
v = xps.ReadFromRegistry("HKEY_CURRENT_USER\Software\C4B\Log\EnableFileLog", "0")
If ( v = "1" ) Then
  MsgBox "Logging ist aktiviert!"
End If
```

XPS.GETCALLGUID
Liefert eine eindeutige ID für einen signalisierten Anruf. Steht nur während eines "echten" Anrufs zur Verfügung. Im Kontext einer reinen Journal- oder Kontaktanzeige wird ein leerer String zurückgeliefert.

FEHLERBEHANDLUNG

Die Dashboard Script Library bietet einen sehr einfachen Mechanismus für die Fehlerbehandlung. Das Konzept basiert auf der globalen VBScript Err-Variable, deren Property Err.Number solange auf 0 steht, bis ein Fehler aufgetreten ist. Mit der Anweisung "On Error Resume Next" weist man das Script an, beim Auftreten eines Fehlers einfach weiter zu laufen. Das ermöglicht es, zu jedem Zeitpunkt die Property Err.Number abzufragen und geeignet zu reagieren.

Im Dashboard Script steht zu diesem Zweck die Methode HandleError() zur Verfügung.

XPS.HANDLEERROR(INFO)

Achtung: Der Aufruf von HandelError() beendet das Script mit WScript.Quit!

Das folgende Beispiel demonstriert die Fehlerbehandlung im Dashboard Script. Der Fehler wird in der LogDatei protokolliert (inkl. Fehler-Nummer und Fehler-Beschreibung), anschließend wird das Script beendet. HandleError() kann auch aufgerufen werden, wenn KEIN tatsächlicher Fehler aufgetreten ist (z.B. wenn ein unsinniger Zustand oder ungültige Parameter erkannt wurden). Dann beendet sich das Script ebenfalls, nachdem der Fehler-Protokoll-Eintrag geschrieben wurde.

Beispiel

```
' Fehlerbehandlung im Dashboard Script
' Als erste Zeile im Script immer die automatische Fehlerbehandlung abschalten
On Error Resume Next
' Ausführung einer (fehlerhaften) Anweisung
x = 2 / 0
'Fehlerbehandlung
If Err.Number <> 0 Then
  xps.ErrorHandler("Ungültige Division.")
Else
  xps.Log.INFO("Division war gültig!")
End If
' Hier geht es nur weiter, wenn Err.Number immer noch 0 ist ' ...
```

VERBUNDENE SCRIPTS (STARTCALL – ENDCALL)

Eine immer wiederkehrende Anforderung ist es, dass die beiden Scripts für "StartCall" und "EndCall" über gemeinsame Variablen kommunizieren können.

Beispiel: das StartCall-Script startet einen Request an ein externes System. Mit dem EndCall-Script soll diese Aktion abgeschlossen werden. Dafür soll in beiden Aufrufen (häufig mittels eines REST API) eine eindeutige ID übergeben werden.

Die c4bScriptLibrary unterstützt diese Anforderung mit Hilfe von Environment-Variablen im User-Context. Im Gegensatz zu den ansonsten verwendeten Environment-Variablen Process-Context bleiben User-ContextVariablen nach dem Beenden eines Batch-Scripts im Windows-Profil des Benutzers erhalten und können somit von einem folgenden Batch-File wieder abgefragt und weiterverwendet werden.
Die Anforderung der Eindeutigkeit von User-Context-Variablen löst die c4bScriptLibrary durch den Einsatz von GUIDs.

Das gesamte Funktionspaket ist in diese 5 Methoden verpackt:

XPS.CREATEACTIONID()
Erzeugt eine neue, eindeutige ActionID (in Form einer GUID) und speichert sie im User-Context "XP.ACTION.ID".

XPS.SETACTIONID(NEWACTIONID)
Setzt eine vom Aufrufer(!) erzeugte eindeutige ActionID und speichert sie im User-Context "XP.ACTION.ID". Diese Methode wird dann gebraucht, wenn ein Dritt-System (z.B. CATCH CRM) für das Erzeugen der ID verantwortlich ist und nur seine eigenen IDs verwenden kann.

XPS.HASACTIONID()
Liefert "True", wenn eine ActionID im User-Context vorhanden ist.

XPS.GETACTIONID()
Liefert die aktuelle ActionID als String zurück (sofern vorhanden - ggf. vorher mit HasActionID() prüfen).

XPS.REMOVEACTIONID()
Entfernt die aktuelle ActionID aus dem User-Context.

BEISPIEL FÜR VERBUNDENE SCRIPTS
In dem folgenden Beispiel werden aufeinanderfolgende Aufrufe desselben Scripts über eine gemeinsame ActionID verbunden.

Das Script "NotifyService.bat" wird dabei sowohl als StartCall- als auch als EndCall-Script im XPhone Server konfiguriert.

NotifyService.Bat

Dieses Script unterscheidet die Callstates "Ringing", "Connected" und "Dropped". Im Ringing-Status wird eine eindeutige ActionID erzeugt. Im Connected-Status wird diese verwendet und im Dropped-Status wieder

gelöscht. In jedem Status kann die ActionID bei Bedarf als Parameter für einen REST API Call verwendet werden.

NotifyService.bat

```
<!-- : Begin batch script NotifyService.bat
Rem Wenn als Parameter %1 der Wert "Endcall" übergeben wird,
Rem stellt das den CallState auf xps.state.IsEndcall = True

cscript //nologo "%~f0?.wsf" %1
Rem cscript //nologo //D //x "%~f0?.wsf" ^""%1"^" 'debug version

Exit /b
----- Begin wsf script --->

<job><script language="VBScript" src="c4bScriptLibrary.vbx">

'Debugging. Simuliere die Call-States "Connected", "Ringing" bzw. „Dropped“
If WScript.Arguments.Count > 0 Then
  If WScript.Arguments.Item(0) = "Connected" Then
    xps.SetContext "XP.CALL.State", CALLSTATE_CONNECTED
  End If
  If WScript.Arguments.Item(0) = "Ringing" Then
    xps.SetContext "XP.CALL.State", CALLSTATE_RINGING
  End If
  If WScript.Arguments.Item(0) = "Dropped" Then
    xps.SetContext "XP.CALL.State", 0
  End If
End If

If xps.State.IsDropped Then
  xps.RemoveActionID        'ActionID wird entfernt
  If ( xps.HasActionID ) Then
    MsgBox "xps.State.IsDropped: Fehler. RemoveActionID fehlgeschlagen. ActionID noch vorhanden!"
  Else
    MsgBox "xps.State.IsDropped: ActionID erfolgreich entfernt!"   End If

  ElseIf xps.State.IsConnected Then
    'ActionID, die im Status IsRinging erzeugt wurde, wird abgefragt
    'und kann für REST-API Calls etc. verwendet werden.
    If ( xps.HasActionID ) Then
      MsgBox "xps.State.IsConnected: " + xps.GetActionID   Else
      MsgBox "xps.State.IsConnected: Fehler. Keine ActionID gefunden!"
    End If
  Else

    xps.CreateActionID    'Neue, eindeutige ActionID wird erzeugt und im User-Profil abgelegt.
    If ( xps.HasActionID ) Then
      MsgBox "xps.State.IsRinging: Neue ActionID = " + xps.GetActionID   Else
      MsgBox "xps.State.IsRinging: Fehler. ActionID konnte nicht erzeugt werden!"
    End If
  End If

</script></job>
```

Copyright und Rechtliche Hinweise

C4B Com For Business AG

Untere Point 8
82110 Germering | Germany
+49 (89) 840798 - 0
E-Mail: support@c4b.de
Website: www.c4b.com