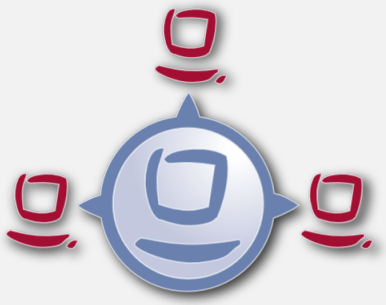

Entwurf: opsi Erweiterung für Roaming Profiles

uib gmbh
Bonifaziusplatz 1b
55118 Mainz
Tel.:+49 6131 275610
www.uib.de
info@uib.de

Stand: 06.09.2011

Inhaltsverzeichnis

1 opsi Erweiterung für Roaming Profiles

1.1 Vorbedingungen für die opsi Erweiterung für Roaming Profiles

Dieses Modul ist momentan eine kofinanzierte opsi Erweiterung.
Es sind eine Reihe von Vorbedingungen nötig, um dieses Modul einsetzen zu können. Das bedeutet, das Sie zum Einsatz eine Freischaltdatei benötigen. Diese Freischaltung erhalten Sie wenn Sie die Erweiterung kaufen. Zu Evaluierungszwecken stellen wir Ihnen auch eine zeitlich befristete Freischaltung kostenlos zur Verfügung (→ mail an info@uib.de).

Technische Voraussetzungen sind opsi 4.0.1 mit den Paketständen:

Tabelle 1: Benötigte Pakete

opsi-Paket	**Version**
opsi-client-agent	>=4.0.1-20
opsi-winst	>=4.11.2.1

1.2 Einführung

Der *opsi-winst* verfügt über eine Reihe von speziellen Befehlen um Modifikationen in Profilen vorzunehmen. Diese Arbeiten aber auf den lokalen Profilen und sind beim Einsatz von *Roaming Profiles* weitgehend nutzlos. Mit der opsi Erweiterung für *Roaming Profiles* wird nun eine Möglichkeit geschaffen auch hier Veränderungen an den Profilen vorzunehmen. Dies geschieht in dem beim Userlogin der *opsi-winst* gestartet wird um spezielle *userLoginScripte* auszuführen.

1.3 Konzept

Wenn die Profile nicht bei der Installation der Software gleich mit gepatcht werden können, muss zwischen dem *Maschinen Teil* und dem *Profil Teil* der Installation deutlicher unterschieden werden. Die kann sowohl innerhalb eines Scriptes geschehen als auch durch die Auslagerung des *Profil Teils* in ein eigenes Script. Vielerorts passiert dies auch jetzt schon, in dem die *Profil Teile* im Rahmen eines Domain Login Scripts ausgeführt werden.

Je nach Praxis liegen daher die *Profil Teile* von opsi-Produkten als Bestandteil der opsi-scripte zur Installation und Deinstalltion vor, als auch als Bestandteil eines Domain Loginscriptes. Ziel dieser Erweiterung ist es, beide Varianten möglichst einfach in den neuen Mechanismus integrieren zu können.

Die Kernkonzepte dieser opsi Erweiterung sind:

- Ausführen spezieller UserLogin Scripte beim Login des users
  Im Rahmen des User Logins wird der *opsi-winst* gestartet aber in einem Speziellem Modus ausgeführt in dem nur bei den Produkten hinterlegte *userLoginScripte* ausgeführt werden.
- Ausführen der Scripte mit administrativen Rechten aber im Userkontext
  Domain Login Scripte werden vom User mit user Rechten ausgeführt. Die opsi *userLoginScripte* werden vom *opsi-winst* ausgeführt, welcher mit administrativen Rechten läuft. Gleichzeitig begibt sich der *opsi-winst* aber in den Kontext des Users der sich eingelogged hat, so dass die Manipulation der Profile mit den selben Befehlen durchgeführt werden kann, wie in einem Domain Loginscript.
- Beschränkung der auszuführenden Scripte auf die tatsächlich notwendigen:
  Logged sich ein Anwender auf einer Maschine ein, so wird ermittelt für welche Produkte gleichzeitig die folgenden Bedinungen erfüllt sind:

  - das Produkt ist auf dem aktuellen Rechner erfolgreich (last result = success) installiert (lastaction = setup) oder deinstalliert (lastaction = uninstall) worden.

  - für das Produkt gibt es eine UserLoginScript

    Nur wenn beide Bedingunen zutreffen, wird das *userLoginScript* auch ausgeführt.

1.4 Neue und erweiterte *opsi-winst* Funktionen

- Aufrufparameter `/loginscripts`
  Wird der *opsi-winst* im opsi-service Kontext mit dem zusätzlichen Parameter */loginscripts* aufgerufen, so hat das im wesentlichen folgende Auswirkungen:

  - Es werden die Produkte ermittelt welche ein *userLoginScript* haben und gleichzeitig auf der aktuellen Maschine installiert oder deinstalliert worden sind. Nur für diese Produkte werden die *userLoginScripte* ausgeführt.
  - Es wird der user der sich eingelogged hat ermittelt und dafür gesorgt, dass die Konstanten zum aktuellen User wie z.B. `%CurrentAppdataDir%` auf die entsprechenden Verzeichnisse des eingeloggten user zeigen. Ebenso werden Registry Operationen (`Registry` Sektionen und `GetRegistryString`) welche sich auf `HKCU` beziehen, so ausgeführt, das die Daten aus dem Registryzweig des Users kommen.

- Aufrufparameter `/silent`
  Der Aufrufparameter `/silent` sorgt dafür, das während der Scriptabarbeitung das Fenster des *opsi-winst* nicht angezeigt wird.

- Funktion `GetScriptMode`
  Um innerhalb eines Scriptes zu unterscheiden in welchem Modus das Script gerade ausgeführt wird, liefert die Funktion `GetScriptMode` drei mögliche Werte zurück:

  - *Machine*
    Das Script wird **nicht** als *userLoginScript* ausgeführt (sondern z.B. als setup oder uninstall Script).
  - *LoginSetup*
    Das Script wird als *userLoginScript* ausgeführt und zwar weil das entsprechende Produkt den Reportstatus *success (setup)* hat, also zuletzt erfolgreich auf dieser Maschine installiert wurde.
  - *LoginUninstall*
    Das Script wird als *userLoginScript* ausgeführt und zwar weil das entsprechende Produkt den Reportstatus *success (uninstall)* hat, also zuletzt erfolgreich auf dieser Maschine deinstalliert wurde.

- Registry Sektionen

  - Registry Sektionen welche auf *HKCU* bzw. *HKEY_CURRENT_USER* arbeiten werden lim Loginscript mode so ausgeführt, dass die Änderungen im Zweig des eingeloggten users landen. Entsprechendes gilt für die Funktionen `GetRegistryStringValue*`.
  - Registry Sektionen welche im Normalen Modus (*Machine*) mit dem Modifier */AllNtUserDats* aufgerufen werden, dürfen jetzt in der `openkey` Anweisung den Root *HKCU* bzw. *HKEY_CURRENT_USER* enthalten. Dies ermöglicht es die selbe Registry Sektion in den unterschiedlichen Modi auszuführen.

- Logging
  Die Logs von userLoginScripten werden geschrieben nach:
  `c:\opsi.org\log\<login user name>_login.log`

1.5 Beispiele von Userloginscripten

Zunächst zwei Beispiele die so aufgebaut sind, wie sie auch in Domainloginscripten eingestzt werden könnten.

Ein sehr einfaches allgemeines Beispiel:

```
[Actions]
Message "Example Profile Patch ...."

Files_profile_copy
Registry_current_user

[Files_profile_copy]
copy "%Scriptpath%\profiles\*.*" "%CurrentAppdataDir%\ACME"

[Registry_current_user]
openkey [HKCU\Software\ACME]
set "show_greeting_window" = "no"
```

Ein Beispiel zur Firefoxkonfiguration:

```
[Actions]
Message "Firefox Profile Patch ...."

DefVar $akt_profile_ini$
DefVar $rel_prefs_path$

comment "check for existing profile ..."
Set $akt_profile_ini$ = "%CurrentAppdataDir%\Mozilla\Firefox\profiles.ini"
if FileExists($akt_profile_ini$)
        Set $rel_prefs_path$ = GetValueFromInifile($akt_profile_ini$,"Profile0","Path","")
        if FileExists("%CurrentAppdataDir%\Mozilla\Firefox\\"+$rel_prefs_path$)
                comment "We found the profile and will now patch it ....."
        endif
else
        comment "no firefox profile found for user"
endif
```

Als nächstes zeigen wir ein Beispiel welches das erste erweitert um die Möglichkeit Dinge aus dem Profil auch wieder zu entfernen. Je nachdem ob das Produkt auf dem Rechner installiert oder deinstalliert wurde bekommt die Funktion `GetScriptMode` einen anderen Wert:

```
[Actions]
Message "Example Profile Patch ...."

if GetScriptMode = "LoginSetup"
        comment "Product is installed"
        Files_profile_copy
        Registry_currentuser_set
endif

if GetScriptMode = "LoginUninstall"
        comment "Product was uninstalled"
        Files_profile_del
        Registry_currentuser_del
endif

[Files_profile_copy]
```

```
copy "%Scriptpath%\profiles\*.*" "%CurrentAppdataDir%\ACME"

[Registry_currentuser_set]
openkey [HKCU\Software\ACME]
set "show_greeting_window" = "no"

[Files_profile_del]
del -s -f "%CurrentAppdataDir%\ACME"

[Registry_currentuser_del]
deletekey [HKCU\Software\ACME]
```

Nun ein Beispiel welches das Setup Skript (setup.ins und delsub.ins) nutzt um unnötige Verdopplung des Codes zu vermeiden:

setup32.ins:

```
[Actions]
requiredWinstVersion >= "4.11.2"

DefVar $MsiId$
DefVar $UninstallProgram$
DefVar $ProductId$
DefVar $InstallDir$

; ----------------------------------------------------------------
; - Please edit the following values                             -
; ----------------------------------------------------------------
Set $ProductId$       = "ACME"
Set $InstallDir$      = "%ProgramFiles32Dir%\ACME"
; ----------------------------------------------------------------
        comment "Show product picture"
        ShowBitmap "%ScriptPath%\\" + $ProductId$ + ".png" $ProductId$

        if FileExists("%ScriptPath%\delsub32.ins")
                comment "Start uninstall sub section"
                Sub "%ScriptPath%\delsub32.ins"
        endif

if GetScriptMode = "Machine"
        Message "Installing " + $ProductId$ + " ..."

        comment "Start setup program"
        Winbatch_install

        comment "Patch the local Profiles ..."
        Registry_currentuser_set /AllNtUserDats
        Files_profile_copy /AllNtUserProfiles
endif

if GetScriptMode = "LoginSetup"
        comment "Product is installed"
        Files_profile_copy
        Registry_currentuser_set
endif


[Winbatch_install]
```

```
"%ScriptPath%\setup.exe" /sp- /silent /norestart

[Files_profile_copy]
copy "%Scriptpath%\profiles\*.*" "%UserProfileDir%\Appdata\ACME"

[Registry_currentuser_set]
openkey [HKCU\Software\ACME]
set "show_greeting_window" = "no"
```

delsub32.ins:

```
Message "Uninstalling " + $ProductId$ + " ..."

if GetScriptMode = "Machine"
        comment "The machine part ..."
        Set $UninstallProgram$ = $InstallDir$ + "\uninstall.exe"
        if FileExists($UninstallProgram$)
                comment "Uninstall program found, starting uninstall"
                Winbatch_uninstall
        endif
        ; does not work: Registry_currentuser_del /AllNtUserDats
        Files_profile_del /AllNtUserProfiles
endif

if (GetScriptMode = "LoginSetup") or (GetScriptMode = "LoginUninstall")
        comment "The profile part ..."
        Files_profile_del
        Registry_currentuser_del
endif

[Winbatch_uninstall]
"$UninstallProgram$" /silent /norestart

[Files_profile_del]
del -s -f "%UserProfileDir%\Appdata\ACME"

[Registry_currentuser_del]
deletekey [HKCU\Software\ACME]
```

Nun eine Variante, welche sich in der Registry merkt ob das Skript für dieses Produkt in dieser Version und diesen User schon mal ausgeführt wurde.

```
[Actions]
DefVar $mytmpreg$
DefVar $myScriptMode$
Message "Example Profile Patch ...."

set $mytmpreg$ = "HKCU\Software\opsi.org\loginscripts"
set $myScriptMode$ = GetScriptMode

comment "Did we run this script before ?"
if not ("%installingProdVersion%" = GetRegistryStringValue32("["+$mytmpreg$+"\\"+$myScriptMode$+"
    ] %installingProdName%"))

        comment "loginscript was not run yet "

        if $myScriptMode$ = "LoginSetup"
```

```
                comment "Product is installed"
                Files_profile_copy
                Registry_currentuser_set
        endif

        if $myScriptMode$ = "LoginUninstall"
                comment "Product was uninstalled"
                Files_profile_del
                Registry_currentuser_del
        endif

        Registry_mark_run
endif

[Files_profile_copy]
copy "%Scriptpath%\profiles\*.*" "%CurrentAppdataDir%\ACME"

[Registry_currentuser_set]
openkey [HKCU\Software\ACME]
set "show_greeting_window" = "no"

[Files_profile_del]
del -s -f "%CurrentAppdataDir%\ACME"

[Registry_currentuser_del]
deletekey [HKCU\Software\ACME]

[Registry_mark_run]
openkey [$mytmpreg$\$myScriptMode$]
set "%installingProdName%" = "%installingProdVersion%"
```

1.6 Konfiguration

Um die Roaming Profiles Erweiterung zu nutzen muss in der Konfiguration des opsiclientd das Loginevent aktiviert werden und dann der *opsi-winst* mit dem ergänzenden Parameter */loginscripts* gestartet werden.

```
opsi-admin -d method config_createBool opsiclientd.event_user_login.active "user_login active" true
opsi-admin -d method config_createUnicode opsiclientd.event_user_login.action_processor_command "user_login \
    action_processor" "%action_processor.command% /loginscripts"
```

Als weiterer *opsi-winst* Parameter kann zusätzlich auch noch der Parameter */silent* verwendet werden, welcher die Anzeige des *opsi-winst* Fensters unterbindet.

```
opsi-admin -d method config_createUnicode opsiclientd.event_user_login.action_processor_command "user_login \
    action_processor" "%action_processor.command% /loginscripts /silent"
```