



opsi -script internal flow

Inhaltsverzeichnis

1	Uebersicht1	1
2	Uebersicht2	1
3	Uebersicht3	1
4	Sequentiell	1
5	Entscheidungen1	1
6	Entscheidungen2	2
7	Error detection	2
8	Error detection 1: Exitcode 1	2
9	Error detection 1: Exitcode 2	3
10	Error detection 1: Exitcode 3	3
11	Error detection 1: Exitcode 3a	3
12	Error detection 1: Exitcode 4	4
13	Error detection 1: Exitcode 5	4
14	Error detection 1: Exitcode 6	4
15	Error detection 2: markErrorNumber	5
16	Fehlerauswertung 1	5
17	Fehlerauswertung 2: isFatalError 1	5
18	Fehlerauswertung 2: isFatalError 2	5
19	Fehlerauswertung 3	6

1 Uebersicht1

- Sequentiell
- Entscheidungen
 - if then else endif
 - switch

2 Uebersicht2

- Error detection
 - exitcode
 - markErrorNumber / errorsOccurredSinceMark
- Reaktion auf Errors
 - LogError
 - isFatalError
 - isSuspended
 - do not run updatescript

3 Uebersicht3

CodeBlöcke

- sub Sektionen
 - externe Sub-Sektionen
- include
- setup / update
- lokale Funktionen

4 Sequentiell

Standard: Zeile für Zeile

5 Entscheidungen1

if / else / endif

Syntax:

```
if <boolean expression>
  <statement (s)>
[ else
  <statement (s)>]
endif
```

6 Entscheidungen2

switch

Syntax:

```
Switch <string expression>
  Case <string const>
    <statement(s)>
  EndCase
  [DefaultCase
    <statement(s)>
  EndCase]
EndSwitch
```

Beispiel:

```
switch $distrotype$
  case "redhat"
    set $package_lock_file$ = '/var/run/yum.pid.'
  endcase
  case "suse"
    set $package_lock_file$ = '/run/zypp.pid'
  endcase
  case "debian"
    set $package_lock_file$ = '/var/lib/dpkg/lock'
  endcase
  DefaultCase
    LogError "unknown Release"
  EndCase
endswitch
```

7 Error detection

Methoden:

```
getLastExitCode :string (exitcode) [W/L]
shellCall (<command string>) :string (exitcode) [W/L]
startProcess(<string>) :string (exitcode) [W/L]

markErrorNumber / errorsOccurredSinceMark [W/L]
LogError
```

8 Error detection 1: Exitcode 1

Methoden:

```
getLastExitCode :string (exitcode) [W/L]
```

- Die String-Funktion `getLastExitCode` gibt den ExitCode des letzten Prozessaufrufs der vorausgehenden WinBatch / DosBatch / ExecWith Sektion aus.
- Der Aufruf anderer opsi-winst Befehle (wie z.B. einer Files Sektion) verändert den gefundenen ExitCode nicht.
- Bei DosBatch und ExecWith Sektionen erhalten wir den Exitcode des Interpreters. Daher muss in der Regel der gewünschte Exitcode in der Sektion explizit übergeben werden.

9 Errordetection 1: Exitcode 2

cmd.exe

```
[DosInAnIcon_exit1]
rem create an errolevel= 1
VERIFY OTHER 2> NUL
exit %ERRORLEVEL%
```

bash

```
[ShellInAnIcon_Upgrade_ucs]
set -x
univention-upgrade --noninteractive --ignoreterm
exit $?
```

10 Errordetection 1: Exitcode 3

bash

```
[ShellInAnIcon_Upgrade_deb]
set -x
export DEBIAN_FRONTEND=noninteractive
apt-get update
apt-get --yes dist-upgrade
exit $?
```

bash

```
[ShellInAnIcon_Upgrade_deb]
set -x
export DEBIAN_FRONTEND=noninteractive
EXITCODE=0
apt-get update
EC=$?; if [ $EC -ne 0 ]; then EXITCODE=$EC; fi
apt-get --yes dist-upgrade
EC=$?; if [ $EC -ne 0 ]; then EXITCODE=$EC; fi
exit $EXITCODE
```

11 Errordetection 1: Exitcode 3a

cmd.exe

```
[ShellInAnIcon_exitcode1]
set exitcode=0
VERIFY OTHER 2> NUL
if %ERRORLEVEL% NEQ 0 set exitcode=%ERRORLEVEL%
echo "huhu"
if %ERRORLEVEL% NEQ 0 set exitcode=%ERRORLEVEL%
exit %exitcode%
```

12 Error detection 1: Exitcode 4

Methoden:

```
shellCall (<command string>) :string (exitcode) [W/L]
```

Beispiel:

```
set $exitcode$ = shellCall('net start')
```

Ist unter Windows eine Abkürzung für den Ausdruck:

```
DosInAnIcon_netstart winst /sysnative  
set $exitcode$ = getLastExitcode
```

```
[DosInAnIcon_netstart]  
net start
```

13 Error detection 1: Exitcode 5

```
set $exitcode$ = shellCall('ping -c 3 foo.bar')
```

Ist unter Linux eine Abkürzung für den Ausdruck:

```
shellInAnIcon_ping  
set $exitcode$ = getLastExitcode
```

```
[shellInAnIcon_netstart]  
ping -c 3 foo.bar || exit $?
```

14 Error detection 1: Exitcode 6

Methoden:

```
startProcess(<string>) :string (exitcode) [W/L]
```

Startet das Programm <string> als Prozess und liefert den Exitcode zurück.

```
set $exitcode$ = startProcess('setup.exe /S')
```

Ist eine Abkürzung für den Ausdruck:

```
Winbatch_setup  
set $exitcode$ = getLastExitcode
```

```
[Winbatch_setup]  
setup.exe /S
```

15 Error detection 2: markErrorNumber

Methoden:

```
markErrorNumber : noresult [W/L]
errorsOccurredSinceMark <relation> <integer> : boolean [W/L]
LogError <error - string> [W/L]
```

Beispiel:

```
markErrorNumber
logError "test error"
if errorsOccurredSinceMark > 0
    comment "error occured"
else
    comment "no error occured"
endif
```

16 Fehlerauswertung 1

Methoden:

```
isFatalError [W/L]
isSuspended [W/L]
isSuccess [W/L]
noUpdateScript [W/L]
```

17 Fehlerauswertung 2: isFatalError 1

```
isFatalError
isFatalError <short message>
```

Nach dem der Befehl aufgerufen wurde, werden keine Anweisungen mehr ausgeführt und als Skriptergebnis wird *failed* zurückgeliefert. Wird dieser Befehl nicht aufgerufen, so ist das Skriptergebnis *success*.

18 Fehlerauswertung 2: isFatalError 2

```
DefStringList $ErrorList$

Message "Configure depotadmin as depotadmin..."
ShellInAnIcon_config_depotadmin
if not("0" = getLastExitCode)
    LogError "failed config_depotadmin"
    set $fatal_error$ = "true"
    set $ErrorList$ = addtolist($ErrorList$, " failed config_depotadmin")
endif

if count($ErrorList$) > "0"
    logError "Error summary:"
    setloglevel = 6
    for %akterror% in $ErrorList$ do LogError "%akterror%"
```

```
endif  
  
if $fatal_error$ = "true"  
isFatalError  
endif
```

19 Fehlerauswertung 3

- `isSuccess` //since 4.11.3.7 [W/L]
Abort the script as successful.
- `noUpdateScript` //since 4.11.3.7 [W/L]
Do not run a update script after setup even if there is one.
- `isSuspended` //since 4.11.4.1 [W/L]
Abort the script without notice to the server. The action request remain unchanged.